Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Partial gathering of mobile agents in asynchronous unidirectional rings[☆,☆☆]

Masahiro Shibata[*], Shinji Kawai[*], Fukuhito Ooshita[*], Hirotsugu Kakugawa[*], Toshimitsu Masuzawa[*]

Graduate School of Information Science and Technology, Osaka University, 1-5 Yamadaoka, Suita, Osaka 565-0871, Japan

A R T I C L E I N F O

Article history:
Received 2 December 2014
Received in revised form 10 September 2015
Accepted 11 September 2015
Available online 25 September 2015
Communicated by D. Peleg

Keywords:
Distributed system
Mobile agent
Gathering problem
Partial gathering

A B S T R A C T

In this paper, we consider the partial gathering problem of mobile agents in asynchronous unidirectional rings equipped with whiteboards on nodes. The partial gathering problem is a new generalization of the total gathering problem. The partial gathering problem requires, for a given integer g, that each agent should move to a node and terminate so that at least g agents should meet at the same node. The requirement for the partial gathering problem is weaker than that for the (well-investigated) total gathering problem, and thus, we have interests in clarifying the difference on the move complexity between them. We propose three algorithms to solve the partial gathering problem. The first algorithm is deterministic but requires unique ID of each agent. This algorithm achieves the partial gathering in $O(gn)$ total moves, where n is the number of nodes. The second algorithm is randomized and requires no unique ID of each agent (i.e., anonymous). This algorithm achieves the partial gathering in expected $O(gn)$ total moves. The third algorithm is deterministic and requires no unique ID of each agent. For this case, we show that there exist initial configurations in which no algorithm can solve the problem and agents can achieve the partial gathering in $O(kn)$ total moves for solvable initial configurations, where k is the number of agents. Note that the total gathering problem requires $\Omega(kn)$ total moves, while the partial gathering problem requires $\Omega(gn)$ total moves in each model. Hence, we show that the move complexity of the first and second algorithms is asymptotically optimal.

1. Introduction

1.1. Background and our contribution

A *distributed system* is a system that consists of a set of computers (*nodes*) and communication links. In recent years, distributed systems have become large and design of distributed systems has become complicated. As an effective way

[☆] The conference version of this paper is published in the proceedings of 16th International Conference on Principles of Distributed Systems (OPODIS 2012).

[☆☆] This work was supported by JSPS KAKENHI Grant Numbers 24500039, 24650012, 25104516, 26280022, and 26330084.

[*] Corresponding authors. Tel.: +81 6 6879 4117; fax: +81 6 6879 4119.
E-mail addresses: m-sibata@ist.osaka-u.ac.jp (M. Shibata), f-oosita@ist.osaka-u.ac.jp (F. Ooshita), kakugawa@ist.osaka-u.ac.jp (H. Kakugawa), masuzawa@ist.osaka-u.ac.jp (T. Masuzawa).

http://dx.doi.org/10.1016/j.tcs.2015.09.012

Table 1
Proposed algorithms for the g-partial gathering problem in asynchronous unidirectional rings.

	Model 1 (Section 3)	Model 2 (Section 4)	Model 3 (Section 5)
Unique agent ID	Available	Not available	Not available
Deterministic/Randomized	Deterministic	Randomized	Deterministic
Knowledge of k	Not available	Available	Available
The total moves	$O(gn)$	$O(gn)$	$O(kn)$
Note	–	–	There exist unsolvable configurations

to design distributed systems, (mobile) agents have attracted a lot of attention [1]. Design of distributed systems can be simplified using agents because they can traverse the system with carrying information and process tasks on each node.

The total gathering problem is a fundamental problem for cooperation of agents [1–3]. The total gathering problem requires all agents to meet at a single node in finite time. The total gathering problem is useful because, by meeting at a single node, all agents can share information or synchronize behaviors among them.

In this paper, we consider a new generalization of the total gathering problem, called the *partial gathering problem*. The partial gathering problem does not always require all agents to gather at a single node, but requires agents to gather partially at several nodes. More precisely, we consider the problem which requires, for a given integer g, that each agent should move to a node and terminate at a node so that at least g agents should meet at the node. We define this problem as the *g-partial gathering problem*. We assume that k is the number of agents. Clearly if $k/2 < g \leq k$ holds, the g-partial gathering problem is equivalent to the total gathering problem. If $g \leq k/2$ holds, the requirement for the g-partial gathering problem is weaker than that for the total gathering problem, and thus it seems possible to solve the g-partial gathering problem with fewer total moves. From a practical point of view, the g-partial gathering problem is still useful because agents can share information and process tasks cooperatively among at least g agents.

The contribution of this paper is to clarify the difference on the move complexity between the total gathering problem and the g-partial gathering problem. We investigate the difference in asynchronous unidirectional rings equipped with whiteboards on nodes. The contribution of this paper is summarized in Table 1, where n is the number of nodes.

First, we propose a deterministic algorithm to solve the g-partial gathering problem for the case that agents have distinct IDs. This algorithm requires $O(gn)$ total moves. Second, we propose a randomized algorithm to solve the g-partial gathering problem for the case that agents have no IDs but agents know the number k of agents. This algorithm requires expected $O(gn)$ total moves. Third, we consider a deterministic algorithm to solve the g-partial gathering problem for the case that agents have no IDs but agents know the number k of agents. In this case, we show that there exist initial configurations for which the g-partial gathering problem is unsolvable. Next, we propose a deterministic algorithm to solve the g-partial gathering problem for any solvable initial configuration. This algorithm requires $O(kn)$ total moves. Note that the total gathering problem requires $\Omega(kn)$ total moves regardless of deterministic or randomized settings. This is because in the case that all the agents are uniformly deployed, at least half agents require $O(n)$ moves to meet at one node. Hence, the first and second algorithms imply that the g-partial gathering problem can be solved with fewer total moves compared to the total gathering problem for the both cases. In addition, we show a lower bound $\Omega(gn)$ of the total moves for the g-partial gathering problem if $g \geq 2$. This means the first and second algorithms are asymptotically optimal in terms of the total moves.

1.2. Related works

Many fundamental problems for cooperation of mobile agents have been studied. For example, the searching problem [4], the gossip problem [5], the election problem [6], and the gathering problem [1–3,6–16] have been studied.

In particular, the gathering problem has received a lot of attention and has been extensively studied in many topologies including trees [1,5,12–15], tori [1,8], and rings [1–3,6,7,9,4,11]. The gathering problem for rings has been extensively studied because algorithms for such highly symmetric topologies give techniques to treat the essential difficulty of the gathering problem such as breaking symmetry.

For example, Kranakis et al. [2] considered the gathering problem for two mobile agents in ring networks. This algorithm allows each agent to use a token to select the gathering node based on the token locations. Later this work has been extended to consider any number of agents [3,7]. Flocchini et al. [3] showed that if one token is available for each agent, the lower bound on the space complexity per agent is $\Omega(\log k + \log\log n)$ bits, where k is the number of agents and n is the number of nodes. Later, Gasieniec et al. [7] proposed the asymptotically space-optimal algorithm for uni-directional ring networks. Barriere et al. [6] considered the relationship between the gathering problem and the leader agent election problem. They showed that the gathering problem and the leader agent election problem are solvable under only the assumption that the ring has sense of direction and the numbers of nodes and agents are relatively prime.

A fault tolerant gathering problem is considered in [9,10]. Flocchini et al. [9] considered the gathering problem when tokens fail and showed that knowledge of n (number of agents) allows better time complexity than knowledge of k (number of agents). Dobrev et al. [10] considered the gathering problem for the case that there exists a dangerous node, called a black hole. A black hole destroys any agent that visits there. They showed that it is impossible for all agents to gather and they considered how many agents can survive and gather.

A randomized algorithm to solve the gathering problem is shown in [11]. Kawai et al. considered the gathering problem for multiple agents under the assumption that agents know neither k nor n, and proposed a randomized algorithm to solve the gathering problem with high probability in $O(kn)$ total moves.

1.3. Organization

The paper is organized as follows. Section 2 presents the system models and the problem to be solved. In Section 3 we consider the first model, that is, the algorithm is deterministic and each agent has a distinct ID. In Section 4 we consider the second model, that is, the algorithm is randomized and agents are anonymous. In Section 5 we consider the third model, that is, the algorithm is deterministic and agents are anonymous. Section 6 concludes the paper. Note that in this paper, we explain algorithms and proofs at higher level. The readers can see the details in the technical report [17].

2. Preliminaries

2.1. Network model

A *unidirectional ring network* R is a tuple $R = (V, L)$, where V is a set of nodes and L is a set of unidirectional communication links. We denote by n $(= |V|)$ the number of nodes. Then, ring R is defined as follows.

- $V = \{v_0, v_1, \dots, v_{n-1}\}$
- $L = \{(v_i, v_{(i+1) \bmod n}) \mid 0 \le i \le n-1\}$

We define the direction from v_i to v_{i+1} as the *forward* direction, and the direction from v_{i+1} to v_i as the *backward* direction. In addition, we define the i-th $(i \neq 0)$ forward (resp., backward) agent a_h^i of agent a_h as the agent that exists in the a_h's forward (resp., backward) direction and there are $i-1$ agents between a_h and $a_{h'}$. Moreover, we call the a_h's 1-st forward and backward agents *neighboring agents* of a_h respectively.

In this paper, we assume nodes are anonymous, i.e., they do not have IDs. Every node $v_i \in V$ has a whiteboard that agents on node v_i can read from and write on the whiteboard of v_i. We define W as a set of all states (contents) of a whiteboard.

2.2. Agent model

Let $A = \{a_1, a_2, \dots, a_k\}$ be a set of agents. We consider three model variants.

In the first model, we consider agents that are distinct (i.e., agents have distinct IDs) and execute a deterministic algorithm. We model an agent a_h as a finite automaton $(S, \delta, s_{initial}, s_{final})$. The first element S is the set of the a_h's all states, which includes initial state $s_{initial}$ and final state s_{final}. When a_h changes its state to s_{final}, it terminates the algorithm. The second element δ is the state transition function. Since we treat deterministic algorithms, δ is a mapping $S \times W \to S \times W \times M$, where $M = \{1, 0\}$ represents whether the agent makes a movement or not in the step. The value 1 represents movement to the next node and 0 represents stay at the current node. Since rings are unidirectional, each agent moves only to its forward node. Note that if the state of a_h is s_{final} and the state of its current node's whiteboard is w_i, then $\delta(s_{final}, w_i) = (s_{final}, w_i, 0)$ holds. In addition, we assume that each agent cannot detect whether other agents exist at the current node or not. Moreover, we assume that each agent knows neither the number of nodes n nor agents k. Notice that $S, \delta, s_{initial}$, and s_{final} can be dependent on the agent's ID.

In the second model, we consider agents that are anonymous (i.e., agents have no IDs) and execute a randomized algorithm. We model an agent similarly to the first model except for state transition function δ. Since we treat randomized algorithms, δ is a mapping $S \times W \times R \to S \times W \times M$, where R represents a set of random values. Note that if the state of some agent is s_{final} and the state of its current node's whiteboard is w_i, then $\delta(s_{final}, w_i, R) = (s_{final}, w_i, 0)$ holds. In addition, we assume that each agent cannot detect whether other agents exist at the current node or not, but we assume that each agent knows the number of agents k. Notice that all the agents are modeled by the same state machine since they are anonymous.

In the third model, we consider agents that are anonymous and execute a deterministic algorithm. We also model an agent similarly to the first model. We assume that each agent knows the number of agents k. Note that all the agents are modeled by the same state machine.

2.3. System configuration

In an agent system, (global) *configuration* c is defined as a product of states of agents, states of nodes (whiteboards' contents), and locations of agents. We define C as a set of all configurations. In initial configuration $c_0 \in C$, we assume that no pair of agents stay at the same node. We assume that each node v_j has boolean variable $v_j.initial$ at the whiteboard that indicates existence of agents in the initial configuration. If there exists an agent on node v_j in the initial configuration, the value of $v_j.initial$ is true. Otherwise, the value of $v_j.initial$ is false.

Let A_i be an arbitrary non-empty set of agents. When configuration c_i changes to c_{i+1} by the step of every agent in A_i, we denote the transition by $c_i \xrightarrow{A_i} c_{i+1}$. In one atomic step, each agent $a \in A_j$ executes the following series of events: 1) reads the contents of its current node's whiteboard, 2) executes local computation, 3) updates the contents of the node's whiteboard, and 4) moves to the next node or stays at the current node. We assume that agents move instantaneously, that is, agents always exist at nodes (do not exist at links). If multiple agents at the same node are included in A_i, the agents take steps in an arbitrary order. When $A_i = A$ holds for every i, all agents take steps every time. This model is called the *synchronous model*. Otherwise, the model is called the *asynchronous model*. In this paper, we consider the asynchronous system.

If sequence of configurations $E = c_0, c_1, \ldots$ satisfies $c_i \xrightarrow{A_i} c_{i+1}$ $(i \geq 0)$, E is called an *execution* starting from c_0 by schedule $A_1, A_2, \ldots$. We consider only fair schedules, where every agent appears infinitely often. Execution E is infinite, or ends in final configuration c_{final} where every agent's state is s_{final}.

2.4. Partial gathering problem

The requirement of the partial gathering problem is that, for a given integer g, each agent should move to a node and terminate so that at least g agents should meet at every node where an agent terminates. Formally, we define the g-partial gathering problem as follows.

Definition 1. Execution E solves the g-partial gathering problem when the following conditions hold:

- Execution E is finite (i.e., all agents terminate).
- In the final configuration, all agents are in the finial states, and for any node v_j where an agent terminates, there exist at least g agents on v_j.

For the g-partial gathering problem, we have the following lower bound on the total number of agent moves. This lemma holds in both deterministic and randomized algorithms.

Theorem 1. *The total number of agent moves required to solve the g-partial gathering problem is $\Omega(gn)$ if $g \geq 2$.*

Proof. We consider an initial configuration such that all agents are scattered evenly (i.e., all the agents have the same distances to their nearest agents). We assume $n = ck$ holds for some positive integer c. Let V' be the set of nodes where agents exist in the final configuration, and let $x = |V'|$. Since at least g agents meet at v_j for any $v_j \in V'$, we have $k \geq gx$.

For each $v_j \in V'$, we define A_j as the set of agents that meet at v_j and T_j as the total number of moves of agents in A_j. Then, among agents in A_j, the i-th smallest number of moves to get to v_j is at least $(i-1)n/k$. Hence, we have

$$\begin{aligned} T_j &\geq \sum_{i=1}^{|A_i|}(i-1)\cdot\frac{n}{k} \\ &\geq \sum_{i=1}^{g}(i-1)\cdot\frac{n}{k} + (|A_j|-g)\cdot\frac{gn}{k} \\ &= \frac{n}{k}\cdot\frac{g(g-1)}{2} + (|A_j|-g)\cdot\frac{gn}{k}. \end{aligned}$$

Therefore, the total number of moves is at least

$$\begin{aligned} T &= \sum_{v_j \in V'} T_j \\ &\geq x\cdot\frac{n}{k}\cdot\frac{g(g-1)}{2} + (k-gx)\cdot\frac{gn}{k} \\ &= gn - \frac{gnx}{2k}(g+1). \end{aligned}$$

Since $k \geq gx$ holds, we have

$$T \geq \frac{n}{2}(g-1).$$

Thus, the total number of moves is at least $\Omega(gn)$. □

3. The first model: a deterministic algorithm for distinct agents

In this section, we propose a deterministic algorithm to solve the g-partial gathering problem for distinct agents (i.e., agents have distinct IDs). The basic idea is that agents elect a leader and then the leader instructs other agents which nodes they meet at. However, since $\Omega(n \log k)$ total moves are required to elect one leader [5], this approach cannot lead to the g-partial gathering in asymptotically optimal total moves (i.e., $O(gn)$). To achieve the partial gathering in $O(gn)$ total moves, we elect multiple agents as leaders by executing the leader agent election partially. By this behavior, the number of moves for the election can be bounded by $O(n \log g)$. In addition, we show that the total number of moves for agents to move to their gathering nodes by leaders' instruction is $O(gn)$. Thus, our algorithm solves the g-partial gathering problem in $O(gn)$ total moves.

The algorithm consists of two parts. In the first part, multiple agents are elected as leader agents. In the second part, the leader agents instruct the other agents which nodes they meet at, and the other agents move to the nodes by the instruction.

3.1. The first part: leader election

The aim of the first part is to elect leaders that satisfy the following conditions called *leader election conditions*: 1) at least one agent is elected as a leader, and 2) there exist at least $g-1$ non-leader agents between two leader agents. To attain this goal, we use a traditional leader election algorithm [18]. However, the algorithm in [18] is executed by nodes and the goal is to elect exactly one leader. Hence we modify the algorithm to be executed by agents, and then agents elect multiple leader agents by executing the algorithm partially.

During the execution of leader election, the states of agents are divided into the following three types:

- *active*: The agent is performing the leader agent election as a candidate of leaders.
- *inactive*: The agent has dropped out from the candidate of leaders.
- *leader*: The agent has been elected as a leader.

First, we explain the idea of leader election by assuming that the ring is bidirectional. The algorithm consists of several phases. In each phase, each active agent compares its own ID with IDs of its backward and forward neighboring active agents. More concretely, each active agent a_h writes its own ID id_2 on the whiteboard of its current node, and moves backward and forward. Then, a_h observes ID id_1 of its backward active agent and id_3 of its forward active agent. After this, a_h decides if it remains active or drops out from the candidates of leaders. Concretely, if its own ID id_2 is the smallest among the three IDs, a_h remains active (as a candidate of leaders) in the next phase. Otherwise, a_h drops out from the candidate of leaders and becomes inactive. Note that, in each phase, neighboring active agents never remain as candidates of leaders. Thus, at least half active agents become inactive in each phase. Moreover from [18], after executing j phases, there exist at least $2^j - 1$ inactive agents between two active agents. Thus, after executing $\lceil \log g \rceil$ phases, the following properties are satisfied: 1) at least one agent remains as a candidate of leaders, and 2) the number of inactive agents between two active agents is at least $g-1$. Therefore, all remaining active agents become leaders since they satisfy the leader election conditions. Note that, before executing $\lceil \log g \rceil$ phases, the number of active agents may become one. In this case, the active agent immediately becomes a leader.

Now, we implement the above algorithm in unidirectional rings by applying a traditional technique [18]. Let us consider the behavior of active agent a_h. In unidirectional rings, a_h cannot move backward and cannot observe the ID of its backward active agent. Instead, a_h moves forward until it observes IDs of two active agents. Then, a_h observes IDs of three successive active agents. We assume a_h observes id_1, id_2, id_3 in this order. Note that id_1 is the ID of a_h. Here this situation is similar to that the active agent with ID id_2 observes id_1 as its backward active agent and id_3 as its forward active agent in bidirectional rings. For this reason, a_h behaves as if it would be an active agent with ID id_2 in bidirectional rings. That is, if id_2 is the smallest among the three IDs, a_h remains active as a candidate of leaders. Otherwise, a_h drops out from the candidate of leaders and becomes inactive. After the phase if a_h remains active as a candidate, it assigns id_2 to its ID and starts the next phase.[1] Agents repeat these behaviors until they complete the $\lceil \log g \rceil$-th phase.

For example, consider the initial configuration in Fig. 1 (a). In the figures, the number near each agent is the ID of the agent and the box of each node represents the whiteboard. First, each agent writes its own ID on the whiteboard of its initial node. Next, each agent moves forward until it observes two IDs, and then the configuration is changed to the one in Fig. 1 (b). In this configuration, each agent compares three IDs. The agent with ID 1 observes IDs (1, 8, 3), and hence it drops out from the candidate because the middle ID 8 is not the smallest. The agents with IDs 3, 2, and 5 also drop out from the candidates. The agent with ID 7 observes IDs (7, 1, 8), and hence it remains active as a candidate because the middle ID 1 is the smallest. Then, it updates its ID to 1 and starts the next phase. The agents with IDs 8, 4, and 6 also remain active as candidates and similarly update their IDs and start the next phase.

[1] We imitate the way in [18], but active agent a_h may still use its own ID id_1 in the next phase.

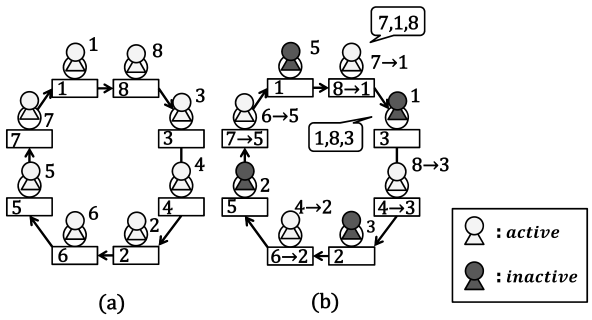

Fig. 1. An execution example of the leader election part ($k = 8$, $g = 3$).

Note that during the execution of the leader election, it is possible that a_h becomes the only one candidate of leaders. In this case, a_h immediately becomes a leader. In addition since agents move asynchronously, it may happen that some agent a_h passes another agent a_i. In this case, a_h needs to waits for a_i at some node, and we explain the detail in the full version. We can prove the following lemmas similarly to [18].

Lemma 1. *The leader election algorithm eventually terminates, and the configuration satisfies the following properties.*

- *There exists at least one leader agent.*
- *There exist at least $g - 1$ inactive agents between two leader agents.*

Proof. We omit the proof that the algorithm eventually terminates. First, we show that there exists at least one leader agent. In each phase if id_2 is the smallest of the three IDs, a_h remains active. Otherwise, a_h becomes inactive. Since each agent uses a unique ID, if there exist at least two active agents in some phase i, at least one agent remains active after executing the phase i. In addition, if there exists exactly one candidate of leaders and the other agents remain inactive, the candidate becomes a leader. Therefore, there exists at least one leader agent.

Next, we show that there exist at least $g - 1$ inactive agents between two leader agents. At first, we show that after executing j phases, there exist at least $2^j - 1$ inactive agents between two active agents. We show it by induction on the phase and by using the fact that in each phase if agent a_h remains as a candidate of leaders, then its backward and forward active agents drop out from candidates of leaders. For the case $j = 1$, there exists at least $1 = 2^1 - 1$ inactive agents between two active agents. For the case $j = l$, we assume that there exist at least $2^l - 1$ inactive agents between two active agents. Then, after executing $l + 1$ phases, since at least one of neighboring active agents becomes inactive, the number of inactive agents between two active agents is at least $(2^l - 1) + 1 + (2^l - 1) = 2^{l+1} - 1$. Hence, we can show that after executing j phases, there exist at least $2^j - 1$ inactive agents between two active agents. Therefore, after executing $\lceil \log g \rceil$ phases, there exist at least $g - 1$ inactive agents between two leader agents. □

Lemma 2. *The total number of agent moves to elect multiple leader agents is $O(n \log g)$.*

Proof. In each phase, each active agent moves until it observes two IDs of active agents. This costs $O(n)$ moves in total because each communication link is passed by two agents. Since agents execute $\lceil \log g \rceil$ phases, we have the lemma. □

3.2. The second part: movement to gathering nodes

The second part achieves the g-partial gathering by using leaders elected in the first part. Let leader nodes (resp., inactive nodes) be the nodes where agents become leaders (resp., inactive agents) in the first part. In this part, states of agents are divided into the following three types:

- *leader*: The agent instructs inactive agents where they should move.
- *inactive*: The agent waits for the leader's instruction.
- *moving*: The agent moves to its gathering node.

The idea of the algorithm is to divide agents into groups each of which consists of at least g agents. Concretely, first each leader agent a_h writes 0 on the whiteboard of the current node (i.e., the leader node). Next, a_h moves to the next leader node, that is, the node where 0 is already written on the whiteboard. While moving, whenever a_h visits an inactive node v_j, it counts the number of inactive nodes that a_h has visited. If the number plus one is not a multiple of g, a_h writes 0 on the whiteboard. Otherwise, a_h writes 1 on the whiteboard. These numbers are used to indicate whether the node is a gathering node or not. The number 0 means that agents do not meet at the node and the number 1 means that at least g

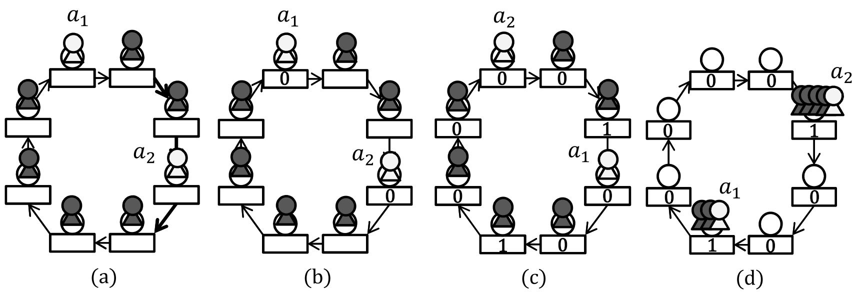

Fig. 2. The realization of partial gathering ($g = 3$).

agents meet at the node. When a_h reaches the next leader node, it changes its own state to a moving state, and we explain the behavior of moving agents later. For example, consider the configuration in Fig. 2 (a). In this configuration, agents a_1 and a_2 are leader agents. First, a_1 and a_2 write 0 on their current whiteboards (Fig. 2 (b)), and then they move and write numbers on whiteboards until they visit the node where 0 is already written on the whiteboard. Then, the system reaches the configuration in Fig. 2 (c).

Each non-leader (i.e., inactive agent) a_h waits at the current node until the value of the whiteboard is updated. When the value is updated, a_h changes its own state to a moving state. Each moving agent moves to the nearest node where 1 is written on the whiteboard. For example, after the configuration in Fig. 2 (c), each non-leader agent moves to the node where 1 is written on the whiteboard and the system reaches the configuration in Fig. 2 (d). After that, agents can solve the g-partial gathering problem.

We have the following lemma about the algorithm in Section 3.2.

Lemma 3. *After the leader agent election, agents solve the g-partial gathering problem in $O(gn)$ total moves.*

Proof. At first, we show the correctness of the proposed algorithm. Let $v_0^g, v_1^g, \dots, v_l^g$ be nodes written 1 on their whiteboards after all leader agents complete their behaviors, and we call these nodes *gathering nodes*. From the algorithm, each moving agent moves to the nearest gathering node v_j^g $(0 \le j \le l)$. By Lemma 1, there exist at least $g - 1$ moving agents between v_j^g and v_{j+1}^g Hence, agents can solve the g-partial gathering problem. In the following, we consider the total number of moves required to execute the algorithm.

First, the total number of moves required for each leader agent to move to its next leader node is obviously n. Next, let us consider the total number of moves required for each moving agent to move to nearest gathering node v_j^g (for example, the total moves from Fig. 2 (c) to Fig. 2 (d)). Remind that there are at least $g - 1$ inactive agents between two leader agents and each leader agent a_h writes 1 after writing 0 $g - 1$ times. Hence, there are at most $2g - 1$ moving agents between v_j^g and v_{j+1}^g. Thus, the total number of these moves is $O(gn)$ because each link is passed by at most $2g$ agents. Therefore, we have the lemma. □

From Lemmas 2 and 3, we have the following theorem.

Theorem 2. *When agents have distinct IDs, our deterministic algorithm solves the g-partial gathering problem in $O(gn)$ total moves.* □

4. The second model: a randomized algorithm for anonymous agents

In this section, we propose a randomized algorithm to solve the g-partial gathering problem for anonymous agents under the assumption that each agent knows the total number k of agents. The idea of the algorithm is the same as that in Section 3. In the first part, agents execute the leader election partially and elect multiple leader agents. In the second part, the leader agents determine gathering nodes and all agents move to the nearest gathering nodes. In the previous section each agent uses distinct IDs to elect multiple leader agents, but in this section each agent is anonymous and uses random IDs. We also show that the g-partial gathering problem is solved in $O(gn)$ expected total moves.

4.1. The first part: leader election

In this subsection, we explain a randomized algorithm to elect multiple leaders by using random IDs. Similarly to Section 3.1, the aim in this part is to satisfy the following conditions (leader election conditions): 1) at least one agent is elected as a leader, and 2) there exist at least $g - 1$ non-leader agents between two leader agents. The basic idea is the same as Section 3.1, that is, each active agent moves in the ring and compares three random IDs. If the ID in the middle

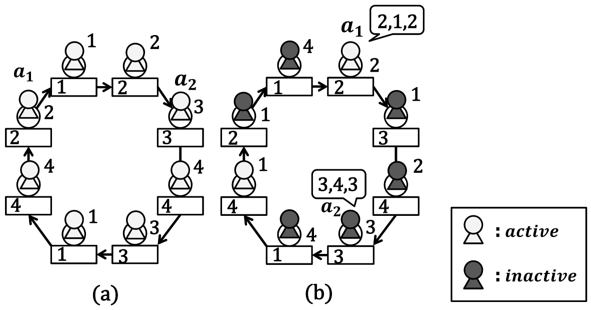

Fig. 3. An example that some agent observes the same random IDs.

is the smallest of the three random IDs, the active agent remains active. Otherwise, the active agent drops out from the candidate of leaders.

Now we explain details of the algorithm. In the beginning of each phase, each active agent selects $3\log k$ random bits as its own ID. After this, each agent executes in the same way as Section 3.1, that is, each active agent moves until it observes two random IDs of active agents and compares three random IDs. If the observed three IDs are distinct, the agent can execute the leader agent election similarly to Section 3.1. In addition to the behavior of the leader election in Section 3.1, when an agent becomes a leader at node v_j, the agent sets a *leader-flag* at v_j, and we explain how leader-flags are used later. If no agent observes a same random ID, the total number of moves for the leader agent election is the same as in Section 3.1, that is, $O(n\log g)$. In the following, we consider the case that some agent observes a same random ID.

Let $a_h.id_1, a_h.id_2$, and $a_h.id_3$ be random IDs that an active agent a_h observes in some phase. If $a_h.id_1 = a_h.id_3 \neq a_h.id_2$ holds, then a_h behaves similarly to Section 3.1, that is, if $a_h.id_2 < a_h.id_1 = a_h.id_3$ holds, a_h remains active and a_h becomes inactive otherwise. For example, let us consider a configuration of Fig. 3 (a). Each active agent moves until it observes two random IDs (Fig. 3 (b)). Then, agent a_1 observes three random IDs $(2,1,2)$ and remains active because $a_1.id_2 < a_1.id_1 = a_1.id_3$ holds. On the other hand, agent a_2 observes three random IDs $(3,4,3)$ and becomes inactive because $a_2.id_2 > a_2.id_1 = a_2.id_3$ holds. The other agents do not observe same random IDs and behave similarly to Section 3.1, that is, if their middle IDs are the smallest, they remain active and execute the next phase. If their middle IDs are not the smallest, they become inactive.

Next, we consider the case that either $a_h.id_2 = a_h.id_1$ or $a_h.id_2 = a_h.id_3$ holds. In this case, a_h changes its own state to a *semi-leader* state. A semi-leader is an agent that has a possibility to become a leader if there exists no leader agent in the ring. When at least one agent becomes a semi-leader, each active agent becomes inactive. The outline of the behavior of each semi-leader agent is as follows: First each semi-leader travels once around the ring. After this, if there already exists a leader agent in the ring, each semi-leader becomes inactive. Otherwise, the leader election is executed among all semi-leader agents, and exactly one semi-leader is elected as a leader and the other agents become inactive (including active agents). Note that, we can show that the probability some active agent becomes a semi-leader is sufficiently low and the expected number of semi-leader agents during the leader election is also sufficiently small. Hence even when each semi-leader travels once around the ring several times, the expected total moves to complete the leader agent election can be bounded by $O(n\log g)$.

Now, we explain the detailed behavior for semi-leader agents. When an active agent a_h becomes a semi-leader, it sets a *semi-leader-flag* on its current whiteboard. In the following, the node where the semi-leader flag is set (resp., not set) is called *a semi-leader node* (resp., *a non-semi-leader node*). After that, semi-leader agent a_h travels once around the ring. In the travel, when a_h visits a non-semi-leader node v_j where there exists an agent in the initial configuration, that is, a non-semi-leader node v_j such that $v_j.initial = true$ holds, a_h sets the *tour-flag* at v_j. This flag is used so that other agents notice the existence of a semi-leader and become inactive. Moreover when a_h visits a semi-leader node, a_h compares its random ID with the random ID written on the current whiteboard. Then, a_h memorizes whether its random ID is smaller or not and whether another semi-leader has the same random ID as its random ID or not.

After traveling once around the ring, a_h decides if it becomes a leader or inactive. While traveling in the ring, if a_h observes a leader-flag, it learns that there already exists a leader agent in the ring. In this case, a_h becomes inactive. Otherwise, a_h decides if it becomes a leader or inactive depending on random IDs. Let $a_h.id$ be a_h's random ID and A_{min} be the set of semi-leaders such that each semi-leader $a_h \in A_{min}$ has the smallest random ID id_{min} among all semi-leaders. In this case, each semi-leader $a_h \notin A_{min}$ clears a semi-leaders-flag and becomes inactive. On the other hand, if a_h has the unique minimum random ID (i.e., $|A_{min}| = 1$), a_h becomes a leader. Otherwise, a_h selects a random ID again, writes the ID on the current whiteboard, travels once around the ring. Then, a_h obtains new random IDs of semi-leaders. Each semi-leader a_h repeats such a behavior until $|A_{min}| = 1$ holds.

We have the following lemmas similarly to Section 3.1.

Lemma 4. *The leader agent election algorithm eventually terminates, and the condition satisfies the following properties.*

- *There exists at least one leader agent.*
- *There exist at least $g-1$ inactive agents between two leader agents.*

Proof. The above properties are the same as Lemma 1. Thus, if no agent becomes a semi-leader during the algorithm, each agent behaves similarly to Section 3.1 and the above properties are satisfied. Moreover if at least one agent becomes a semi-leader, exactly one semi-leader is elected as a leader and the other agents become inactive. Then, the above properties are clearly satisfied. Therefore, we have the lemma. □

Lemma 5. *The expected total number of agent moves to elect multiple leader agents is $O(n\log g)$.*

Proof. If there exist no neighboring active agents having the same random IDs, the algorithm works similarly to Section 3.1, and the total number of moves is $O(n\log g)$. In the following, we consider the case that some neighboring active agents have the same random IDs.

Let l be the length of a random ID. Then, the probability that two active neighboring active agents have the same random ID is $(\frac{1}{2})^l$. Thus, when there exist k_i active agents in the i-th phase, the probability that there exist neighboring active agents having the same random IDs is at most $k_i \times (\frac{1}{2})^l$. Since at least half active agents drop out from candidates in each phase, the probability that neighboring active agents have the same random IDs until the end of the $\lceil\log g\rceil$ phases is at most $k\times(\frac{1}{2})^l + \frac{k}{2}\times(\frac{1}{2})^l + \cdots + \frac{k}{2^{\lceil\log g\rceil-1}}\times(\frac{1}{2})^l < 2k\times(\frac{1}{2})^l$. Since $l = 3\log k$ holds, the probability is at most $\frac{2}{k^2} < \frac{1}{k}$. We assume that k active agents become semi-leaders and circulate around the ring because this case requires the most total moves. Then, each semi-leader a_h compares its random ID with random IDs of each semi-leader. Let A_{min} be the set of semi-leader agents whose random IDs are the smallest. If $|A_{min}| = 1$ holds, agents finish the leader agent election and the total number of moves is at most $O(kn)$. Otherwise, at least two semi-leaders have the same smallest random IDs. This probability is at most $k\times(\frac{1}{2})^l$. In this case, each semi-leader a_h updates its phase and random ID again, travels once around the ring, and obtains new random IDs of each semi-leader. Each semi-leader a_h repeats such a behavior until $|A_{min}| = 1$ holds. We assume that $t = k\times(\frac{1}{2})^l$ and semi-leaders complete the leader agent election after they circulate around the ring s times. In this case, before they circulate around the ring $s-1$ times, $|A_{min}| \neq 1$ holds every time they circulate around the ring. In addition when they circulate around the ring s times, $|A_{min}| = 1$ holds, and the probability such that $|A_{min}| = 1$ holds is clearly less than 1. Hence, the probability such that agents complete the leader election after they circulate around the ring s times is at most $t^{s-1}\times 1 = t^{s-1}$, and the total number of moves is at most skn. Since the probability that at least one agent becomes a semi-leader is at most $\frac{1}{k}$, the expected total number of moves for the case that some agents become semi-leaders and complete the leader agent election is at most $O(n\log g) + \frac{1}{k}\times\sum_{s=1}^{\infty} t^{s-1}\times skn = n\sum_{s=1}^{\infty} st^{s-1}$. Let S_n be $1\times 1 + 2\times t + \cdots + nt^{n-1}$. Then, we have $S_n = (nt^{n+1} - (n-1)t^n + 1)/(1-t)^2$. When $n = \infty$, we have $S_n = 1/(1-t)^2$. Moreover since $t = k\times(\frac{1}{2})^l$ and $l = 3\log k$ hold, we have $t < \frac{1}{2}$ and $S_n < 4$. Furthermore, the expected total number of moves is at most $O(n)$. Since the total moves to elect multiple leaders for the case that no agent becomes a semi-leader is $O(n\log g)$, the expected total moves for the leader election is $O(n\log g)$.

Therefore, we have the lemma. □

4.2. The second part: movement to gathering nodes

After executing the leader agent election in Section 4.1, the conditions shown by Lemma 4 is satisfied, that is, 1) at least one agent is elected as a leader, and 2) there exist at least $g-1$ inactive agents between two leader agents. Thus, we can execute the algorithms in Section 3.2 after the algorithms in Section 4.1. Therefore, agents can solve the g-partial gathering problem.

From Lemmas 3, 4, and 5, we have the following theorem.

Theorem 3. *When agents have no IDs, our randomized algorithm solves the g-partial gathering problem in expected $O(gn)$ total moves.* □

5. The third model: a deterministic algorithm for anonymous agents

In this section, we consider a deterministic algorithm to solve the g-partial gathering problem for anonymous agents. At first, we show that there exist unsolvable initial configurations in this model. Later, we propose a deterministic algorithm that solves the g-partial gathering problem in $O(kn)$ total moves for any solvable initial configuration.

5.1. Existence of unsolvable initial configurations

To explain unsolvable initial configurations, we define the *distance sequence* of a configuration. For configuration c, we define the distance sequence of agent a_h as $D_h(c) = (d_0^h(c), \ldots, d_{k-1}^h(c))$, where $d_i^h(c)$ is the distance between the i-th forward agent of a_h and the $(i+1)$-th forward agent of a_h in c. Then, we define the distance sequence of configuration c as the lexicographically minimum sequence among $\{D_h(c)|a_h \in A\}$, and we denote it by $D(c)$. In addition, we define several functions and variables for sequence $D = (d_0, d_1, \ldots, d_{k-1})$. Let $\mathit{shift}(D, x) = (d_x, d_{x+1}, \ldots, d_{k-1}, d_0, d_1, \ldots, d_{x-1})$ and when $D = \mathit{shift}(D, x)$ holds for some x such that $0 < x < k$ holds, we say D is *periodic*. Moreover, we define *period* of D as the minimum (positive) value such that $\mathit{shift}(D, \mathit{period}) = D$ holds.

Then, we have the following theorem.

Theorem 4. *Let c_0 be an initial configuration. If $D(c_0)$ is periodic and period is less than g, the g-partial gathering problem is not solvable.*

Proof. Let $m = k/\mathit{period}$. Let A_j $(0 \le j \le \mathit{period} - 1)$ be a set of agents a_h such that $D_h(c_0) = \mathit{shift}(D(c_0), j)$ holds. Then, when all agents move in the synchronous manner, all agents in A_j continue to do the same behavior and thus they cannot break the periodicity of the initial configuration. Since the number of agents in A_j is m and no two agents in A_j stay at the same node, there exist m nodes where agents stay in the final configuration. However, since $k/m = \mathit{period} < g$ holds, it is impossible that at least g agents meet at the m nodes. Therefore, the g-partial gathering problem is not solvable. □

5.2. Proposed algorithm

In this section, we propose a deterministic algorithm to solve the g-partial gathering problem in $O(kn)$ total moves for solvable initial configurations. Let $D = D(c_0)$ be the distance sequence of initial configuration c_0. From Theorem 4, the g-partial gathering problem is not solvable if $\mathit{period} < g$. On the other hand, our proposed algorithm solves the g-partial gathering problem if $\mathit{period} \ge g$ holds. In this section, we assume that each agent knows the number k of agents.

The idea of the algorithm is as follows: First each agent a_h travels once around the ring and obtains the distance sequence $D_h(c_0)$. After that, a_h computes D and *period*. If $\mathit{period} < g$ holds, a_h terminates the algorithm because the g-partial gathering problem is not solvable. Otherwise, agent a_h identifies nodes such that agents in $\{a_\ell | D = D_\ell(c_0)\}$ initially exist. Then, a_h moves to the nearest node among them. Clearly $\mathit{period}\,(\ge g)$ agents meet at the node, and the algorithm solves the g-partial gathering problem.

We have the following theorem.

Theorem 5. *When agents have no IDs, our deterministic algorithm solves the g-partial gathering problem in $O(kn)$ total moves if the initial configuration is solvable.*

Proof. At first, we show the correctness of the algorithm. Each agent a_h travels once around the ring, and computes the distance sequence D and *period*. If $\mathit{period} < g$ holds, the g-partial gathering problem is not solvable from Theorem 4 and a_h terminates the algorithm. In the following, we consider the case that $\mathit{period} \ge g$ holds. Each agent a_h moves to the nearest node such that the agent in $\{a_\ell | D = D_\ell(c_0)\}$ initially exists. Since $\mathit{period}\,(\ge g)$ agents move to the node, the algorithm solves the g-partial gathering problem.

Next, we analyze the total moves required to solve the g-partial gathering problem. From the algorithm, all agents circulate the ring, which requires $O(kn)$ total moves. After this, each agent moves at most n times to meet other agents, which requires $O(kn)$ total moves. Therefore, agents solve the g-partial gathering problem in $O(kn)$ total moves. □

6. Conclusion

In this paper, we have proposed three algorithms to solve the g-partial gathering problem in asynchronous unidirectional rings. The first algorithm is deterministic and works for distinct agents. The second algorithm is randomized and works for anonymous agents under the assumption that each agent knows the total number of agents. The third algorithm is deterministic and works for anonymous agents under the assumption that each agent knows the total number of agents. In the first and second algorithms, several agents are elected as leaders by executing the leader agent election partially. The first algorithm uses agents' distinct IDs and the second algorithm uses random IDs. In the both algorithms, after the leader election, leader agents instruct the other agents where they meet. On the other hand, in the third algorithm, each agent travel once around the ring and moves to a node and terminates so that at least g agents should meet at the same node. We have showed that the first and second algorithms requires $O(gn)$ total moves, which is asymptotically optimal. The future work is to analyze the lower bound under the assumption that the algorithm is deterministic and each agent is anonymous. We conjecture that it is $\Omega(kn)$, and if the conjecture is correct, we can show that the third algorithm is asymptotically optimal in terms of total moves.

References

[1] E. Kranakis, D. Krozanc, E. Markou, The mobile agent rendezvous problem in the ring, in: Synthesis Lectures on Distributed Computing Theory, vol. 1, 2010, pp. 1–122.
[2] E. Kranakis, N. Santoro, C. Sawchuk, D. Krizanc, Mobile agent rendezvous in a ring, in: Proc. of the 23rd International Conference on Distributed Computing Systems, 2003, pp. 592–599.

[3] P. Flocchini, E. Kranakis, D. Krizanc, N. Santoro, C. Sawchuk, Multiple mobile agent rendezvous in a ring, in: Proc. of the 6th Latin American Theoretical Informatics, in: LNCS, vol. 2976, 2004, pp. 599–608.
[4] S. Dobrev, P. Flocchini, G. Prencipe, N. Santoro, Mobile search for a black hole in an anonymous ring, Algorithmica 48 (1) (2007) 67–90.
[5] T. Suzuki, T. Izumi, F. Ooshita, H. Kakugawa, T. Masuzawa, Move-optimal gossiping among mobile agents, Theoret. Comput. Sci. 393 (1) (2008) 90–101.
[6] L. Barriere, P. Flocchini, P. Fraigniaud, N. Santoro, Rendezvous and election of mobile agents: impact of sense of direction, Theory Comput. Syst. 40 (2) (2007) 143–162.
[7] L. Gasieniec, E. Kranakis, D. Krizanc, X. Zhang, Optimal memory rendezvous of anonymous mobile agents in a unidirectional ring, in: Proc. of the 32nd International Conference on Current Trends in Theory and Practice of Computer Science, in: LNCS, vol. 3831, 2006, pp. 282–292.
[8] E. Kranakis, D. Krizanc, E. Markou, Mobile agent rendezvous in a synchronous torus, in: Proc. of the 8th Latin American Theoretical Informatics, in: LNCS, vol. 3887, 2006, pp. 653–664.
[9] P. Flocchini, E. Kranakis, D. Krizanc, F.L. Luccio, N. Santoro, C. Sawchuk, Mobile agents rendezvous when tokens fail, in: Proc. of the 11th International Colloquium on Structural Information and Communication Complexity, in: LNCS, vol. 3104, 2004, pp. 161–172.
[10] S. Dobrev, P. Flocchini, G. Prencipe, N. Santoro, Multiple agents rendezvous in a ring in spite of a black hole, in: Proc. of the 8th International Conference on Principles of Distributed Systems, in: LNCS, vol. 3144, 2004, pp. 34–46.
[11] S. Kawai, F. Ooshita, H. Kakugawa, T. Masuzawa, Randomized rendezvous of mobile agents in anonymous unidirectional ring networks, in: Proc. of the 19th International Colloquium on Structural Information and Communication Complexity, in: LNCS, vol. 7355, 2012, pp. 303–314.
[12] A. Collins, J. Czyzowicz, L. Gasieniec, A. Kosowski, R. Martin, Synchronous rendezvous for location-aware agents, in: Proc. of the 25th International Symposium on Distributed Computing, in: LNCS, vol. 6950, 2011, pp. 447–459.
[13] S. Elouasbi, A. Pelc, Time of anonymous rendezvous in trees: determinism vs. randomization, in: Proc. of the 19th International Colloquium on Structural Information and Communication Complexity, in: LNCS, vol. 7355, 2012, pp. 291–302.
[14] D. Baba, T. Izumi, F. Ooshita, H. Kakugawa, T. Masuzawa, Linear time and space gathering of anonymous mobile agents in asynchronous trees, Theoret. Comput. Sci. (2013) 118–126.
[15] J. Czyzowicz, A. Kosowski, A. Pelc, Time vs. space trade-offs for rendezvous in trees, in: Proc. of the 24th ACM Symposium on Parallelism in Algorithms and Architectures, 2012, pp. 1–10.
[16] S. Guilbault, A. Pelc, Asynchronous rendezvous of anonymous agents in arbitrary graphs, in: Proc. of the 32nd International Symposium on Distributed Computing, in: LNCS, vol. 7109, 2011, pp. 421–434.
[17] M. Shibata, S. Kawai, F. Ooshita, H. Kakugawa, T. Masuzawa, Partial gathering of mobile agents in asynchronous unidirectional rings, Technical report, http://arxiv.org/abs/1505.06596v1, 2015.
[18] G.L. Peterson, An $O(n \log n)$ unidirectional algorithm for the circular extrema problem, ACM Trans. Program. Lang. Syst. 4 (4) (1982) 758–762.