

Partial gathering of mobile agents in dynamic rings

Masahiro Shibata[†] (✉), Yuichi Sudo[◇], Junya Nakamura[‡], and Yonghwan Kim[#]

[†] Kyushu Institute of Technology, Fukuoka, Japan

Email: shibata@cse.kyutech.ac.jp

[◇] Hosei University, Tokyo, Japan

Email: sudo@hosei.ac.jp

[‡] Toyohashi University of Technology, Aichi, Japan

Email: junya@imc.tut.ac.jp

[#] Nagoya Institute of Technology, Aichi, Japan

Email: kim@nitech.ac.jp

Abstract. In this paper, we consider the partial gathering problem of mobile agents in synchronous dynamic bidirectional rings. The partial gathering problem is a generalization of the (well-investigated) total gathering problem, which requires that all k agents distributed in the network terminate at a non-predetermined single node. The partial gathering problem requires, for a given positive integer $g (< k)$, that agents terminate in a configuration such that either at least g agents or no agent exists at each node. The requirement for the partial gathering problem is strictly weaker than that for the total gathering problem, and thus it is interesting to clarify the difference in the move complexity between them. So far, partial gathering has been considered in static graphs. In this paper, we consider this problem in 1-interval connected rings, that is, one of the links in the ring may be missing at each time step. In such networks, we aim to clarify the solvability of the partial gathering problem and the move complexity, focusing on the relationship between values of k and g . First, we consider the case of $3g \leq k \leq 8g - 2$. In this case, we show that our algorithm can solve the problem with the total number of $O(kn)$ moves, where n is the number of nodes. Since $k = O(g)$ holds when $3g \leq k \leq 8g - 2$, the move complexity $O(kn)$ in this case can be represented also as $O(gn)$. Next, we consider the case of $k \geq 8g - 3$. In this case, we show that our algorithm can also solve the problem and its move complexity is $O(gn)$. These results mean that, when $k \geq 3g$, the partial gathering problem can be solved also in dynamic rings. In addition, agents require a total number of $\Omega(gn)$ (resp., $\Omega(kn)$) moves to solve the partial (resp., total) gathering problem. Thus, the both proposed algorithms can solve the partial gathering problem with the asymptotically optimal total number of $O(gn)$ moves, which is strictly smaller than that for the total gathering problem.

Keywords: mobile agent, partial gathering problem, dynamic ring

1 Introduction

1.1 Background and Related Work

A *distributed system* comprises a set of computing entities (*nodes*) connected by communication links. As a promising design paradigm of distributed systems, (mobile) agents have attracted much attention [6]. The agents can traverse the system, carrying information collected at visited nodes, and execute an action at each node using the information to achieve a task. In other words, agents can encapsulate the process code and data, which simplifies design of distributed systems [10].

The *total gathering problem* (or the rendezvous problem) is a fundamental problem for agents' coordination. When a set of k agents are arbitrarily placed at nodes, this problem requires that all the k agents terminate at a non-predetermined single node. By meeting at a single node, all agents can share information or synchronize their behaviors. The total gathering problem has been considered in various kinds of networks such as rings [8, 9, 4], trees [1], tori [7], and arbitrary networks [3].

Recently, a variant of the total gathering problem, called the *g -partial gathering problem* [13], has been considered. This problem does not require all agents to meet at a single node, but allows agents to meet at several nodes separately. Concretely, for a given positive integer g ($< k$), this problem requires that agents terminate in a configuration such that either at least g agents or no agent exists at each node. From a practical point of view, the g -partial gathering problem is still useful especially in large-scale networks. That is, when g -partial gathering is achieved, agents are partitioned into groups each of which has at least g agents, each agent can share information and tasks with agents in the same group, and each group can partition the network and then patrol its area that it should monitor efficiently. The g -partial gathering problem is interesting also from a theoretical point of view. Clearly, if $k < 2g$ holds, the g -partial gathering problem is equivalent to the total gathering problem. On the other hand, if $k \geq 2g$ holds, the requirement for the g -partial gathering problem is strictly weaker than that for the total gathering problem. Thus, there exists possibility that the g -partial gathering problem can be solved with strictly smaller total number of moves (i.e., lower costs) compared to the total gathering problem.

As related work, in case of $k \geq 2g$, Shibata et al. considered the g -partial gathering problem in rings [13, 14, 18], trees [16], and arbitrary networks [15]. In [13, 14], they considered it in unidirectional ring networks with whiteboards (or memory spaces that agents can read and write) at nodes. They mainly showed that, if agents have distinct IDs and the algorithm is deterministic, or if agents do not have distinct IDs and the algorithm is randomized, agents can achieve g -partial gathering with the total number of $O(gn)$ moves (in expectation), where n is the number of nodes. Notice that in the above results agents do not have any global knowledge such as n or k . In [18], they considered g -partial gathering for another mobile entity called *mobile robots* that have no memory but can observe all nodes and robots in the network. In case of using mobile robots,

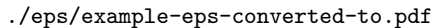


Fig. 1. An example of the g -partial gathering problem in a dynamic ring ($g = 3$).

they also showed that g -partial gathering can be achieved with the total number of $O(gn)$ moves. In addition, the g -partial (resp., the total) gathering problem in ring networks requires a total number of $\Omega(gn)$ (resp., $\Omega(kn)$) moves. Thus, the above results are asymptotically optimal in terms of the total number of moves, and the total number $O(gn)$ of moves is strictly smaller than that for the total gathering problem when $g = o(k)$. In tree and arbitrary networks, they also proposed algorithms to solve the g -partial gathering problem with strictly smaller total number of moves compared to the total gathering problem for some settings, but we omit the details in this paper.

Although all the above work on the total gathering problem and the g -partial gathering problem are considered in *static graphs* where a network topology does not change during an execution, recently many problems involving agents have been studied in *dynamic graphs*, where a topology changes during an execution. For example, the total gathering problem [12], the exploration problem [11, 5], the compact configuration problem [2] and the uniform deployment problem [17] are considered in dynamic graphs. However, to the best of our knowledge, there is no work for g -partial gathering in dynamic graphs, and hence in this paper we consider it in dynamic rings as a first step.

1.2 Our Contribution

In this paper, we consider the g -partial gathering problem of mobile agents in synchronous dynamic bidirectional rings with whiteboards at nodes. In this paper, we consider *1-interval connected rings* [12, 11, 2, 17], that is, one of the links may be missing at each time step. An example is given in Fig. 1. In such networks, we aim to clarify the solvability of the g -partial gathering problem and the move complexity, focusing on the relationship between values of k and g .

In this paper, we assume that agents have distinct IDs, chirality, and knowledge of n and k . In Table 1, we compare our contributions with the result for agents with distinct IDs in static rings. We also analyze the time complexity for solving the problem. First, we consider the case of $3g \leq k \leq 8g - 2$. In this case, we show that our algorithm can solve the problem with $O(n)$ time and the total number of $O(kn)$ moves. Next, we consider the case of $k \geq 8g - 3$. In this case,

Table 1. Results of g -partial gathering for agents with distinct IDs in ring networks (n : #nodes, k : #agents).

| | Result in [13] | Results of this paper | |
|------------------------------|----------------|-------------------------|-------------------|
| | | Result 1 (Sec. 3) | Result 2 (Sec. 4) |
| Static/Dynamic ring | Static | Dynamic | Dynamic |
| Knowledge of n and k | No | Available | Available |
| Relation between k and g | $k \geq 2g$ | $3g \leq k \leq 8g - 2$ | $k \geq 8g - 3$ |
| Time complexity | $\Theta(n)$ | $\Theta(n)$ | $\Theta(n)$ |
| Total number of agent moves | $\Theta(gn)$ | $O(kn)(= O(gn))$ | $\Theta(gn)$ |

we show that our algorithm can also solve the problem and the time complexity and the move complexity are $O(n)$ and $O(gn)$, respectively. These results mean that, although it is open that whether or not the g -partial gathering problem can be solved in dynamic rings when $2g \leq k < 3g$, it can be solved when $k \geq 3g$ and the time complexity $O(n)$ of our algorithms is asymptotically optimal. In addition, since $k = O(g)$ holds when $3g \leq k \leq 8g - 2$ holds like the first case, the both proposed algorithms can achieve g -partial gathering also with the asymptotically the total number of $O(gn)$ moves, which is strictly smaller than the move complexity for the total gathering problem. Furthermore, it is worthwhile to mention that, while total gathering (i.e., all agents gather at a single node) cannot be solved in dynamic rings and it needs to relax the requirement so that agents stay at either of two nodes connected by a link [12], g -partial gathering can be achieved without relaxing the requirement.

Due to the page limitation, we omit several pseudocodes and proofs of theorems and lemmas.

2 Preliminaries

2.1 System Model

We basically follow the model defined in [12]. A *dynamic bidirectional ring* R is defined as 2-tuple $R = (V, E)$, where $V = \{v_0, v_1, \dots, v_{n-1}\}$ is a set of n nodes and $E = \{e_0, e_1, \dots, e_{n-1}\}$ ($e_i = \{v_i, v_{(i+1) \bmod n}\}$) is a set of links. For simplicity, we denote $v_{(i+j) \bmod n}$ (resp., $e_{(i+j) \bmod n}$) by v_{i+j} (resp., $e_{(i+j)}$) for any integers i and j . We define the direction from v_i to v_{i+1} (resp., v_i to v_{i-1}) as the *forward* or *clockwise* (resp., *backward* or *counterclockwise*) direction. In addition, one of links in the ring may be missing at each time step, and which link is missing is controlled by an *adversarial scheduler*. Such a dynamic ring is known as a *1-interval connected ring*. The *distance* from node v_i to v_j is defined to be $(j - i) \bmod n$. Note that this definition of the distance is correct when any of the links from v_i to v_j is not missing. Moreover, we assume that nodes are anonymous, i.e., they do not have IDs. Every node $v_i \in V$ has a whiteboard that agents at node v_i can read from and write on.

Let $A = \{a_0, a_1, \dots, a_{k-1}\}$ be a set of $k (\leq n)$ agents. Agents can move through directed links, that is, they can move from v_i to v_{i+1} (i.e., move forward)

or from v_i to v_{i-1} (i.e., move backward) for any i . Agents have distinct IDs and knowledge of n and k ¹. Agents have *chirality*, that is, they agree on the orientation of clockwise and counterclockwise direction in the ring. In addition, agents cannot detect whether other agents exist at the current node or not. An agent a_i is defined as a deterministic finite automaton $(S, W, \delta, s_{initial}, s_{final}, w_{initial}, w'_{initial})$. The first element S is the set of all states of an agent, including two special states, initial state $s_{initial}$ and final state s_{final} . The second element W is the set of all states (contents) of a whiteboard, including two special initial states $w_{initial}$ and $w'_{initial}$. We explain $w_{initial}$ and $w'_{initial}$ in the next paragraph. The third element $\delta : S \times W \mapsto S \times W \times M$ is the state transition function that decides, from the current states of a_i and the current node's whiteboard, the next states of a_i and the whiteboard, and whether a_i moves to its neighboring node or not. The last element $M = \{-1, 0, 1\}$ in δ represents whether a_i makes a movement or not. The value 1 (resp., -1) means moving forward (resp., backward) and 0 means staying at the current node. We assume that $\delta(s_{final}, w_j) = (s_{final}, w_j, 0)$ holds for any state $w_j \in W$, which means that a_i never changes its state, updates the contents of a whiteboard, or leaves the current node once it reaches state s_{final} . We say that an agent *terminates* when its state changes to s_{final} . Notice that $S, \delta, s_{initial}$, and s_{final} can be dependent on the agent's ID.

In an agent system, (global) *configuration* c is defined as a product of the states of all agents, the states (whiteboards' contents) of all nodes, and the locations (i.e., the current nodes) of all agents. We define C as a set of all configurations. In an initial configuration $c_0 \in C$, we assume that agents are deployed arbitrarily at mutually distinct nodes, (or no two agents start at the same node), and the state of each whiteboard is $w_{initial}$ or $w'_{initial}$ depending on the existence of an agent. That is, when an agent exists at node v in the initial configuration, the initial state of v 's whiteboard is $w_{initial}$. Otherwise, the state is $w'_{initial}$.

During an execution of the algorithm, we assume that agents move instantaneously, that is, they always exist at nodes (do not exist on links). Each agent executes the following four operations in an *atomic action*: 1) reads the contents of its current node's whiteboard, 2) executes local computation (or changes its state), 3) updates the contents of the current node's whiteboard, and 4) moves to its neighboring node or stays at the current node. If several agents exist at the same node, they take atomic actions interleavingly in an arbitrary order. In addition, when an agent tries to move to its neighboring node (e.g., from node v_j to v_{j+1}) but the corresponding link (e.g., link e_j) is missing, we say that the agent is *blocked*, and it still exists at v_j at the beginning of the next atomic action.

In this paper, we consider a *synchronous execution*, that is, in each time step called *round*, all agents perform atomic actions. Then, an *execution* starting from c_0 is defined as $E = c_0, c_1, \dots$ where each c_i ($i \geq 1$) is the configuration reached

¹ The knowledge of k is used for agents to decide which proposed algorithm they apply by comparing it with the value of g .

from c_{i-1} by atomic actions of all agents. An execution is infinite, or ends in a *final configuration* where the state of every agent is s_{final} .

2.2 The Partial Gathering Problem

The requirement for the g -partial gathering problem is that, for a given integer g , agents terminate in a configuration such that either at least g agents or no agent exists at each node. Formally, we define the problem as follows.

Definition 1. *An algorithm solves the g -partial gathering problem in dynamic rings when the following conditions hold:*

- *Execution E is finite (i.e., all agents terminate in state s_{final}).*
- *In the final configuration, at least g agents exist at any node where an agent exists.*

In this paper, we evaluate the proposed algorithms by the time complexity (the number of rounds for agents to solve the problem) and the total number of agents moves. In [13], the lower bound on the total number of agent moves for static rings is shown to be $\Omega(gn)$. This theorem clearly holds also in dynamic rings.

Theorem 1. *A lower bound on the total number of agent moves required to solve the g -partial gathering problem in dynamic rings is $\Omega(gn)$ if $g \geq 2$.*

On the time complexity, the following theorem holds. Intuitively, this is because there exist an initial configuration and link-missings such that the distance between some agent a_i and its nearest agent is $\Omega(n)$, which requires $\Omega(n)$ rounds for a_i to meet with other agents.

Theorem 2. *A lower bound on the time complexity required to solve the g -partial gathering problem in dynamic rings is $\Omega(n)$.*

3 The case of $3g \leq k \leq 8g - 2$

In this section, when $3g \leq k \leq 8g - 2$, we propose a naive algorithm to solve the g -partial gathering problem in dynamic rings with $O(n)$ rounds and the total number of $O(kn)$ moves. Since $k = O(g)$ holds in this case, this algorithm is asymptotically in terms of both the time and move complexities, similar to the second algorithm explained in Section 4. In this algorithm, all agents try to travel once around the ring to get IDs of all agents, and then determine a single common node where all agents should gather. However, it is possible that some agent cannot travel once around the ring and get IDs of all agents due to missing links. Agents treat this by additional behaviors explained by the following subsections. The algorithm comprises two phases: the selection phase and the gathering phase. In the selection phase, agents move in the ring and determine the *gathering node* where they should gather. In the gathering phase, agents try to stay at the gathering node.

3.1 Selection phase

The aim of this phase is that each agent achieves either of the following two goals: (i) It travels once around the ring and gets IDs of all agents, or (ii) it detects that all agents stay at the same node. To this end, we use an idea similar to [12] which considers total gathering in dynamic rings. First, each agent a_i writes its ID on the current whiteboard, and then tries to move forward for $3n$ rounds. During the movement, a_i memorizes values of observed IDs to array $a_i.ids[]$. After the $3n$ rounds, the number $a_i.nVisited$ of nodes that a_i has visited is (a) at least n or (b) less than n due to missing links. In case (a), a_i must have completed traveling once around the ring. Thus, a_i can get IDs of all k agents (goal (i) is achieved). Then, a_i (and the other agents) select the gathering node v_{gather} as the node where the minimum ID min is written.

In case (b) (i.e., a_i has visited less than n nodes during the $3n$ rounds), we show in Lemma 1 that all k agents stay at the same node (goal (ii) is achieved). This situation means that agents already achieve g -partial (or total) gathering, and they terminate the algorithm execution.

Concerning the selection phase, we have the following lemma.

Lemma 1. *After finishing the selection phase, each agent achieves either of the following two goals: (i) It travels once around the ring and gets IDs of all agents, or (ii) it detects that all agents stay at the same node.*

3.2 Gathering phase

In this phase, agents aim to achieve g -partial (or total) gathering by trying to visit the gathering node v_{gather} . Concretely, for $3n$ rounds from the beginning of this phase, each agent a_i tries to move forward until it reaches v_{gather} . If agents are blocked few times, all agents can reach v_{gather} and they achieve g -partial (or total) gathering. However, it is possible that some agent cannot reach v_{gather} due to link-missings. To treat this, we introduce a technique called *splitting*. Intuitively, in this technique, when at least $2g$ agents exist at some node, from there an agent group with at least g agents tries to move forward and another agent group with at least g agents tries to move backward. In addition, when an agent group with at least g agents visits a node where less than g agents exist, the less than g agents join the agent group and try to move to the same direction as that of the group. By this behavior, it does not happen that all agents are blocked, and agents can eventually terminate in a configuration such that at least g agents exist at each node where an agent exists.

Concretely, after the $3n$ round from when agents tried to move forward to reach v_{gather} , by the similar discussion of Lemma 1, all agents that do not reach v_{gather} stay at the same node. Let v' be the node. Then, there are at most two nodes v_{gather} and v' where agents exist after the movement. If between g and $2g - 1$ agents exist at v_{gather} or v' (or both), the agents staying there terminate the algorithm execution. On the other hand, if less than g agents exist at v_{gather} (resp., v'), at least $2g$ agents exist at v' (resp., v_{gather}) since we consider the

eps/smallKgathering-eps-converted-to.pdf

Fig. 2. An execution example of the gathering phase ($g = 3$).

case of $k \geq 3g$. We call the node with at least $2g$ agents v_{more} . Notice that it is possible that at least $2g$ agents exist at both v_{gather} and v' . Let $k' (\geq 2g)$ be the number of agents staying at v_{more} . Then, each agent a_i at v_{more} calculates how small its ID is among the k' agents. We denote the ordinal number by $a_i.rank$. Then, if $1 \leq a_i.rank \leq g$ holds, it belongs to the *forward agent group* A_f and tries to move forward. Else if $(k' < 3g) \vee (g + 1 \leq a_i.rank \leq 2g)$ holds, it belongs to the *backward agent group* A_b and tries to move backward. If a_i does not satisfy any of the above conditions, it terminates the algorithm execution because there still exist at least g agents even after A_f and A_b leave v_{more} .

While A_f and A_b move in the ring, if A_f (resp., A_b) visits a new node v_j , it sets a flag $v_j.fMarked$ (resp., $v_j.bMarked$) representing that v_j is visited by A_f (resp., A_b). These flags are used for an agent group A to check whether or not the current node is visited by another agent group and A can stop moving in the ring. In addition, if A_f (resp., A_b) visits a node with less than g agents, the less than g agents join A_f (resp., A_b) and try to move forward (resp., backward). However, it is possible that the number num of agents in the updated group is more than $2g$. In this case, using their IDs, only g agents continue to try moving and the remaining $num - g$ agents terminate the algorithm execution at the current node. By this behavior, each link is passed by at most $2g$ agents and the total number of moves for agent groups can be reduced to $O(gn)$ (this technique is used in Section 4). Moreover, since A_f or A_b can visit a next node at each round even when some link is missing, A_f (resp., A_b) repeats such a behavior for n rounds or until it visits some node v_j with $v_j.bMarked = true$ (resp., $v_j.fMarked = true$), which implies that all the remaining nodes that A_f (resp., A_b) should visit are already visited by another agent group A_b (resp., A_f).

An example is given in Fig. 2 (we omit nodes unrelated to the example). From (a) to (b), a backward group A_b visits a node with two ($< g$) agents, and the two agents join A_b . Then, since the number of agents in the updated A_b is 7, ($> 2g$), only three agents continue to try moving and the remaining four

agents terminate the algorithm execution there ((b) to (c)). From (c) to (d), we assume that a forward agent group A_f continues to be blocked due to a missing link. Even in this case, A_b can continue to move since there is only one missing link at each round. When A_f (resp., A_b) visits a node with a flag set by A_b (resp., A_f) like (e), or n rounds passed from when agent groups started trying to move, agents achieve g -partial gathering.

Concerning the gathering phase, we have the following lemma.

Lemma 2. *After finishing the gathering phase, agents achieve g -partial gathering.*

We have the following theorem for the proposed algorithm.

Theorem 3. *When $3g \leq k \leq 8g - 2$ holds, the proposed algorithm solves the g -partial gathering problem in dynamic rings with $O(n)$ rounds and the total number of $O(kn)$ moves.*

4 The case of $k \geq 8g - 3$

In this section, when $k \geq 8g - 3$, we propose an algorithm to solve the problem with $O(n)$ rounds and the total number of $O(gn)$ (i.e., optimal) moves. Since the move complexity is not $O(kn)$ but $O(gn)$, it is not possible that all agents try to travel once around the ring as in Section 3. Hence, in this section agents aim to reduce the total number of moves using distinct IDs and the fact of $k \geq 8g - 3$. The algorithm comprises three phases: the semi-selection phase, the semi-gathering phase, and the achievement phase. In the semi-selection phase, agents select a set of *gathering-candidate nodes* each of where at least $2g$ agents may gather. In the semi-gathering phase, agents try to stay at a gathering-candidate node. As a result, at least $2g$ agents gather at some node (the node may not be a gathering-candidate node due to link-missings). In the achievement phase, agents achieve g -partial gathering by the same method as that for the gathering phase in Section 3.2.

4.1 Semi-selection phase

The aim of this part is to select a set of gathering-candidate nodes each of where at least $2g$ agents may gather. A possible approach is that each agent a_i moves forward and backward for getting IDs of its 1-st, 2-nd, \dots , $(2g - 1)$ -st forward agents and IDs of its 1-st, 2-nd, \dots , $(2g - 1)$ -st backward agents, and then returns to its initial node. Here, the i -th ($i \neq 0$) forward (resp., backward) agent a' of agent a represents the agent such that $i - 1$ agents exist between a and a' in a 's forward (resp., backward) direction in the initial configuration. Thereafter, a_i compares its ID and the obtained $4g - 2$ IDs. If its ID is the minimum, a_i selects its initial node as a gathering-candidate node v_{candi} . Then, the $2g - 1$ agents existing in a_i 's backward direction try to move forward to stay at v_{candi} and eventually $2g$ agents may gather at v_{gather} . However, since

we consider 1-interval connected rings, there are two problems: (1) it is possible that no gathering-candidate node is selected since some agent may not be able to collect $4g - 2$ IDs due to link-missings, and (2) even if a gathering-candidate node v_{candi} is selected, it is possible that some agent cannot reach v_{candi} due to link-missings and only less than $2g$ agents gather at each node.

To treat these problems, each agent a_i in this phase keeps trying to move forward, tries to observe more than $4g - 2$ IDs, and considers some observed ID as its own ID when it observed the necessary number of IDs. Concretely, for $3n$ rounds, each agent a_i tries to move forward until it observes $10g - 4$ IDs or at least $2g$ agents exist at the current node. Thereafter, a_i determines its behavior depending on whether it observed at least $8g - 3$ IDs or not. If a_i did not observe at least $8g - 3$ IDs, we show in Lemma 3 that at least $2g$ agents exist at some node v_j and then a flag $v_j.candi$ is set to true to represent that v_j is a gathering-candidate node (problem (1) is solved). Intuitively, this is because a_i does not observe at least $(10g - 4) - (8g - 4) = 2g$ IDs and this means that at least $2g - 1$ agents existing in a_i 's backward direction also do not observe the necessary number of IDs and they eventually stay at a_i 's node.

On the other hand, if a_i observed at least $8g - 3$ IDs, it uses the first $8g - 3$ IDs for comparison and considers the $(4g - 1)$ -st ID as its own ID. Then, this situation is similar to one that a_i compares its ID with $4g - 2$ forward IDs and $4g - 2$ backward IDs. Hence, if the $(4g - 1)$ -st ID is the minimum among the $8g - 3$ IDs, a_i sets $v_j.candi = true$ at the current node v_j . Then, since $k \geq 8g - 3$ holds, all the $8g - 3$ IDs are distinct and thus $4g - 2$ agents existing in a_i 's backward direction can recognize a_i 's staying node as the nearest gathering-candidate node v_{candi} in the forward direction when they observed at least $8g - 3$ IDs. Thus, the $4g - 1$ agents in total (a_i and the $4g - 2$ agents) try to move forward and stay at v_{candi} (the detail is explained in the next subsection). Then, when some link continues to be missing, the $4g - 1$ agents are partitioned into two groups and at least one group has $2g$ agents (problem (2) is solved).

The pseudocode of the semi-selection phase is described in Algorithm 1. Global variables used in the algorithm is summarized in Table 2 (several variables are used in other sections). Concerning the semi-selection phase, we have the following lemma.

Lemma 3. *After finishing the semi-selection phase, there exists at least one node v_j with $v_j.candi = true$.*

Proof. Let a_{min} be the agent with minimum ID among all agents and a_i be the $(4g - 2)$ -nd backward agent of a_{min} . We consider the cases that the value of $a_i.nIDs$ after executing Algorithm 1 is (a) less than $8g - 3$ and (b) at least $8g - 3$ in this order. First, (a) if $a_i.nIDs < 8g - 3$ holds, let $g' = (10g - 4) - a_i.nIDs$ be the number of IDs that a_i could not observe and $a_{i-1}, a_{i-2}, \dots, a_{i-(g'-1)}$ be the 1-st, 2-nd, \dots , $(g' - 1)$ -st backward agents of a_i . Then, since $(g' - 1) + a_i.nIDs = ((10g - 4) - a_i.nIDs) - 1 + a_i.nIDs = 10g - 5 < 10g - 4$, agent $a_{i-(g'-1)}$ does not observe the required number $10g - 4$ of IDs. Thus, $a_{i-1}, a_{i-2}, \dots, a_{i-(g'-1)}$ also observed less than $10g - 4$ IDs and they stay at the same node (a_i 's node) by the

Table 2. Global variables used in the proposed algorithm.

Variables for agent a_i

| Type | Name | Meaning | Initial value |
|-------|----------------|--|---------------|
| int | $a_i.rounds$ | number of rounds from some round | 1 |
| int | $a_i.nIDs$ | number of different IDs that a_i has observed from some round | 0 |
| int | $a_i.nVisited$ | number of nodes that a_i has ever visited | 0 |
| int | $a_i.rank$ | ordinal number of how its ID is small among IDs of agents at the same node | 0 |
| array | $a_i.ids[]$ | sequence of IDs that a_i has observed | \perp |

Variables for node v_j

| Type | Name | Meaning | Initial value |
|---------|---------------|---|---------------|
| int | $v_j.id$ | ID stored by v_j | \perp |
| int | $v_j.nAgents$ | number of agents staying at v_j | 0 |
| boolean | $v_j.fMarked$ | whether v_j is visited by a forward group or not | false |
| boolean | $v_j.bMarked$ | whether v_j is visited by a backward group or not | false |
| boolean | $v_j.candi$ | whether v_j is a gathering-candidate node or not | false |

Algorithm 1 The behavior of agent a_i in the semi-selection phase (v_j is the current node of a_i .)

Main Routine of Agent a_i

```

1:  $v_j.id := a_i.id, a_i.ids[a_i.nIDs] := v_j.id$ 
2:  $a_i.nIDs := a_i.nIDs + 1, v_j.nAgents := v_j.nAgents + 1,$ 
3: while  $a_i.rounds < 3n$  do
4:   if  $(a_i.nIDs < 10g - 4) \wedge (v_j.nAgents < 2g)$  then
5:      $v_j.nAgents := v_j.nAgents - 1$ 
6:     Try to move from the current node  $v_j$  to the forward node  $v_{j+1}$ 
7:     if  $(a_i$  reached  $v_{j+1}$  (that becomes new  $v_j$ ))  $\wedge (v_j.id \neq \perp)$  then
8:        $a_i.ids[a_i.nIDs] := v_j.id, a_i.nIDs := a_i.nIDs + 1$ 
9:     end if
10:     $v_j.nAgents := v_j.nAgents + 1, a_i.rounds := a_i.rounds + 1$ 
11:   end if
12: end while
13: if  $(v_j.nAgents \geq 2g) \vee ((a_i.nIDs \geq 8g - 3) \wedge (\forall h \in [0, 8g - 2] \setminus \{4g - 2\}; a_i.ids[4g - 2] <$ 
     $a_i.id[h]))$  then
14:    $v_j.candi := true$ 
15:   Terminate the semi-selection phase and enter the semi-gathering phase
16: end if

```

similar discussion of Lemma 1. Since $g' - 1 \geq (10g - 4) - (8g - 4) - 1 = 2g - 1$ holds, at least $2g$ agents (including a_i) stay at the same node v_j and thus $v_j.candi$ is set to true. Next, (b) if $a_i.nIDs \geq 8g - 3$ holds, a_i recognizes that a_{min} 's ID is its own ID and the ID is the minimum among the $8g - 3$ IDs. Hence, a_i sets $v_j.candi = true$ at the current node v_j . Therefore, the lemma follows. \square

Algorithm 2 The behavior of agent a_i in the semi-gathering phase (v_j is the current node of a_i .)

Main Routine of Agent a_i

```

1:  $a_i.rounds := 1, a_i.nIDs := 1$ 
2: while  $(a_i.rounds < 3n) \wedge (a_i.nIDs \neq 4g - 1)$  do
3:   if  $v_j.candi = false$  then
4:      $v_j.nAgents := v_j.nAgents - 1$ 
5:     Try to move from the current node  $v_j$  to the forward node  $v_{j+1}$ 
6:     if  $(a_i$  reached  $v_{j+1}$  (that becomes new  $v_j$ ))  $\wedge (v_j.id \neq \perp)$  then  $a_i.nIDs :=$ 
        $a_i.nIDs + 1$ 
7:      $v_j.nAgents := v_j.nAgents + 1$ 
8:     if  $v_j.nAgents \geq 2g$  then  $v_j.candi := true$ 
9:   end if
10:   $a_i.rounds = a_i.rounds + 1$ 
11: end while
12: Terminate the semi-gathering phase and enter the achievement phase

```

4.2 Semi-gathering phase

In this phase, agents aim to make a configuration such that at least $2g$ agents exist at some node. By Lemma 3, there exists at least one gathering-candidate node v_j with $v_j.candi = true$ at the end of the semi-selection phase. In the following, we call such a candidate node v_{candi} . Then, if less than $2g$ agents exist at v_{candi} , $4g - 2$ agents in total that already stay at v_{candi} and exist in v_{candi} 's backward direction try to stay at v_{candi} . Concretely, in this phase, for $3n$ rounds each agent tries to move forward until it stays v_{candi} or at least $2g$ agents exist at the current node. Then, due to link-missings, it is possible that only less than $2g$ agents gather at v_{candi} after the movement. In this case, we can show by the similar discussion of Lemma 1 that all the agents that do not reach v_{candi} among the $4g - 2$ agents stay at the same node. Then, the $4g - 1$ agents (the $4g - 2$ agents and the agent originally staying at v_{candi}) are partitioned into two groups and at least one group has at least $2g$ agents in any partition. Thus, agents can make a configuration such that at least $2g$ agents exist at some node.

The pseudocode of the semi-gathering phase is described in Algorithm 2. Note that, during the movement, when agents are blocked few times and they do not stay at a node with at least $2g$ agents, agents may require the total number of more than $O(gn)$ moves. To avoid this, each agent stop moving when it observed $4g - 1$ IDs even if it does not stay at a node with at least $2g$ agents (line 2).

Concerning the semi-gathering phase, we have the following lemma.

Lemma 4. *After finishing the semi-gathering phase, there exists at one node v_j with $v_j.nAgents \geq 2g$.*

Proof. We consider a configuration such that there exists no node with at least $2g$ agents at the beginning of the semi-gathering phase. By Lemma 3, there exists

at least one node v_j with $v_j.candi = true$, and $4g - 2$ agents in total that already stay at v_j and exist in v_j 's backward direction try to stay at v_j by Algorithms 1 and 2. Then, by the similar discussion of the proof of Lemma 1, after executing Algorithm 2 for $3n$ rounds, all agents among the $4g - 2$ agents that do not reach v_j stay at the same node. Thus, the $4g - 1$ agents (the $4g - 2$ agents and the agent originally staying at v_j) are partitioned into two groups and at least one group has at least $2g$ agents in any partition. Therefore, the lemma follows. \square

4.3 Achievement phase

In this phase, agents aim to achieve g -partial gathering. By Lemma 4, there exists at least one node with at least $2g$ agents as in Section 3.2. The difference from Section 3.2 is that there may exist more than two nodes with agents and there may exist several nodes each of which has at least $2g$ agents. Also from this situation, agents can achieve g -partial gathering using the same method as that in Section 3.2, that is, (1) agents staying at a node with at least $2g$ agents are partitioned into a forward group and a backward group and they try to move forward and backward respectively, and (2) when a forward group (resp., a backward group) visits a node with less than $2g$ agents, the less than $2g$ agents join the forward group (resp., a backward group).

An example is given in Fig. 3. In Fig. 3 (a), there exist two nodes v_p and v_q each of which has 6 ($= 2g$) agents. Hence, a forward group A_{f_p} and a backward group A_{b_p} (resp., A_{f_q} and A_{b_q}) start moving from node v_p (resp., from node v_q). From (a) to (b), A_{b_p} reaches node v_ℓ with less than $2g$ agents, and the less than $2g$ agents join A_{b_p} and try to move backward. From (b) to (e), we assume that A_{f_q} continues to be blocked by a missing link. From (b) to (c), A_{f_p} and A_{b_q} crossed and they recognize the fact by the existence of flags, and they terminate the algorithm execution. From (c) to (d), A_{b_p} reaches node v_m with less than $2g$ agents, and the less than $2g$ agents join A_{b_p} and try to move backward. Then, since the number num of agents in the updated A_{b_p} is 7 ($> 2g$), by using their IDs, only g agents continue to try moving backward and the remaining $num - g$ agents terminate the algorithm execution there ((d) to (e)). By this behavior, during this phase each link is passed by at most $2g$ agents and the achievement phase can be achieved with the total number of $O(gn)$ moves. From (e) to (f), A_{f_q} and A_{b_p} reach some node simultaneously and recognize the fact by flags, and they terminate the algorithm execution and agents achieve g -partial gathering.

Concerning the achievement phase, we have the following lemma.

Lemma 5. *After executing the achievement phase, agents achieve g -partial gathering.*

We have the following theorem for the proposed algorithm.

Theorem 4. *When $k \geq 8g - 3$ holds, the proposed algorithm solves the g -partial gathering problem in dynamic rings with $O(n)$ rounds and the total number of $O(gn)$ moves.*



Fig. 3. An execution example of the achievement phase ($g = 3$).

5 Conclusion

In this paper, we considered the g -partial gathering problem in bidirectional dynamic rings and considered the solvability of the problem and the move complexity, focusing on the relationship between values of k and g . First, when $3g \leq k \leq 8g - 2$, we showed that the proposed algorithm can solve the problem with $O(n)$ rounds and the total number of $O(kn)$ moves. Next, when $k \geq 8g - 3$, we showed that the proposed algorithm can solve the problem with $O(n)$ rounds and the total number of $O(gn)$ moves. These results show that, when $k \geq 3g$, the g -partial gathering problem can be solved also in dynamic rings. In addition, since $k = O(g)$ holds when $3g \leq k \leq 8g - 2$ holds like the first case, the both proposed algorithms can achieve g -partial gathering with the asymptotically optimal total number of agent moves.

Future works are as follows. First, we consider the solvability in case of $2g \leq k < 3g$. Second, when $3g \leq k < 8g - 3$, we consider whether agents can achieve g -partial gathering with the total number of moves smaller than $O(kn)$ or not. Finally, we will consider agents with weaker capabilities, e.g., agents without distinct IDs, without chirality, or agents that behave semi-synchronously or asynchronously. In any of the above cases, we conjecture that agents cannot solve the problem or require more total number of moves than the proposed algorithms.

Acknowledgement

This work was partially supported by JSPS KAKENHI Grant Number 18K18029, 18K18031, 20H04140, 20KK0232, and 21K17706; the Hibi Science Foundation; and Foundation of Public Interest of Tatematsu.

References

1. Baba, D., Izumi, T., Ooshita, F., Kakugawa, H., Masuzawa, T.: Linear time and space gathering of anonymous mobile agents in asynchronous trees. *Theoretical Computer Science* **478**, 118–126 (2013)
2. Das, S., Luna, D.G., Mazzei, D., Prencipe, G.: Compacting oblivious agents on dynamic rings. *PeerJ Computer Science* **7**, 1–29 (2021)
3. Dieudonné, Y., Pelc, A.: Anonymous meeting in networks. *Algorithmica* **74**(2), 908–946 (2016)
4. Flocchini, P., Kranakis, E., Krizanc, D., Santoro, N., Sawchuk, C.: Multiple mobile agent rendezvous in a ring. *LATIN* pp. 599–608 (2004)
5. Gotoh, T., Sudo, Y., Ooshita, F., Kakugawa, H., Masuzawa, T.: Group exploration of dynamic tori. *ICDCS* pp. 775–785 (2018)
6. Gray, R.S., Kotz, D., Cybenko, G., Rus, D.: D’agents: Applications and performance of a mobile-agent system. *Softw., Pract. Exper.* **32**(6), 543–573 (2002)
7. Kranakis, E., Krizanc, D., Markou, E.: Mobile agent rendezvous in a synchronous torus. *LATIN* pp. 653–664 (2006)
8. Kranakis, E., Krozanc, D., Markou, E.: The Mobile Agent Rendezvous Problem in the Ring. *Synthesis Lectures on Distributed Computing Theory*, Vol. 1 (2010)
9. Kranakis, E., Santoro, N., Sawchuk, C., Krizanc, D.: Mobile agent rendezvous in a ring. *ICDCS* pp. 592–599 (2003)
10. Lange, D., Oshima, M.: Seven good reasons for mobile agents. *CACM* **42**(3), 88–89 (1999)
11. Luna, D.G., Dobrev, S., Flocchini, P., Santoro, N.: Distributed exploration of dynamic rings. *Distributed Computing* **33**(1), 41–67 (2020)
12. Luna, D.G., Flocchini, P., Pagli, L., Prencipe, G., Santoro, N., Viglietta, G.: Gathering in dynamic rings. *Theoretical Computer Science* **811**, 79–98 (2018)
13. Shibata, M., Kawai, S., Ooshita, F., Kakugawa, H., Masuzawa, T.: Partial gathering of mobile agents in asynchronous unidirectional rings. *Theoretical Computer Science* **617**, 1–11 (2016)
14. Shibata, M., Kawata, N., Sudo, Y., Ooshita, F., Kakugawa, H., Masuzawa, T.: Move-optimal partial gathering of mobile agents without identifiers or global knowledge in asynchronous unidirectional rings. *Theoretical Computer Science* **822**, 92–109 (2020)
15. Shibata, M., Nakamura, D., Ooshita, F., Kakugawa, H., Masuzawa, T.: Partial gathering of mobile agents in arbitrary networks. *IEICE Transactions on Information and Systems* **102**(3), 444–453 (2019)
16. Shibata, M., Ooshita, F., Kakugawa, H., Masuzawa, T.: Move-optimal partial gathering of mobile agents in asynchronous trees. *Theoretical Computer Science* **705**, 9–30 (2018)
17. Shibata, M., Sudo, Y., Nakamura, J., Kim, Y.: Uniform deployment of mobile agents in dynamic rings. *SSS* pp. 248–263 (2020)
18. Shibata, M., Tixeuil, S.: Partial gathering of mobile robots from multiplicity-allowed configurations in rings. *SSS* pp. 264–279 (2020)