Mobile Robots Gathering Algorithm with Local Weak Multiplicity in Rings

Tomoko IZUMI[1], Taisuke IZUMI[2], Sayaka KAMEI[3], and Fukuhito OOSHITA[4]

[1] College of Information Science and Engineering, Ritsumeikan University, Kusatsu, 525-8577 Japan.
izumi-t@fc.ritsumei.ac.jp
[2] Graduate School of Engineering, Nagoya Institute of Technology, Nagoya, 466-8555, Japan.
t-izumi@nitech.ac.jp
[3] Graduate School of Engineering, Hiroshima University, Higashi-Hiroshima, 736-8511, Japan
s-kamei@se.hiroshima-u.ac.jp
[4] Graduate School of Information Science and Technology, Osaka University, Suita, 565-0871, Japan
f-oosita@ist.osaka-u.ac.jp

Abstract. The gathering problem of anonymous and oblivious mobile robots is one of fundamental problems in the theoretical mobile robotics. We consider the gathering problem in unoriented and anonymous rings, which requires that all robots gather at a non-predefined node. Since the gathering problem cannot be solved without any additional capability to robots, all the previous works assume some capability of robots, such as accessing the memory on node. In this paper, we focus on the multiplicity capability. This paper presents a deterministic gathering algorithm with *local-weak* multiplicity, which provides the robot with the information about whether its current node has more than one robot or not. This assumption is strictly weaker than that by previous works. Moreover, we show that our algorithm is asymptotically time-optimal one, that is, the time complexity of our algorithm is $O(n)$, where n is the number of nodes. Interestingly, in spite of assuming the weaker assumption, it achieves significant improvement compared to the previous algorithm, which takes $O(kn)$ time for k robots.

1 Introduction

1.1 Background and Motivation

Mobile robots are the entities that cooperate with each other by computing, moving and communicating in a plane or in a network. The computational power of mobile robots with quite weak capability is attracting much attention of researchers in the field of distributed computing. In most of the studies, it is assumed that robots are oblivious (no memory to record past situations), anonymous (no IDs to distinguish two robots) and uniform (all robots run the same algorithm). In addition, it is also assumed that each robot has no direct means of communication. Typically, communication among two robots is done in the implicit way that each robot observes the environment, which includes the positions of other robots.

We consider the gathering problem of mobile robots, which is one of fundamental coordination tasks in the mobile robots system. The problem requires all robots to gather on a non-predefined location. Because of its simplicity, the gathering problem is actively studied in various settings. While most of previous works consider the gathering problem on two-dimensional Euclidean space [1, 2, 4, 7], a number of researches deal with it in graphs [3, 5, 6, 8, 9, 11, 10]. In this paper, we focus on unoriented anonymous rings. That is, we make all robots moving in a graph of ring topology gather on a non-predefined node. Unfortunately, regardless of its settings (graph or 2D-plane), it has been proved that the gathering problem is unsolvable in oblivious and anonymous robot systems without no additional assumption. All of the possibility results depend on some additional assumptions for capability of robots, such as distinct identifiers [10], accessing memory on nodes [3, 6, 11] or memory on robots [5].

The capability of robots considered in this paper is locality of *multiplicity detection*. The multiplicity detection specifies how each robot observes a node where two or more robots stay. In most of previous works, three types of multiplicity detection are considered: No multiplicity (each robot cannot distinguish the node with a single robot from that with multiple robots), weak multiplicity (each robot can detect whether the number of robots on a node is only one or more than one), and strong multiplicity (each robot can know the number of robots on a node). Recently, in addition to the above types, the notion of *locality* for multiplicity detection capability is introduced [7]. The local multiplicity detection implies that each robot can detect the multiplicity only for its current node. On the other hand, the global multiplicity allows each robot to detect the multiplicity of any node. In the original paper about gathering on ring [9] assumes global-weak multiplicity, and investigate the relationship between its feasibility and initial configurations: It presents a sufficient class of gatherable initial configurations, called *rigid configurations*, which is the set of configurations excluding one having a certain kind of symmetricity. It also proposes a gathering algorithm with global-weak multiplicity for any rigid configuration. However, the strategy of that algorithm strongly depends on the capability of global multiplicity. Its idea is to obtain exactly one node occupied by two or more robots and to make the others reach it. It is clear that this idea does not work correctly if we assume local multiplicity, because any robot on a node cannot detect the multiplicity of other nodes. In this sense, it is an interesting and non-trivial problem to reveal the set of initial configurations for which gathering is possible only with local multiplicity.

1.2 Our Contribution

In this paper, we investigate the feasibility of the gathering with local-weak multiplicity, which is strictly weaker setting than the original works by Klasing et. al. [9]. The contribution of this paper is that any rigid configuration is gatherable only with local-weak multiplicity detection. We propose a deterministic gathering algorithm in rings with k asynchronous robots ($2 < k \leq \lfloor n/2 \rfloor - 1$, where n is the number of nodes) that is applicable to any rigid configuration. Our algorithm assumes that robots are oblivious, anonymous and uniform, and that they have no device to communicate with others directly. To make the gathering problem solvable, we assume that robots have the local-weak multiplicity.

Moreover, the time complexity of our algorithm is also analyzed. In this paper. we evaluate time complexity based on maximum number of *asynchronous rounds* in the worst case, which is one of well-known model for measuring time complexity in asynchronous systems. The time complexity of our gathering algorithm with local-weak multiplicity is asymptotically optimal, that is, $O(n)$. Interestingly, it is significant improvement compared to the previous algorithm, which takes $O(kn)$ rounds [9]: in the previous algorithm, the robots first creates a configuration in which exactly one node v is occupied by two robots. Then, the robots which have no robot on the paths to v move to v, and other robots stay their current nodes. Since it takes $O(n)$ rounds to gather two robots to v, the time complexity of the algorithm is $O(kn)$.

1.3 Roadmap

In Section 2 we present the model of autonomous mobile robots considered in this paper, and introduce other necessary notations and definitions. The gathering algorithm with local-weak multiplicity and its time complexity are shown in Section 3. Finally we conclude this paper in Section 4.

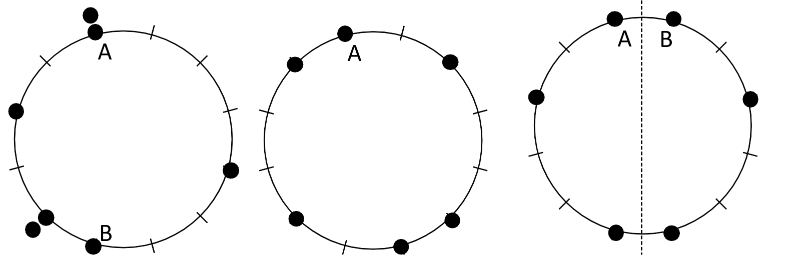
2 Preliminaries

2.1 System Models

The system consists of sets of nodes V and mobile robots R. The numbers of nodes and robots are denoted by $n = |V|$ and $k = |R|$ respectively. The nodes construct unoriented and undirected anonymous ring: Neither nodes nor links of the ring have any identifiers and labels. Some nodes of the ring are occupied by robots.

The robots are *anonymous* and *oblivious*. That is, each robot has no identifiers distinguishing itself and others, and cannot explicitly remember the history of its execution. In addition, no device for direct communication is equipped on each robot, such as marks which can be left on nodes and communication devices for sending messages. The cooperation of robots is done in an implicit manner: Each robot observes the configuration (i.e., the nodes occupied by other robots). Each robot executes the same deterministic algorithm in computational cycles (or briefly cycles). At the beginning of a cycle, each robot observes the current configuration and determines to stay idle or to move to one of adjacent nodes based on the algorithm. Then, if the robot decides to move, it moves to the adjacent node. We assume that the robot reaches the destination instantaneously. That is, when a robot observes the configuration, it sees all other robots at nodes and not on links. Cycles are performed asynchronously: The length of time for performing each operation is finite but unbounded. Due to these delay, when a robot moves to an adjacent node which is computed based on the last observation, some robots may stay at different nodes from those observed by the robot.

A configuration is defined by node on which each robot stays. The number of robots staying on a node is called the *multiplicity number* of the node. If the multiplicity number of node v is more than one, we say v is *multiple*. If v has one robot, v is said to be *single*. If there is no robot on a node, the node is called *free*. The *segment* $[v_p, v_q]$



a. a configuration b. a periodic configuration c. a symmetric configuration

Fig. 1. Examples of configurations: In figure a, the configuration is represented by ((4, 3, 1, 2, 2), (2, 1, 1, 2, 1)) which starts from node *A*. The periodic configuration in figure b is (2, 3, 1, 2, 3, 1), which is two copies of (2, 3, 1). The configuration in figure c has the axis of symmetry.

is defined by the sequence $(v_p, v_{p+1}, \ldots, v_q)$ of consecutive nodes in the ring, where nodes v_p and v_q are single or multiple and the others are free. The distance of segment $[v_p, v_q]$ is equal to the number of nodes in $[v_p, v_q]$ minus 1. Configuration C_i in which $[v_1, v_2], [v_2, v_3] \ldots, [v_w, v_1]$ are consecutive segments in a direction on the ring is defined by a pair of sequences $((d^i_1, d^i_2, \ldots, d^i_w), (m^i_1, m^i_2, \ldots, m^i_w))$, where d^i_h is the distance of $[v_h, v_{(h+1) \bmod w}]$ and m^i_h is the multiplicity number of v_h $(1 \leq h \leq w)$. We get different pairs of sequences when different segment is selected as the first segment or when the distances are listed in the opposite direction in the ring. These sequences represent the same configuration. In this paper, to simplify notation, we use one of the pairs of sequences which represent a configuration to denote the configuration. When no node is multiple in a configuration, we use only sequence $(d^i_1, d^i_2, \ldots, d^i_w)$.

A configuration is changed by movements of robots. Let C_0 be an initial configuration and M_i be a set of movements that occur simultaneously at configuration C_i. An execution is an alternate sequence of configurations and sets of movements $E = C_0, M_0, C_1, M_1, \ldots$, such that occurrence of movements M_i changes the configuration from C_i to C_{i+1}.

Configurations which have no multiple node are classified into three classes in [9]: Configuration C is called *periodic* if C is represented by a concatenation of at least two copies of a subsequence. If there is an axis of symmetry of the ring in configuration C, C is called *symmetric*. The last class is called *rigid*, in which the other configurations are included. Figure 1 shows examples of these configurations.

By observing a configuration, a robot gets a sequence of distances of segments and information about multiplicity number. Configuration C_i observed by a robot on node v is represented by two sequences of distances $(d^i_1, d^i_2, \ldots, d^i_w)$ and $(d^i_w d^i_{w-1}, \ldots, d^i_1)$ starting from v: One is the sequence in the opposite direction of the other. Since the ring is unoriented, each robot cannot know which is clockwise or counterclockwise one. In this paper, each robot chooses the larger one in lexicographic order: Sequence $(a_i, a_{i+1}, \ldots, a_j)$ is larger than $(b_i, b_{i+1}, \ldots, b_j)$ if there is h $(i \leq h \leq j)$ such that $a_l = b_l$ for $i \leq l \leq h - 1$ and $a_h > b_h$. The capacity of *multiplicity detection* specifies how each robot observes multiple nodes. In this paper, we consider the *local-weak multi-*

plicity: Each robot can detect whether the multiplicity number of its current node is more than one. The *view* of a robot on node v at configuration C_i is represented by $s_v(C_i) = (\max\{(d^i_1, d^i_2, \ldots, d^i_w), (d^i_w, d^i_{w-1}, \ldots, d^i_1)\}, m^i)$, where m^i is true if the multiplicity number of node v is more than one and false otherwise. Note that the sequences of distance in views of different robots may be in different direction on the ring. For example, in Figure 1. a., the view of the robots on node A is ((4, 3, 1, 2, 2), true) and that on B is ((3, 4, 2, 2, 1), false). In [9], it is said that a configuration without multiple nodes is rigid when the views of all the robots are different.

2.2 Gathering Problem

The goal of the gathering problem is to gather all the robots on one node that is not predefined and to keep the configuration. For the gathering problem, some impossibility results are shown in [9].

Theorem 1. *The gathering problem is insolvable for the following cases:*

1. *The number of robots is two.*
2. *Robots have no multiplicity detection.*
3. *The initial configuration is any periodic configuration.*
4. *The initial configuration is any symmetric configuration in which axis of symmetry goes through two antipodal links.*

In this paper, we assume that initial configuration is rigid and that $2 < k \leq \lfloor n/2 \rfloor - 1$. That is, in an initial configuration, there is no multiple node and each robot has different view.

The algorithms in this paper are described in some rules. Each rule consists of some conditions and actions of the robots. The robots observe the current configuration and execute actions of one of the rules whose condition matches the observing configuration. Notice that some conditions are described nested if-then-else statements.

3 Gathering Algorithm with Local-Weak Multiplicity

In this section, we present a gathering algorithm with the local-weak multiplicity for any rigid configuration with $2 < k \leq \lfloor n/2 \rfloor - 1$ robots.

Before presenting the details of the algorithm, we introduce several definitions and notation. The *maximum segment* at configuration C_i is the segment with the longest distance in C_i. The *maximum node* at C_i is the non-free node on which the view of robot is the *maximum view* at C_i. If only one node is the maximum one in C_i, the maximum node is denoted by v^i_1 and each non-free node is labeled in the same order as the distance sequence in the view on v^i_1. That is, for view $s_{v^i_1}(C_i) = ((d^i_1, d^i_2, \ldots, d^i_w), m^i)$, node v^i_h is the node such that the distance of segment $[v^i_h, v^i_{h+1}]$ is d^i_h and called h-th node. In this case, for simplicity, distance d^i_h implies the h-th element of the distance sequence on the maximum node.

From the definitions, the following lemma is trivial.

Lemma 1. *Let C_i be a configuration without multiple nodes. The number of the maximum nodes at configuration C_i is one if and only if C_i is rigid.*

In what follows, we present our algorithm in some subsections. In Section 3.1, the gathering algorithm for any rigid configuration C_i where $d_1^i \geq 4$ and $d_2^i \geq 3$ is introduced. If a given initial configuration does not match the above conditions, the robots try to create the configuration satisfying them: The algorithm presented in Section 3.2 creates a desired configuration from a rigid one where $d_1^i \geq 4$ and $d_2^i < 3$, and in Section 3.3, we present the algorithm for creation of a rigid configuration where $d_1^i = 4$ from a rigid one where $d_1^i = 3$. In this paper, we assume that $k \leq \lfloor n/2 \rfloor - 1$ and that initial configuration C_0 is rigid. That is, the distance of the maximum segment is longer than 2 at C_0.

3.1 Gathering from a rigid configuration where $d_1^i \geq 4$ and $d_2^i \geq 3$

In this subsection, we assume that the initial configuration C_0 is rigid and satisfies $d_1^0 \geq 4$ and $d_2^0 \geq 3$.

To gather only with the local-weak multiplicity, we must lead a configuration where one node has $k-1$ robots and the other has 1 robot when the number of non-free nodes is 2. The reason is that when two nodes are multiple and the others are free, the robots on the two nodes take the symmetric movements because they have the same view. Hence, when the two nodes are neighboring, all the robots keep staying the current nodes or passing each other. Thus, the robots cannot gather on one node. The idea of our algorithm is that node v_2^0 is kept single, and that the robots on nodes v_1^0 and $v_4^0, \ldots, v_w^0$ gather to node v_3^0. The key to achieve this strategy is that nodes v_2^0 and v_3^0 are kept as the second and the third nodes respectively during the execution.

The details of our algorithm are as follows

- In the case that the number w of non-free nodes is more than or equal to 3,
  - **R1** : When $d_w^i \geq 2$, robots on v_1^i move to v_w^i.
  - **R2** : When $w \neq 3$, $d_w^i = 1$, $d_{w-1}^i \geq 2$,
    - **R2-1** : if the maximum segment is only $[v_1^i, v_2^i]$ then robots on v_w^i move to v_{w-1}^i,
    - **R2-2** : otherwise robots on v_2^i move to v_3^i.
  - **R3** : When $w \neq 3$ and $d_w^i = 1$ and $d_{w-1}^i = 1$, robots on v_1^i move to v_w^i.
  - **R4** : When $w = 3$ and $d_3^i = 1$, robots on v_1^i move to v_3^i.
- In the case that the number w of non-free nodes is 2,
  - **R5** : robot on the single node moves to the other node occupied by robots.

First, let consider an initial configuration where the number of the maximum segments is one. In our algorithm, the robots move so that the first element on the maximum node is increased, the second element is not changed and the others are decreased. In addition, the last element is kept shorter than the second element. That is, during the execution, the number of the maximum segments is one, and node v_2^0 remains the second node not the first. Figure 2 illustrates an example of executions of the algorithm. The robots on node v_1^i move to the neighboring node of the last node (rule **R1**). After that, the robots on v_w^i move to the other neighboring node of v_w^i if the neighbor is free (rule **R2-1**). Then, the distance of the last segment becomes 2, and rule **R1** is executed again.

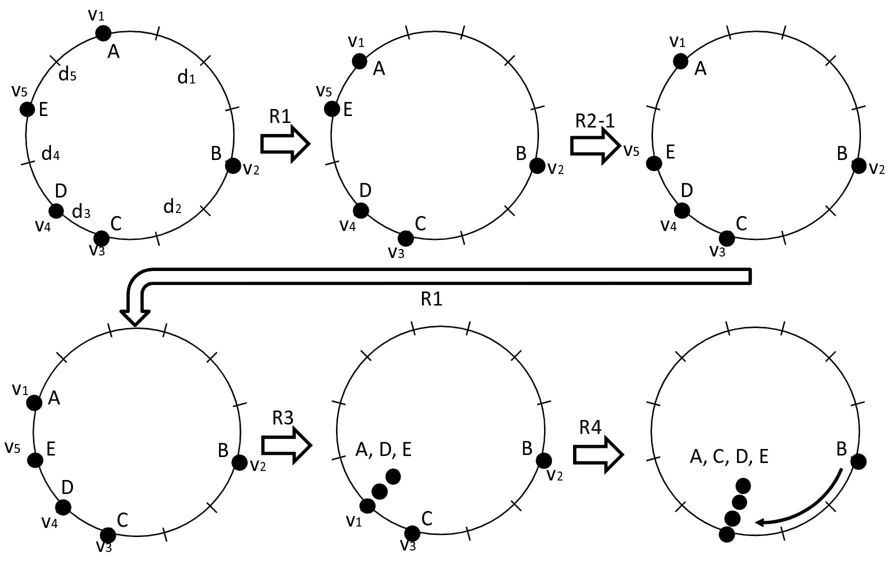


Fig. 2. An example of executions of the gathering algorithm

That is, the last element of the maximum view is kept shorter than the second element, which is larger than or equal to 3. When d^j_{w-1} becomes 1, the robots on v^j_1 join on v^j_w (rule **R3**). By repeating these actions, the configuration eventually becomes one where $w = 3$ and $d^h_3 = 1$, and then, all the robots except one on the second node gather on v^h_3 (rule **R4**). That is, a configuration where one node is single and the other is multiple is created. Lastly, the robot on the single node moves to the multiple node.

We consider the initial configuration C_0 such that more than one the maximum segments exist. In this case, one of rules **R1**, **R2-2** and **R3** is applied. At the next configuration C_1 after executing one of them, the first element on the maximum node v^0_1 is extended by one. That is, the number of the maximum segments becomes one at C_1, and then the robots gather on one node by the above way. However, only when rule **R2-2** is applied, the second element of v^0_1 may become less than 3. It does not satisfy the conditions for execution of this algorithm. In this case, the algorithm presented in Section 3.2, in which the second element becomes 3 keeping only the first segment having the maximum distance, is executed.

Lemma 2. *For $k(2 < k \leq \lfloor n/2 \rfloor - 1)$ robots with the local-weak multiplicity, the algorithm achieves the gathering from a rigid configuration where $d^0_1 \geq 4$ and $d^0_2 \geq 3$.*

Proof. If the configuration becomes one where $w = 3$, $d^i_3 = 1$ and the second node is single, it is clear that rules **R4** and **R5** lead to gathering.

In what follows, we explain that the desired configuration is created by rules **R1**-**R3**. The important fact to show this lemma is that the maximum node is only one and the second node is not changed during the execution.

First, we show that in any execution E from a given initial configuration, there is a configuration C_j satisfying the following three conditions: 1) exactly one node v_1^j is the maximum node, 2) only the first segment on v_1^j is the maximum one and 3) $d_2^j \geq 3$. The initial configuration C_0 is rigid. It means that exactly one node is the maximum node (Lemma 1). We consider each case that each rule is appied.

- When rule **R2-1** is applied at C_0, C_0 satisfies the above conditions.
- When rule **R1** or **R3** is executed at C_0, the robot on v_1^0 moves to adjacent node u and the others stay on their current node. Segment $[u, v_2^0]$, whose distance is $d_1^0 + 1$, is only the maximum one because there is no segment whose distance is longer than d_1^0 at C_0. Thus, the first condition is satisfied. The distance of the other segment including u is $d_w^0 - 1$ or 1, and it holds that $d_w^0 \leq d_2^0 = d_2^1$. Thus, node u is only the maximum node and $d_2^1 \geq 3$ at configuration C_1, that is, C_1 satisfies the three conditions.
- When rule **R2-2** is applied at C_0, the robot on v_2^0 moves to adjacent node and the robot on v_1^0 does not move. At the next configuration C_1, since the view on v_1^0 is represented by $(d_1^0 + 1, d_2^0 - 1, \ldots, 1)$, the segment with $d_1^0 + 1$ distance is only the maximum one and v_1^0 is only the maximum node. However, d_2^1 may become less than 3. In this case, the algorithm in Section 3.2 is executed at C_1. Lemma 4 and 5 guarantee that the algorithm leads to a configuration C_j in which $d_2^j = 3$ keeping node v_1^1 and segment $[v_1^1, v_2^1]$ being only the maximum one. That is, configuration C_j satisfies the above three conditions, and rule **R2-2** is not executed at C_j.

Next, we prove that for any configuration $C_i(j \leq i)$ during E, the next configuration C_{i+1} satisfies the following three conditions; 1) only the segment including $[v_1^i, v_2^i]$ has the maximum distance, 2) $v_2^{i+1} = v_2^i$ and 3) $3 \leq d_2^{i+1}$. This implies that the maximum node is only one, node v_2^i remains the second node, and rule **R2-2** is not executed at C_{i+1}. By executing one of rules **R1**, **R2-1** and **R3** at C_i, some of the robots on v_1^i or v_w^i move to adjacent node. We denote the segment including v_1^i and v_2^i at $C_{i+1}$by $[u, v_2^i]$, where u is v_1^i or the neighboring node of v_1^i. Due to the movements, the distance of $[u, v_2^i]$ becomes $d_1^i + 1$ or d_1^i, and each distance of the other segments is decreased, not changed, or kept shorter than 3. Since d_1^i is longer than 3, segment $[u, v_2^i]$ is only the maximum one. Hence, the candidates of the maximum nodes at C_{i+1} are u or v_2^i. Notice that rules **R1**, **R2-1** and **R3** do not change the distance of $[v_2^i, v_3^i]$, and that the other segment neighboring of $[u, v_2^i]$ is shorter than 3 or becomes $d_w^i - 1$. Since $d_w^i \leq d_2^i$ and $3 \leq d_2^i$, the view on v_2^i is smaller than one on u. Thus, node v_2^i remains the second node and $3 \leq d_2^{i+1}$ at C_{i+1}.

At any configuration in the execution until the number of non-free nodes is 2, the second node v_2^j is not changed. From the algorithm, since no robot moves to v_2^j, node v_2^j keeps single. When the last element of view on the maximum node is longer than 2, rule **R1** is applied, and then, the last element eventually becomes 1. After that, rules **R2-1** and **R1** are alternately applied and d_{w-1}^i becomes 1. At this configuration, rule **R3** decrements the number w of nodes occupied by robots. Hence, the configuration becomes eventually one where $w = 3$, $d_3^i = 1$ and the second node is single. $\square$

3.2 Algorithm for a rigid configuration where $d_1^i \geq 4$ and $d_2^i < 3$

In this subsection, we consider the given configuration is rigid and satisfies $d_1^i \geq 4$ and $d_2^i < 3$. The goal of the algorithm presented here is to make a rigid configuration where $d_1^j \geq 4$ and $d_2^j = 3$ from the given configuration.

To present the algorithm, we introduce procedure SHIFT. SHIFT is executed to shift the robots on neighboring nodes in a direction on the ring without creating multiple nodes. We assume that configuration C_i at which SHIFT is executed satisfies the following conditions:

1. C_i is rigid where $d_1^i > d_2^i$.
2. There are nodes v_p^i and v_q^i such that $2 \leq d_x^i < d_1^i (1 < x < p)$, $d_y^i = 1 (p \leq y < q)$ and $d_q^i \geq 2$.

If SHIFT is called then the next rule is executed. By executing SHIFT continuously, the robot on node v_p^i can move away from v_{p-1}^i without creating multiple node.

SHIFT: robot on node v_q^i moves to v_{q+1}^i.

Lemma 3. *Assume that Procedure* SHIFT *is called at a rigid configuration C_i satisfying the two conditions. The configuration C_{i+1} caused by executing* SHIFT *is rigid, and node v_1^i is also the maximum node at C_{i+1}.*

Proof. The execution of SHIFT does not make any node multiple because the robot on v_q^i gets close to v_{q+1}^i and and $d_q^i \geq 2$.

We show that node v_1^i is only the maximum node at C_{i+1}. Let u be the node to which the robot on v_q^i moves. By executing SHIFT, the $(q-1)$-th element of the view on v_1^i is incremented and the q-th element is decremented by one. So, the view on v_1^i is larger than the previous one. If a node has larger view than or equal to v_1^i at C_{i+1} then the incremented element must be h-th element of the view on the node ($h \leq q-1$). In addition, except node u, the view which is opposite direction to one on v_1^i is smaller than the previous one because the decremented element is preceding the incremented element in the view. That is, the other candidates of the maximum nodes are $v_2^i, \ldots, v_{q-1}^i$ and u. For each node $v_2^i, \ldots, v_{p-1}^i$, if the view on the node is the same direction as one on v_1^i then the conditions on which SHIFT is executed imply that its first element is smaller than d_1^i. Similarly, the first element of the view on each node $v_p^i, \ldots, v_{q-2}^i$ is 1 and one for v_{q-1}^i is 2. For node u, its first element is 2 or $d_q^i - 1$. From the fact that $d_1^i \geq 3$ and $d_1^i \geq d_q^i$, these views are smaller than one on v_1^i at C_{i+1}. Hence, node v_1^i is only the maximum node at C_{i+1}. From Lemma 1, C_{i+1} is a rigid configuration. □

Now, we explain the algorithm for making a configuration where $d_2^i = 3$. The algorithm is very simple: Robot on node v_3^i moves to the neighboring node in order to extend the distance from node v_2^i. If there is a robot on the neighboring node then SHIFT is executed until the robot on v_3^i can move to the neighbor without creating multiple node. The details of the algorithm are as follows:

– In the case that $d_1^i \geq 4$ and $d_2^i < 3$ at configuration C_i,

R6: When $d^i_3 \geq 2$, robot on node v^i_3 moves away from v^i_2.
R7: When $d^i_3 = 1$, robot executes SHIFT.

Lemma 4. *Assume that C_i is a rigid configuration where $d^i_1 \geq 4$ and $d^i_2 < 3$. Rules* **R6** *and* **R7** *make a rigid configuration C_j where $d^j_1 \geq 4$ and $d^j_2 = 3$ from C_i. And the maximum node v^i_1 at C_i remains the maximum one.*

Proof. We first show that the next configuration C_{i+1} is rigid and the maximum node v^i_1 at C_i remains the maximum one at C_{i+1}. If rule **R7** is applied, Lemma 3 proves this fact. If rule **R6** is applied, the first element of the view on v^i_1 is not changed and the second element is incremented. That is, the view on v^i_1 becomes larger than the previous one. Let u be the node to which the robot on v^i_3 moves. The other candidates of the maximum nodes are nodes whose views contain the incremented element as the first or the second element. In addition, the direction of the candidate's view must be the same direction as one on v^i_1 except node u. That is, the other candidates are nodes v^i_2 and u. For node v^i_2, if its view is the same direction as one on v^i_1 then its first element is incremented but its value is smaller than 4. Similarly, the first element on node u is smaller than 4 or $d^i_3 - 1$. Since $d^i_1 \geq 4$ and $d^i_1 \geq d^i_3$, they cannot be the maximum node. Thus, node v^i_1 remains only the maximum node, and C_{i+1} is rigid.

If the configuration satisfies the conditions of rule **R6**, the second element of v^i_1 is incremented. Otherwise, rule **R7** is applied until the third element of v^i_1 becomes longer than 1, which is a configuration at which rule **R6** is executed. Therefore, the second element of v^i_1 eventually becomes 3. □

From the proof of Lemma 4, some segments is incremented but its distance is smaller than 4 during the execution of the algorithm presented in this subsection. Since the distance of the first segment of the maximum node is longer than or equal to 4. Thus, the following lemma is proved.

Lemma 5. *Let C_i be a rigid configuration where the first segment of the view on v^i_1 is only the maximum one, $d^i_1 \geq 4$ and $d^i_2 < 3$. During the execution of rules* **R6** *and* **R7** *until d^i_2 becomes 3, the first segment on v^i_1 is kept being only the maximum one.*

3.3 Algorithm for a rigid configuration where $d^i_1 = 3$

In this subsection, we present the algorithm to make a rigid configuration where $d^i_1 = 4$ from the given configuration where $d^i_1 = 3$.

The details of the algorithm are as follows:

- In the case that $d^i_1 = 3$ at the rigid configuration,
  R8 : When $d^i_w \neq 1$, robot on node v^i_1 moves to node v^i_w.
  R9 : When $(d^i_2, d^i_w) = (1, 1)$, robot executes SHIFT.
  R10 : When $(d^i_2, d^i_w) = (2, 1)$,
  R10-1 : if configuration $(4, 1, d^i_3, \ldots, d^i_{w-1}, 1)$ is asymmetric then robot on node v^i_2 moves to node v^i_3,
  R10-2 : else if $d^i_3 = 1$ then robot executes SHIFT,

R10-3 : else if configuration $(3, 3, d_3^i - 1, \ldots, d_{w-1}^i, 1)$ is asymmetric then robot on node v_3^i moves to node v_4^i,

R10-4 : otherwise robot on node v_2^i moves to node v_1^i.

R11 : When $(d_2^i, d_w^i) = (3, 1)$, robot on node v_2^i moves to node v_3^i.

In the algorithm, if the robots on node v_2^i or v_w^i can move to the neighboring node without creating a multiple node or a symmetric configuration, the distance d_1^i is extended by the movements of these robots (rule **R8** or **R11**). If both of robots on v_2^i and v_w^i cannot move, that is, nodes v_2^i and v_w^i are neighbor v_3^i and v_{w-1}^i respectively, then SHIFT is executed to extend distance d_2^i by 2 (rule **R9**). The most sensitive case is that $d_2^i = 2$ and $d_w^i = 1$ because a movement of the robot on v_2^i may lead a symmetric configuration. In the above algorithm, to avoid symmetric configurations, the robots behave different actions in the four cases (rules **10-1** to **10-4**).

Lemma 6. *The algorithm makes a rigid configuration C_i where $d_1^i = 4$ from a rigid configuration where $d_1^i = 3$.*

Proof. We show that when one of the rules is applied at configuration C_i where $d_1^i = 3$, the next configuration C_{i+1} is rigid.

- When the applied rule is **R8** or **R11**, the distance of the segment including v_1^i and v_2^i becomes 4. Since there is no other segment whose distance is 4 at C_{i+1}, the candidates of the maximum nodes are end nodes of the maximum segment. Compared the distances of the neighboring segments of the maximum one, one is smaller than the other. Thus, the maximum node is one of the two end nodes, and C_{i+1} is rigid.
- When rule **R9** or **R10-2** is applied, the next configuration is rigid from Lemma 3.
- When rule **R10-1** is applied, the next configuration $C_{i+1} = (4, 1, d_3^i, \ldots, d_{w-1}^i, 1)$ is asymmetric from the condition for executing **R10-1**. Since the segment whose distance is 4 is only one, the configuration is not periodic. Therefore, C_{i+1} is rigid.
- When rule **R10-3** is applied, it holds that $d_3^i \geq 2$. The next configuration $C_{i+1} = (3, 3, d_3^i - 1, \ldots, d_{w-1}^i, 1)$ has no multiple node and is asymmetric. Since v_1^i is only the maximum node at C_i, there are no other consecutive segments whose distances are 3. That is, C_{i+1} is not periodic. Thus, C_{i+1} is rigid.
- The last case is that rule **R10-4** is executed at C_i. Rule **R10-4** is executed when the configuration does not satifsy the conditions of rules **R10-1** and **R10-3**. It means that configurations $(4, 1, d_3^i, \ldots, d_{w-1}^i, 1)$ and $(3, 3, d_3^i - 1, \ldots, d_{w-1}^i, 1)$ are symmetric, where $d_3^i \geq 2$. In this case, since $(3, 3, d_3^i - 1, \ldots, d_{w-1}^i, 1)$ is symmetric, $d_3^i = 2$. Then, since $(4, 1, 2, \ldots, d_{w-1}^i, 1)$ is symmetric, d_{w-1}^i must be 2. By repeating this way, we know that $d_h^i (3 \leq h \leq w - 1)$ is 2 (see Figure 3). Therefore, configuration C_{i+1} after executing **R10-4** at C_i is $(2, 3, 2, \ldots, 2, 1)$, which is a rigid configuration.

Next, we show that the first element of the maximum view becomes 4. If rule **R8**, **R10-1** or **R11** is applied then it is clear that $d_1^{i+1} = 4$. When rule **R9** is applied and SHIFT is executed repeatedly, the second element eventually becomes 2, in which one of rules **R10** is applied. In the case of rule **R10-2**, by executing SHIFT repeatedly, the configuration becomes one in which **R10-1** is executed, or d_3^j becomes 2 ($j > i$). In the latter case, some robot moves based on one of rules **R10-1**, **R10-3** and **R10-4**. When rule **R10-3** is applied, rule **R11** is executed at the next configuration C_{i+1}, and when rule **R10-4** is applied, rule **R8** is executed at C_{i+1}. □

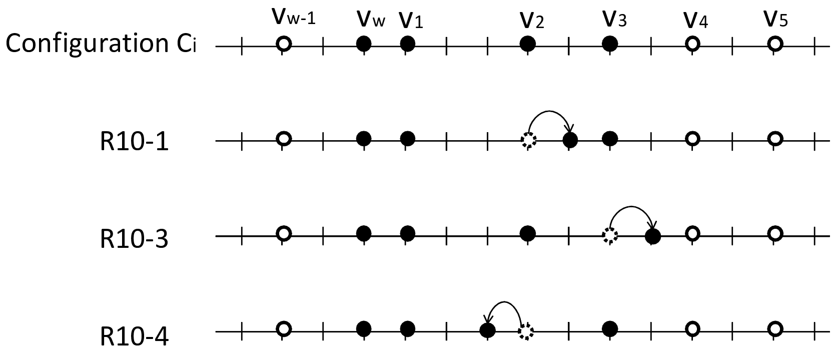


Fig. 3. Each configuration after each execution of **R10-1**, **R10-3** and **R10-4**.

From Lemmas 2, 4 and 6, we prove the following theorem.

Theorem 2. *The gathering is achieved from a rigid configuration with* $k(2 < k \leq \lfloor n/2 \rfloor - 1)$ *robots in the system with the local-weak multiplicity.*

3.4 Time Complexity

We show that our algorithm for the gathering with the local-weak multiplicity is $O(n)$-time algorithm in this subsection. In asynchronous systems, time complexity is usually measured in terms of maximum number of *asynchronous rounds* in the worst case. An asynchronous round is defined as the shortest fragment of an execution in which each robot is activated, and moves to the neighboring node or stays the current node at least once.

At procedure SHIFT, at least one robot, which is one on node v^i_q, moves to the neighboring node during one round. That is, it takes $O(h)$ time to extend distance d^i_p from a configuration where $d^i_p, \ldots, d^i_{p+h} = 1$. Since the number of robots is k, value of h is at most k. In each algorithm presented in Section 3.3 and 3.2, consecutive execution of SHIFT occurs at most 2 times. Thus, in the algorithms, it takes $O(k)$ time to execute SHIFT. From the proofs of Lemma 4 and 6, the each algorithm in Section 3.3 and 3.2 leads each desired configuration by applying at most 2 rules except SHIFT. Therefore, the algorithms in Section 3.3 and 3.2 take $O(k)$ time.

In the gathering algorithm in Section 3.1, it takes $O(k)$ time to make a configuration where only one segment is maximum because SHIFT is executed due to rule **R2-2**. After that, the robots on the maximum node take one node toward node v^i_3 at two rounds even if the robots is activated at the last of the rounds. Thus, the time complexity to gather all the robots except the robot on v^i_2 is $O(n)$. Therefore, the following theorem is proved.

Theorem 3. *The algorithm achieves the gathering in* $O(n)$ *time.*

4 Conclusions

This paper presented the deterministic gathering algorithm in asynchronous rings with oblivious and anonymous robots using the local-weak multiplicity. The time complexity of our algorithm is $O(n)$, while that of the algorithms with global-weak multiplicity in the previous works are $O(kn)$. This result implies the our algorithm is asymptotically time-optimal one.

In this paper, we restrict the number of robots so that $2 < k \leq \lfloor n/2 \rfloor - 1$. When there are more than $\lfloor n/2 \rfloor - 1$ robots on a ring, the distance between every consecutive robots may be less than or equal to 2 at a initial configuration. In this case, it is hard that each robot moves to adjacent node without creating a multiple node or a symmetric configuration. Our feature work is to present the algorithm or to prove the impossibility of the gathering in this situation. Moreover, the gathering with the local-weak multiplicity from a symmetric configuration is also one of interesting problems.

Acknowledgment This work is supported in part by The Telecommunication Advancement Foundation and Grand-in-Aid for Young Scientists ((B)19700075) of JSPS.

References

1. N. Agmon and D. Peleg. Fault-tolerant gathering algorithms for automonous mobile robots. *SIAM Journal of Computing*, 36(1):56–82, 2006.
2. M. Cieliebak, P. Flocchini, G. Prencipe, and N. Santoro. Solving the robots gathering problem. In *Proceedings of the 30th International Colloquium on Automata, Languages and Programming (ICALP'03)*, pages 1181–1196, 2003.
3. P. Flocchini, E. Kranakis, D. Krizanc, C. Sawchuk, and N. Santoro. Multiple mobile agents rendezvous in a ring. In *Proceedings of the 6th Latin American Theoretical Informatics(LATIN'04)*, pages 599–608, 2004.
4. P. Flocchini, G. Prencipe, N. Santoro, and P.Widmayer. Gathering of asynchnorous robots with limited visibility. *Theoretical Computer Science*, 337(1-3):147–168, 2005.
5. P. Fraigniaud and A. Pelec. Deterministic rendezvous in trees with little memory. In *Proceedings of the 22nd International Symposium on Distributed Computing (DISC'08)*, pages 242–256, 2008.
6. L. Gasieniec, E. Kranakis, D. Krizanc, and X. Zhang. Optimal memory rendezvous of anonymous mobile agents in a uni-directional ring. In *Proceedings of the 32th International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM'06)*, pages 282–292, 2006.
7. T. Izumi, T. Izumi, S. Kamei, and F. Ooshita. Randomized gathering of mobile robots with local-multiplicity detection. In *Proceedings of the 11th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS'09)*, pages 384–398, 2009.
8. R. Klasing, A. Kosowski, and A. Navarra. Taking advantage of symmetrics: Gathering of asynchronous oblivious robots on a ring. In *Proceedings of the 12th International Conference on Principles of Distributed Systems (OPODIS'08)*, pages 446–462, 2008.
9. R. Klasing, E. Markou, and A. Pelc. Gathering asynchnorous pblivious mobile robots in a ring. *Theoretical Computer Science*, 390(1):27–39, 2008.
10. D. Kowalski and A. Pelec. Polynomial deterministic rendezvous in arbtrary graphs. In *Proceedings of the 15th Annual Symposium on Algorithms and Computation (ISAAC'04)*, pages 644–656, 2004.

11. E. Kranakis, D. Krizanc, N. Santoro, and C. Sawchuk. Mobile agent rendezvous in a ring. In *Proceedings of the 23th International Conference on Distributed Computing Systems (ICDCS'03)*, pages 592–599, 2003.