

# Partial Gathering of Mobile Agents in Dynamic Tori

Masahiro Shibata  

Kyushu Institute of Technology, Fukuoka, Japan

Naoki Kitamura  

Osaka University, Japan

Ryota Eguchi  



NAIST, Nara, Japan

Yuichi Sudo  

Hosei University, Tokyo, Japan

Junya Nakamura  

Toyohashi University of Technology, Aichi, Japan

Yonghwan Kim  

Nagoya Institute of Technology, Aichi, Japan

---

## Abstract

In this paper, we consider the partial gathering problem of mobile agents in synchronous dynamic tori. 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. So far, in almost cases, the partial gathering problem has been considered in static graphs. As only one exception, it is considered in a kind of dynamic rings called 1-interval connected rings, that is, one of the links in the ring may be missing at each time step. In this paper, we consider partial gathering in another dynamic topology. Concretely, we consider it in  $n \times n$  dynamic tori such that each of row rings and column rings is represented as a 1-interval connected ring. In such networks, when  $k = O(gn)$ , focusing on the relationship between the values of  $k, n$ , and  $g$ , we aim to characterize the solvability of the partial gathering problem and analyze the move complexity of the proposed algorithms when the problem can be solved. First, we show that agents cannot solve the problem when  $k = o(gn)$ , which means that  $\Omega(gn)$  agents are necessary to solve the problem. Second, we show that the problem can be solved with the total number of  $O(gn^3)$  moves when  $2gn + 2n - 1 \leq k \leq 2gn + 6n + 16g - 12$ . Finally, we show that the problem can be solved with the total number of  $O(gn^2)$  moves when  $k \geq 2gn + 6n + 16g - 11$ . From these results, we show that our algorithms can solve the partial gathering problem in dynamic tori with the asymptotically optimal number  $\Theta(gn)$  of agents. In addition, we show that agents require a total number of  $\Omega(gn^2)$  moves to solve the partial gathering problem in dynamic tori when  $k = \Theta(gn)$ . Thus, when  $k \geq 2gn + 6n + 16g - 11$ , our algorithm can solve the problem with asymptotically optimal number  $O(gn^2)$  of agent moves.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Self-organization

**Keywords and phrases** distributed system, mobile agents, partial gathering, dynamic tori

**Digital Object Identifier** 10.4230/LIPIcs.SAND.2023.2

**Funding** This work was partially supported by JSPS KAKENHI Grant Number 18K18031, 20H04140, 20KK0232, 21K17706, and 22K11971; and Foundation of Public Interest of Tatematsu.



© Masahiro Shibata, Naoki Kitamura, Ryota Eguchi, Yuichi Sudo, Junya Nakamura, and Yonghwan Kim;

licensed under Creative Commons License CC-BY 4.0

2nd Symposium on Algorithmic Foundations of Dynamic Networks (SAND 2023).

Editors: David Doty and Paul Spirakis; Article No. 2; pp. 2:1–2:22

Leibniz International Proceedings in Informatics



Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

## 1 Introduction

### 1.1 Background and Related Work

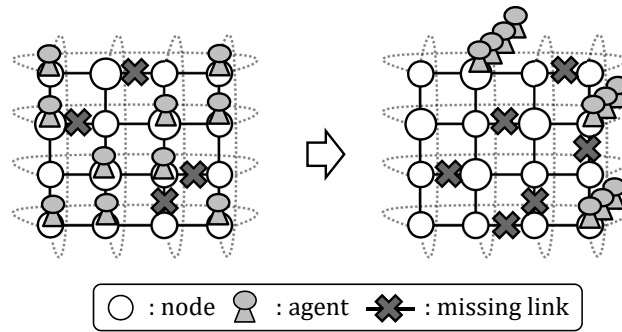
A *distributed system* comprises a set of computing entities (*nodes*) connected by communication links. As a promising design paradigm of distributed systems, (mobile) agents have attracted much attention [7]. The agents can traverse the system, carrying information collected at visited nodes, and execute 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 the 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 [9, 13], trees [5, 1], tori [8], and arbitrary networks [3, 4].

Recently, a variant of the total gathering problem, called the  *$g$ -partial gathering problem* [14], 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. Notice that the  $g$ -partial gathering problem is equivalent to the total gathering problem when  $k < 2g$ . 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.

As related work, Shibata et al. considered the  $g$ -partial gathering problem in rings [14, 15, 20], trees [17], and arbitrary networks [16]. In [14, 15], 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. In [20], 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 the case of using mobile robots, 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 in both agent and robot models. Thus, the above results are asymptotically optimal in terms of the total number of moves, and the number  $O(gn)$  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.

While 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, 6], the compacting and grouping problem [2], and the uniform deployment problem [18] are considered in dynamic graphs. Also, in [19], the  $g$ -partial



■ **Figure 1** An example of the  $g$ -partial gathering problem in a  $4 \times 4$  dynamic torus ( $g = 3$ ).

gathering problem is considered in a kind of dynamic rings called *1-interval connected rings* [12, 11, 18], that is, one of the links in the ring may be missing at each time step. In such networks, focusing on the relationship between the values of  $k$  and  $g$ , they clarified the solvability of the  $g$ -partial gathering problem and analyzed the move complexity of the proposed algorithms when the problem can be solved. As a result, they showed that (i) when  $k \leq 2g$ , the  $g$ -partial gathering problem cannot be solved, (ii) when  $2g + 1 \leq k \leq 3g - 2$ , the problem can be solved with the total number of  $O(gn \log g)$  moves, and (iii) when  $k \geq 3g - 1$ , the problem can be solved with the asymptotically optimal total number  $O(gn)$  of agent moves.

## 1.2 Our Contribution

In this paper, we consider the  $g$ -partial gathering problem of mobile agents in another dynamic topology. Concretely, we consider the problem in  $n \times n$  dynamic tori such that each of row rings and column rings is represented as a 1-interval connected ring. An example is given in Fig. 1. An edge with a cross means that it is missing. In this paper, we assume that each node has a whiteboard. In addition, we assume that agents have distinct IDs, common sense of direction, knowledge of  $k$  and  $n$ , and behave fully synchronously. In such settings, when  $k = O(gn)$ , focusing on the relationship between the values of  $k$ ,  $n$ , and  $g$ , we aim to characterize the solvability of the  $g$ -partial gathering problem and analyze the time and move complexities of the proposed algorithms when the problem can be solved.

We summarize our results in Table 1. First, we show that agents cannot solve the  $g$ -partial gathering problem when  $k = o(gn)$ , which means that  $\Omega(gn)$  agents are necessary to solve the problem. Second, we show that the problem can be solved with  $O(n^2)$  rounds and the total number of  $O(gn^3)$  moves when  $2gn + 2n - 1 \leq k \leq 2gn + 6n + 16g - 12$ . Finally, we show that the problem can be solved with  $O(n^2)$  rounds and the total number of  $O(gn^2)$  moves when  $k \geq 2gn + 6n + 16g - 11$ . From these results, we show that our algorithms can solve the  $g$ -partial gathering problem in dynamic tori with the asymptotically optimal number  $\Theta(gn)$  of agents. In addition, we show that agents require a total number of  $\Omega(gn^2)$  moves to solve the  $g$ -partial gathering problem in  $n \times n$  dynamic tori when  $k = \Theta(gn)$ . Thus, when  $k \geq 2gn + 6n + 16g - 11$ , our algorithm can solve the problem with asymptotically optimal number  $O(gn^2)$  of agent moves.

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

■ **Table 1** Results of  $g$ -partial gathering for agents with distinct IDs in dynamic tori when  $k = O(gn)$  ( $n$ : #nodes,  $k$ : #agents).

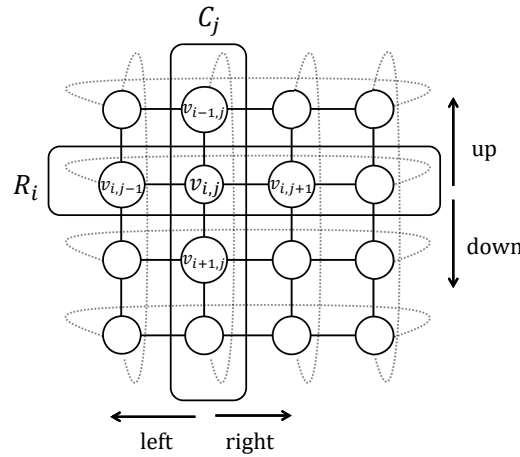
	Result 1 (Sec. 3)	Result 2 (Sec. 5)	Result 3 (Sec. 6)
Relation between $k$ and $g$	$k = o(gn)$	$k \geq 2gn + 2n - 1$ and $k \leq 2gn + 6n + 16g - 12$	$k \geq 2gn + 6n + 16g - 11$
Solvable/Unsolvable	Unsolvable	Solvable	Solvable
Time complexity	-	$O(n^2)$	$O(n^2)$
Total number of agent moves	-	$O(gn^3)$	$\Theta(gn^2)$

## 2 Preliminaries

### 2.1 System Model

We basically follow the model defined in [6]. An  $n \times n$  *dynamic torus*  $T$  is defined as 2-tuple  $T = (V, E)$ , where  $V$  is a set of nodes  $\{v_{i,j} \mid 0 \leq i, j \leq n-1\}$  and  $E$  is a set of links  $\{(v_{i,j}, v_{(i+1) \bmod n, j}), (v_{i,j}, v_{i, (j+1) \bmod n}) \mid 0 \leq i, j \leq n-1\}$ . For simplicity, we denote  $v_{(i+i') \bmod n, (j+j') \bmod n}$  by  $v_{i+i', j+j'}$  for any integers  $i, i', j$ , and  $j'$ . The *distance* between nodes  $v_{i,j}$  and  $v_{p,q}$  is defined as  $\min\{i-p, p-i\} + \min\{j-q, q-j\}$ . Notice that this definition of the distance is correct when no corresponding link that connects  $v_{i,j}$  and  $v_{p,q}$  is missing. We call the direction from  $v_{i,j}$  to  $v_{i,j+1}$  (resp., to  $v_{i+1,j}$ , to  $v_{i,j-1}$ , and to  $v_{i-1,j}$ ) the *right* (resp., *down*, *left*, and *up*) direction. Intuitively, torus  $T$  comprises  $n$  row rings and  $n$  column rings. A row ring  $R_i$  (resp., a column ring  $C_j$ ) is a subgraph of  $T$  induced by  $\{v_{i,j} \mid 0 \leq j \leq n-1\}$  (resp.,  $\{v_{i,j} \mid 0 \leq i \leq n-1\}$ ) (see Fig. 2). We assume that each of row rings and column rings is *1-interval connected*, that is, one of the links in the ring may be missing at each time step, and which link is missing is controlled by an *adversarial scheduler*. Then, since each of the row and column rings is 1-interval connected, the dynamic torus  $T$  is always connected. In addition, we assume that nodes are anonymous, i.e., they do not have IDs (and thus the indices of nodes are used just for notation purposes). Every node  $v_{i,j} \in V$  has a whiteboard that agents at node  $v_{i,j}$  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 links, that is, they can move from  $v_{i,j}$  to  $v_{i,j+1}$  (move right), from  $v_{i,j}$  to  $v_{i+1,j}$  (move down), from  $v_{i,j}$  to  $v_{i,j-1}$  (move left), or from  $v_{i,j}$  to  $v_{i-1,j}$  (move up), for any  $i$  and  $j$ . Agents have distinct IDs and knowledge of  $k$  and  $n$ . Agents have the common sense of directions, that is, they agree on the directions of right, down, left, and up in the torus. In addition, agents cannot detect whether other agents exist at the current node or not. An agent  $a_h$  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 state of  $a_h$  and the current node's whiteboard, the next states of  $a_h$  and the whiteboard, and whether  $a_h$  moves to its neighboring node or not. The last element  $M = \{\text{null}, \text{right}, \text{down}, \text{left}, \text{up}\}$  in  $\delta$  represents which direction  $a_h$  tries to move in the next movement. The value "null" means staying at the current node. We assume that  $\delta(s_{final}, w_{ij}) = (s_{final}, w_{ij}, \text{null})$  holds for any state  $w_{ij} \in W$ , which means that  $a_h$  never changes its state, updates the contents of the current node  $v_{i,j}$ 's whiteboard, or leaves  $v_{i,j}$  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.



■ **Figure 2** An example of a torus graph ( $n = 4$ ).

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  $v_{i,j}$  in the initial configuration, the initial state of  $v'_{i,j}$ '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, agents 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  $v_{i,j}$  to  $v_{i,j+1}$ ) but the corresponding link is missing, we say that the agent is *blocked*, and it still exists at  $v_{i,j}$  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 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 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  $g$ -partial gathering problem as follows.

► **Definition 1.** *An algorithm solves the  $g$ -partial gathering problem in dynamic tori 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 required for agents to solve the problem) and the total number of agent moves.

### 3 The case of $k = o(gn)$

When  $k = o(gn)$ , the following impossibility result holds. Intuitively, this is because (i) when there exists an agent at each of nodes  $v_{0,0}, v_{1,1}, \dots, v_{n-1,n-1}$  in the initial configuration, the adversary can make the  $n$  agents never leave their starting nodes, and thus (ii) to achieve  $g$ -partial gathering, it is necessary that there exist at least  $g$  agents at each of the  $n$  nodes, which requires at least  $gn$  agents in total.

► **Theorem 2.** *When  $k = o(gn)$  holds, the  $g$ -partial gathering problem cannot be solved in dynamic tori.*

### 4 Lower bound on the total number of agent moves when $k = \Theta(gn)$

By Theorem 3, since at least  $\Omega(gn)$  agents are necessary to solve the problem, in the following, we assume that there exist  $\Theta(gn)$  agents in the torus. Then, we have the following theorem on the lower bound of the total number of agent moves. Intuitively, this is because when there exists an agent at each of nodes  $v_{0,0}, v_{1,1}, \dots, v_{n-1,n-1}$  and each of the other agents is placed at a node with distance  $\Omega(n)$  from  $v_{i,i}$  ( $0 \leq i \leq n-1$ ) in the initial configuration, and when the adversary makes agents at  $v_{i,i}$  never leave their starting node, it is necessary that at least  $gn - n$  agents need to stay at either  $v_{0,0}, v_{1,1}, \dots$ , or  $v_{n-1,n-1}$ , which requires the total number of  $\Omega(gn^2)$  moves in total.

► **Theorem 3.** *A lower bound on the total number of agent moves to solve the  $g$ -partial gathering problem in dynamic tori when  $k = \Theta(gn)$  is  $\Omega(gn^2)$ .*

### 5 The case of $2gn + 2n - 1 \leq k \leq 2gn + 6n + 16g - 12$

In this section, when  $2gn + 2n - 1 \leq k \leq 2gn + 6n + 16g - 12$ , we propose an algorithm to solve the  $g$ -partial gathering problem with  $O(n^2)$  rounds and the total number of  $O(gn^3)$  moves. In the algorithm, agents aim to make a configuration such that there exist at least  $2g + 1$  agents in each of row rings. Then, agents achieve  $g$ -partial gathering by applying the existing method [19] for  $g$ -partial gathering in 1-interval connected rings to each row ring independently. To this end, agents repeat the following two phases  $n$  times: the counting phase and the adjusting phase. In the counting phase, each agent in row ring  $R_i$  tries to move horizontally to count the number of agents existing in  $R_i$  at the beginning of the counting phase. In the adjusting phase, several agents in row rings with a lot of agents try to move vertically to a row ring with less agents at the beginning of the adjusting phase.

The overall pseudocode is given in Algorithm 1 (the pseudocodes of the counting phase and the adjusting phase are given as procedures *Counting()* and *Adjusting()*, respectively). Global variables used in the algorithm are given in Table 2. In the following subsections, we explain the details of each phase.

#### 5.1 Counting phase

The aim of this phase is that each agent  $a_h$  in row ring  $R_i$  calculates the total number of agents existing in  $R_i$  at the beginning of this counting phase by making either of the following two configurations: (i) Each agent  $a_h$  travels once around the row ring  $R_i$  and gets IDs of

■ **Table 2** Global variables used in proposed algorithms.

**Variables for agent  $a_h$ .**

Type	Name	Meaning	Initial value
int	$a_h.phase$	phase number stored by $a_h$	0 or 1
int	$a_h.rounds$	number of rounds from some round	1
int	$a_h.nVisited$	number of nodes that $a_i$ has visited from some round	0
int	$a_h.nAgentsRowRing$	number of agents existing in the current row ring	0
int	$a_h.rank$	ordinal number of how its ID is small among IDs of agents at the same node	0
int	$a_h.dir$	direction to which $a_h$ tries to move (1: right or down, -1: left or up)	0
array	$a_h.IDs[p]$	list of IDs that $a_h$ has observed in the phase $p$	$\perp$

**Variables for node  $v_{i,j}$ .**

Type	Name	Meaning	Initial value
int	$v_{ij}.phase$	phase number stored by $v_{i,j}$	1
int	$v_{ij}.nAgentsCurrent$	number of agents staying at the current node $v_{i,j}$	0
array	$v_{ij}.IDs[p]$	list of IDs stored by $v_{i,j}$ in phase $p$	$\perp$
array	$v_{ij}.nAgentsAdjust[p]$	number of agents existing in the current row ring at the beginning of the current adjusting phase with phase number $p$	$\perp$

■ **Algorithm 1** The behavior of agent  $a_h$  for the proposed algorithm when  $2gn + 2n - 1 \leq k \leq 2gn + 6n + 16g - 12$ .

**Main Routine of Agent  $a_h$ :**

- 1  $a_h.phase := 1$
- 2 **while**  $a_h.phase \leq n$  **do**
- 3      $Counting()$
- 4      $Adjusting()$
- 5      $a_h.phase++$
- 6 Apply the existing method [19] to the currently staying row ring
- 7 Terminate the algorithm execution

all the agents existing in  $R_i$ , or (ii) it detects that all the agents existing in  $R_i$  are at the same node. To this end, we use an idea similar to [12] which considers total gathering in 1-interval connected rings. First, each agent  $a_h$  writes its ID  $a_h.id$  and the current phase number  $a_h.phase$  to the variables  $v_{ij}.IDs[a_h.phase]$  and  $v_{ij}.phase$ , respectively, on the current node  $v_{i,j}$ 's whiteboard and then tries to move right for  $3n$  rounds. During the movement,  $a_h$  memorizes values of observed IDs that are written in the current counting phase to array  $a_h.IDs[a_h.phase]$ . After the  $3n$  rounds, the number  $a_h.nVisited$  of nodes that  $a_h$  has visited from the beginning of this counting phase is (a) at least  $n$  or (b) less than  $n$  due to missing links. In case (a),  $a_h$  must have completed traveling once around the row ring  $R_i$ . Hence,  $a_h$  can calculate the number  $a_h.nAgentsRowRing$  of agents existing in  $R_i$  through the number of observed IDs (i.e.,  $|a_h.IDs[a_h.phase]|$ ). Thus, it reaches configuration (i). In case (b) (i.e.,  $a_h$  has visited less than  $n$  nodes during the  $3n$  rounds), we show in Lemma 4 that all the agents existing in  $R_i$  stay at the same node (they reach configuration (ii)). Thus, through the value  $v_{ij}.nAgentsCurrent$  of the current node  $v_{i,j}$ 's whiteboard representing the number of agents currently staying at  $v_{i,j}$ ,  $a_h$  can calculate  $a_h.nAgentsRowRing$ .

■ **Algorithm 2** Procedure *Counting()* ( $v_{i,j}$  is the current node of  $a_h$ ).

---

**Main Routine of Agent  $a_h$ :**

- 1  $v_{ij}.phase := a_h.phase, v_{ij}.IDs[v_{ij}.phase] := v_{ij}.IDs[v_{ij}.phase] \cup a_h.id, v_{ij}.nAgentsCurrent++$
- 2  $a_h.nVisited := 1, a_h.rounds := 1, a_h.IDs[a_h.phase] := v_{ij}.IDs[v_{ij}.phase]$
- 3 **while**  $a_h.rounds \leq 3n$  **do**
- 4      $v_{ij}.nAgentsCurrent--$
- 5     Try to move from the current node  $v_{i,j}$  to the right neighboring node  $v_{i,j+1}$
- 6     **if**  $a_h$  reached  $v_{i,j+1}$  (that becomes new  $v_{i,j}$ ) **then**
- 7          $a_h.nVisited++$
- 8         **if**  $(v_{ij}.phase = a_h.phase) \wedge (a_h.nVisited \leq n)$  **then**
- 9              $a_h.IDs[a_h.phase] := a_h.IDs[a_h.phase] \cup v_{ij}.IDs[v_{ij}.phase]$
- 10      $v_{ij}.nAgentsCurrent++, a_h.rounds++$
- 11 **if**  $a_h.nVisited < n$  **then**  $a_h.nAgentsRowRing := v_{ij}.nAgentsCurrent$  // all the agents in row ring  $R_i$  stay at the current node
- 12 **if**  $a_h.nVisited \geq n$  **then**  $a_h.nAgentsRowRing := |a_h.IDs[a_h.phase]|$  //  $a_h$  traveled once around the row ring
- 13 Terminate the counting phase and enter the adjusting phase

---

The pseudocode of the counting phase is described in Algorithm 2. Note that, during the counting phase, an agent  $a_h$  may visit more than  $n$  nodes and then the number of observed IDs is more than the number of agents in the row ring when it is blocked less than  $2n$  times. In this case,  $a_h$  stops measuring IDs when it has visited more than  $n$  nodes (line 8).

Concerning the counting phase, we have the following lemma.

► **Lemma 4.** *Let  $na_i$  be the number of agents existing in row ring  $R_i$  at the beginning of the current counting phase with phase number  $p$ . Then, at the end of the counting phase, each agent  $a_h$  existing in  $R_i$  in phase  $p$  stores the correct value of  $na_i$  to  $a_h.nAgentsRowRing$ .*

## 5.2 Adjusting phase

In this phase, several agents in a row ring with a lot of agents try to move vertically to a row ring with few agents to reduce the gap of the number of agents between row rings. Concretely, several agents in each row ring  $R_i$  first move horizontally in  $R_i$  and write the number of agents existing in  $R_i$  at the beginning of this adjusting phase (i.e.,  $a_h.nAgentsRowRing$  for agent  $a_h$ ), to each node's whiteboard of  $R_i$ . Then, several agents belonging to a row ring with at least  $2g + 3$  agents try to move in the torus vertically and stay at a node of a row ring with less than  $2g + 1$  agents at the beginning of this adjusting phase. By repeating this behavior and the counting phase explained before, agents eventually reach a configuration such that there exist at least  $2g + 1$  agents in each row ring (and then  $g$ -partial is achieved by applying the existing method [19] to each row ring independently).

First, we explain how to write the number  $na_i$  of agents in  $R_i$  at the beginning of this adjusting phase (or at the end of the counting phase just before) to each node's whiteboard of  $R_i$ . By Lemma 4, at the end of the counting phase just before, each agent  $a_h$  in row ring  $R_i$  knows the number  $na_i$  ( $= a_h.nAgentsRowRing$ ) of agents currently existing in  $R_i$ . In addition, by Algorithm 2,  $a_h$  can get the list of agent IDs existing in  $R_i$ . Among the agents, let  $a_1^i$  (resp.,  $a_2^i$ ) be the agent with the smallest (resp., the second smallest) ID. Then, for  $n$  rounds,  $a_1^i$  (resp.,  $a_2^i$ ) tries to move right (resp., left) and then  $a_1^i$  (resp.,  $a_2^i$ ) tries to move left (resp., right) for the next  $n$  rounds. During the movement,  $a_1^i$  and  $a_2^i$  write values of  $na_i$  for the current phase  $p$  to variable  $v_{ij}.nAgentsAdjust[p]$  of each node  $v_{i,j}$ 's whiteboard. By this



behavior, we show in Lemma 5 that every node in  $R_i$  is visited by  $a_1^i$  or  $a_2^i$  and the value of  $na_i$  is stored when there exist at least two agents in  $R_i$ . Notice that it is possible that there exists only one agent  $a$  in  $R_i$  at the beginning of some adjusting phase and  $a$  cannot visit all nodes in  $R_i$  and write necessary information to nodes' whiteboards in  $R_i$ . In this case, when some agents in a row ring other than  $R_i$  try to move vertically and visit some node in  $R_i$  (this method is explained in the next paragraphs), they can recognize that no information is written to the node of  $R_i$  and there exist less than  $2g + 1$  (actually one) agents in  $R_i$ .

Next, we explain how agents in a row ring with many agents move vertically to a node of a row ring with less agents. First, when  $na_i \geq 2g + 3$ , for  $3n$  rounds, each agent  $a_h$  in  $R_i$  tries to move right until reaching the node  $v_{min}^i$  where the smallest ID is written in the counting phase just before. Then, by the similar discussion of agents' behaviors and the proof for Lemma 4, all the agents in  $R_i$  that does not reach  $v_{min}^i$  stay at the same node  $v'_i$ . From such a situation,  $na_i - (2g + 1)$  agents in total try to move vertically to visit a node in a row ring with less than  $2g + 1$  agents (or node  $v_{i,j}$  with  $v_{i,j}.nAgentsAdjust[p] < 2g + 1$ ). Intuitively, among the  $na_i - (2g + 1)$  agents, around half agents try to move up and another half agents try to move down. Concretely, let  $a_h.rank$  be the ordinal number of how small its ID is among agents at the same node  $v_{i,j}$  ( $1 \leq a_h.rank \leq v_{i,j}.nAgentsCurrent$ ). Then, when all the  $na_i$  agents in  $R_i$  stay at the same node  $v_{min}^i$  (or  $v'_i$ ), each agent  $a_h$  with  $1 \leq a_h.rank \leq \lfloor (na_i - (2g + 1))/2 \rfloor$  (resp.,  $\lfloor (na_i - (2g + 1))/2 \rfloor + 1 \leq a_h.rank \leq na_i - (2g + 1)$ ) belongs to the *up group* (resp., *down group*), like Fig. 3(a). On the other hand, when there exist two nodes  $v_{min}^i$  and  $v'_i$  where an agent exists, let  $v_{more}^i$  (resp.,  $v_{less}^i$ ) be the node where at least  $\lceil na_i/2 \rceil$  (resp., at most  $\lfloor na_i/2 \rfloor$ ) agents exist and let  $na_{more}^i$  (resp.,  $na_{less}^i$ ) be the number of agents staying at  $v_{more}^i$  (resp.,  $v_{less}^i$ ). Then, when  $na_{more}^i \geq 2g + 1$ , each agent  $a_h$  at  $v_{more}^i$  with  $1 \leq a_h.rank \leq \lfloor (na_{more}^i - (2g + 1))/2 \rfloor$  (resp.,  $\lfloor (na_{more}^i - (2g + 1))/2 \rfloor + 1 \leq a_h.rank \leq na_{more}^i - (2g + 1)$ ) and each agent  $a_{h'}$  at node  $v_{less}^i$  with  $1 \leq a_{h'}.rank \leq \lfloor na_{less}^i/2 \rfloor$  (resp.,  $\lfloor na_{less}^i/2 \rfloor + 1 \leq a_{h'}.rank \leq na_{less}^i$ ) belong to the up group (resp., the down group), like Fig. 3(b). When  $na_{more}^i < 2g + 1$ , each agent  $a_h$  at  $v_{more}^i$  with  $1 \leq a_h.rank \leq \lfloor (na_i - (2g + 1))/2 \rfloor$  (resp.,  $\lfloor (na_i - (2g + 1))/2 \rfloor + 1 \leq a_h.rank \leq na_i - (2g + 1)$ ) belongs to the up group (reps., the down group), and agents at  $v_{less}^i$  do not try to move in this classification, like Fig. 3(c).

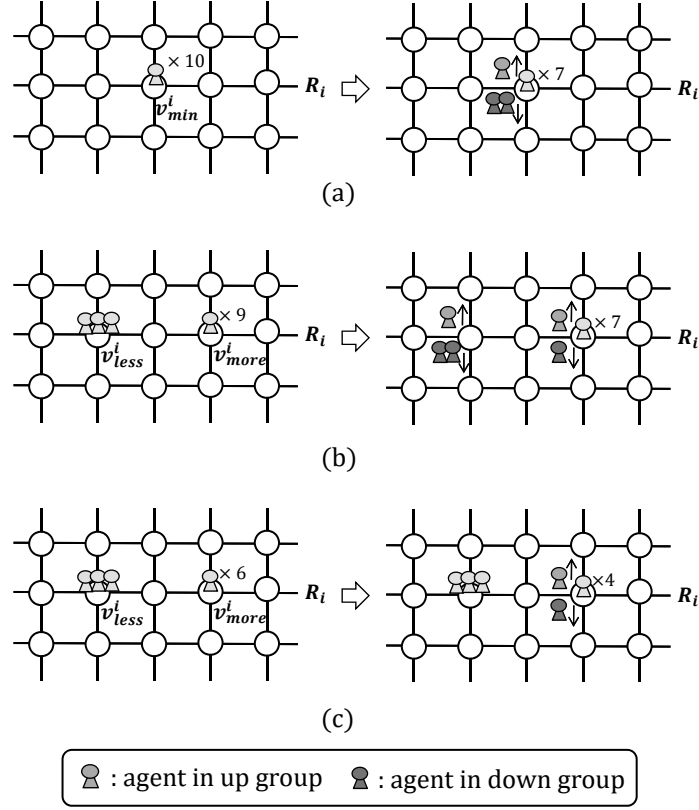
Thereafter, for  $n$  rounds, each agent in the up (resp., down) group tries to move up (resp., down) until it reaches a node in some row ring  $R_{i'}$  with  $na_{i'} < 2g + 1$  (or no value of  $na_{i'}$  is written). By this behavior, since each column ring is represented as a 1-interval connected ring, either an up group or a down group can visit a node in a row ring with less than  $2g + 1$  agents. By repeating such an adjusting phase and the previous counting phase  $n$  times in total, we show that agents eventually reach a configuration such that there exist at least  $2g + 1$  agents in each of row rings (Lemma 5). Thus, they achieve  $g$ -partial gathering by applying the existing method [19] to each row ring independently.

The pseudocode of the adjusting phase is described in Algorithm 3. In Algorithm 3, each agent uses procedure *DecideDirection()* to determine the vertical direction it should move (or it should keep staying at the current node), whose pseudocode is described in Algorithm 4. For simplicity, we omit how to calculate  $a_h.rank$  in Algorithm 4.

Concerning the adjusting phase, we have the following lemma.

► **Lemma 5.** *After executing the adjusting phase  $n$  times in total, agents reach a configuration such that there exist at least  $2g + 1$  agents in each of row rings.*

**Proof.** First, we simply show that, when there exist at least two agents in  $R_i$  at the beginning of the  $p$ -th adjusting phase, after  $a_1^i$  and  $a_2^i$  move right and left for  $2n$  rounds (lines 1 to 15 of Procedure *Adjusting()*), the correct number  $na_i$  of agents existing in a row ring  $R_i$  is stored to variable  $v_{i,j}.nAgentsAdjust[p]$  of each node  $v_{i,j}$ 's whiteboard in  $R_i$ . By Procedure



■ **Figure 3** Examples of forming up groups and down groups ( $g = 3$ ).

*Adjusting()*, agent  $a_1^i$  (resp.,  $a_2^i$ ) first tries to move right (resp., left) for  $n$  rounds. When  $a_1^i$  and  $a_2^i$  start their behaviors at the same node, since at most one link is missing in each of row rings at each round, every node is visited within this  $n$  rounds and the value of  $na_i$  is stored to each node's whiteboard in  $R_i$ . When  $a_1^i$  and  $a_2^i$  start their behaviors at different nodes and there exists at least one unvisited node during the  $n$  rounds, it means that  $a_1^i$  and  $a_2^i$  are blocked by the same link. In this case, for the next  $n$  rounds, they switch their directions and  $a_1^i$  (resp.,  $a_2^i$ ) tries to move left (resp., right). Then, by the similar discussion of the case when they start their movements at the same node, every node is visited within this  $n$  rounds and the value of  $na_i$  is stored to each node's whiteboard in  $R_i$ . Thus, when there exist at least two agents in  $R_i$  at the beginning of the  $p$ -th adjusting phase, the correct value of  $na_i$  is stored to  $v_{ij}.nAgentsAdjust[p]$  of each node  $v_{i,j}$ 's whiteboard in  $R_i$ .

In the following, we discuss the number of agents in each row ring after executing the adjusting phase  $n$  times. In the proof, we assume that  $k = 2gn + 2n - 1$  holds (we can show the lemma in the case of  $k > 2gn + 2n - 1$  similarly). At the beginning of some adjusting phase, we call a row ring  $R_i$  *enough* if there exist at least  $2g + 1$  agents in  $R_i$ . Otherwise, we call the row ring *lacking*. In addition, we define the number  $diff_{enough}^f$  (resp.,  $diff_{lack}^f$ ) of how more (resp., less) agents exist in row ring  $R_i$  compared to  $2g + 1$ , and it is calculated as  $na_i - (2g + 1)$  (resp.,  $(2g + 1) - na_i$ ). Notice that  $diff_{enough}^f$  and  $diff_{lack}^f$  are always non-negative numbers. Moreover, we define  $SumDiff_{enough}^f = \sum_{i|R_i \text{ is enough}} diff_{enough}^f$  (resp.,  $SumDiff_{lack}^f = \sum_{i|R_i \text{ is lacking}} diff_{lack}^f$ ). Then, we first show the following claim.

■ **Algorithm 3** Procedure *Adjusting()* ( $v_{i,j}$  is the current node of  $a_h$ ).

---

**Main Routine of Agent  $a_h$ :**

```

1  $a_h.rounds := 1$ 
2 if  $a_h.nAgentsRowRing \geq 2g + 3$  then
3   if  $a_h.id$  is smallest among  $a_h.IDs[a_h.phase]$  then  $a_h.dir := 1$ 
4   if  $a_h.id$  is the second smallest among  $a_h.IDs[a_h.phase]$  then  $a_h.dir := -1$ 
5   while  $a_h.rounds \leq n$  do
6      $v_{ij}.nAgentsCurrent--$ 
7     Try to move from the current node  $v_{i,j}$  to the neighboring node  $v_{i,j+a_h.dir}$ 
8     if  $a_h$  reached  $v_{i,j+a_h.dir}$  (that becomes new  $v_{i,j}$ ) then
9        $v_{ij}.nAgentsAdjust[a_h.phase] := a_h.nAgentsRowRing$ 
10       $v_{ij}.nAgentsCurrent++, a_h.rounds++$ 
11    $a_h.dir := a_h.dir \times (-1)$ 
12   while  $a_h.rounds \leq 2n$  do
13      $v_{ij}.nAgentsCurrent--$ 
14     Try to move from the current node  $v_{i,j}$  to the neighboring node  $v_{i,j+a_h.dir}$ 
15     if  $a_h$  reached  $v_{i,j+a_h.dir}$  (that becomes new  $v_{i,j}$ ) then
16        $v_{ij}.nAgentsAdjust[a_h.phase] := a_h.nAgentsRowRing$ 
17        $v_{ij}.nAgentsCurrent++, a_h.rounds++$ 
18    $a_h.dir := 1$ 
19   while  $a_h.rounds \leq 5n$  do
20     if  $v_{i,j}$  is not the node where  $\min\{a_h.IDs[a_h.phase]\}$  is not written in
21      $v_{ij}.IDs[v_{ij}.phase]$  then
22        $v_{ij}.nAgentsCurrent--$ 
23       Try to move from the current node  $v_{i,j}$  to the right neighboring node  $v_{i,j+1}$ 
24        $v_{ij}.nAgentsCurrent++$ 
25      $a_h.rounds++$ 
26    $a_h.dir := DecideDirection()$ 
27   while  $a_h.rounds \leq 6n$  do
28     if  $v_{ij}.nAgentsAdjust[a_h.phase] \geq 2g + 1$  then
29        $v_{ij}.nAgentsCurrent--$ 
30       Try to move from the current node  $v_{i,j}$  to the right neighboring node  $v_{i+a_h.dir,j}$ 
31        $v_{ij}.nAgentsCurrent++$ 
32      $a_h.rounds++$ 
33 else
34   While  $a_h.rounds \leq 6n$  do  $a_h.rounds++$ 
35 Terminate the adjusting phase
36 //  $a_h$  increments its phase number and starts the next counting phase

```

---

▷ **Claim 6.** At the end of each adjusting phase, compared to the beginning of the adjusting phase, either of the following two properties holds: (i) The value of  $SumDiff_{lack}$  at least halves, or (ii) The number of lacking rings decreases by at least one.

Proof. First, we briefly show that  $SumDiff_{enough} = SumDiff_{lack} + n - 1$  always holds. Let  $nr_{enough}$  (resp.,  $nr_{lack}$ ) be the number of enough (resp., lacking) row rings. Notice that  $nr_{enough} = n - nr_{lack}$  holds. Then, there exist  $nr_{lack} \times (2g + 1) - SumDiff_{lack}$  agents in total in lacking row rings and hence there exist  $2gn + 2n - 1 - (nr_{lack} \times (2g + 1) - SumDiff_{lack})$  agents in total in enough row rings. Since the total number of agents in enough row rings can be also represented as  $nr_{enough} \times (2g + 1) + SumDiff_{enough}$ ,  $2gn + 2n - 1 - (nr_{lack} \times (2g + 1) - SumDiff_{lack}) = nr_{enough} \times (2g + 1) + SumDiff_{enough}$  holds. Thus, since  $nr_{enough} = n - nr_{lack}$  holds, the following equanality holds.

■ **Algorithm 4** Procedure *DecideDirection()* ( $v_{i,j}$  is the current node of  $a_h$ ).

---

**Main Routine of Agent  $a_h$ :**

```

1  $a_h.dir := 0$ 
2 if  $v_{ij}.nAgentsCurrent = a_h.nAgentsRowRing$  then
3   // All the agents in the current row ring stay at the same node
4   Calculate  $a_h.rank$  among the agents at the same node
5   if  $1 \leq a_h.rank \leq \lfloor (v_{ij}.nAgentsCurrent - (2g + 1))/2 \rfloor$  then  $a_h.dir := -1$ 
6   else if
7      $\lfloor (v_{ij}.nAgentsCurrent - (2g + 1))/2 \rfloor + 1 \leq a_h.rank \leq v_{ij}.nAgentsCurrent - (2g + 1)$ 
8     then  $a_h.dir := 1$ 
9 else
10  // There exist two nodes where an agent exists in the current row ring
11  Calculate  $a_h.rank$  among the agents at the same node
12  if  $v_{ij}.nAgentsCurrent \geq \lceil a_h.nAgentsRowRing/2 \rceil$  then
13    if  $v_{ij}.nAgentsCurrent \geq 2g + 1$  then
14      if  $1 \leq a_h.rank \leq \lfloor (v_{ij}.nAgentsCurrent - (2g + 1))/2 \rfloor$  then  $a_h.dir := -1$ 
15      else if
16         $\lfloor (v_{ij}.nAgentsCurrent - (2g + 1))/2 \rfloor + 1 \leq a_h.rank \leq v_{ij}.nAgentsCurrent - (2g + 1)$ 
17        then  $a_h.dir := 1$ 
18      else
19        if  $1 \leq a_h.rank \leq \lfloor (a_h.nAgentsRowRing - (2g + 1))/2 \rfloor$  then  $a_h.dir := -1$ 
20        else if  $\lfloor (a_h.nAgentsRowRing - (2g + 1))/2 \rfloor + 1 \leq a_h.rank \leq$ 
21           $a_h.nAgentsRowRing - (2g + 1)$  then  $a_h.dir := 1$ 
22    else
23      if  $a_h.nAgentsRowRing - v_{ij}.nAgentsCurrent \geq 2g + 1$  then
24        if  $1 \leq a_h.rank \leq \lfloor (v_{ij}.nAgentsCurrent)/2 \rfloor$  then  $a_h.dir := -1$ 
25        else  $a_h.dir := 1$ 
26  return  $a_h.dir$ 

```

---

$$\begin{aligned}
SumDiff_{enough} &= 2gn + 2n - 1 - (nr_{lack} \times (2g + 1) - SumDiff_{lack}) - nr_{enough} \times (2g + 1) \\
&= 2gn + 2n - 1 - (nr_{lack} \times (2g + 1) - SumDiff_{lack}) - (n - nr_{lack}) \times (2g + 1) \\
&= SumDiff_{lack} + n - 1.
\end{aligned}$$

Then, by line 2 of Algorithm 3, since several agents in a row ring with at least  $2g + 3$  (resp., less than  $2g + 3$ ) agents try to (resp., do not try to) move vertically, the situation where the number of agents trying to move vertically from an enough row ring to a node in lacking row rings is the minimum, is that, among  $nr_{enough}$  enough row rings, there exist  $2g + 2$  agents in each of  $nr_{enough} - 1$  enough row rings, and there exist the remaining  $2g + 1 + SumDiff_{enough} - (nr_{enough} - 1)$  agents in the other enough row ring. Then, by Algorithm 3 and the fact of  $nr_{enough} \leq n$  and  $SumDiff_{enough} = SumDiff_{lack} + n - 1$ , the number of agents trying to move vertically is at least  $2g + 1 + SumDiff_{enough} - (nr_{enough} - 1) - (2g + 1) = SumDiff_{enough} - (nr_{enough} - 1) \geq SumDiff_{enough} - (n - 1) = SumDiff_{lack} + n - 1 - (n - 1) = SumDiff_{lack}$ . In addition, since at most one link is missing in each of column rings,  $\lfloor SumDiff_{lack}/2 \rfloor$  agents can visit a node in a lacking row ring  $R_{lack}^j$ . If  $\lfloor SumDiff_{lack}/2 \rfloor \leq diff_{lack}^j$ ,  $SumDiff_{lack}$  halves (property (i) holds). Otherwise,  $R_{lack}^j$  becomes an enough row ring from the next adjusting phase (property (ii) holds). Therefore, the claim follows.  $\triangleleft$

Thus, by Claim 6 and the fact that the value of  $SumDiff_{lack}$  is at most  $= (2g+1) \times (n-1) = O(gn)$ , after executing the adjusting phase  $n$  times in total,  $SumDiff_{lack}$  becomes 0, which means there exist at least  $2g+1$  agents in each of row rings. Therefore, the lemma follows. ◀

We have the following theorem for the proposed algorithm.

► **Theorem 7.** *When  $2gn + 2n - 1 \leq k \leq 2gn + 6n + 16g - 12$ , the proposed algorithm solves the  $g$ -partial gathering problem in dynamic tori with  $O(n^2)$  rounds and the total number of  $O(gn^3)$  moves.*

## 6 The case of $k \geq 2gn + 6n + 16g - 11$

In this section, when  $k \geq 2gn + 6n + 16g - 11$ , we propose an algorithm to solve the problem with  $O(n^2)$  rounds and the total number of  $O(gn^2)$  (i.e., optimal) moves. In the algorithm, agents aim to make a configuration such that there exist at least  $(n+8)g$  agents in some row ring  $R_i$ . Then, several agents that gathered in  $R_i$  are partitioned into several groups each having at least  $g$  agents and they try to visit all the nodes in the torus in total. During the movement, if some agent group starting from  $R_i$  visits a node with less than  $g$  agents, the less than  $g$  agents join the group's movement. Thus, after all nodes are visited, agents achieve  $g$ -partial gathering.

Concretely, the algorithm comprises the following three phases: the observing phase, the semi-gathering phase, and the achievement phase. In the observing phase, each agent moves horizontally in the current row ring  $R_i$  and recognizes the minimum agent ID among agents in  $R_i$  (the actual behavior is almost the same as that of the counting phase in Section 5.1). In the semi-gathering phase, several agents in row ring  $R_i$  move in the torus, observe the minimum ID written in each of row rings, and share the information with agents in  $R_i$ . Thereafter, each agent tries to move vertically to visit a node in the row ring  $R_{min}$  where there exists an agent with the minimum ID among all agents in the initial configuration. However, there may exist agents that cannot reach a node in  $R_{min}$  due to link-missings. Hence, in the achievement phase, agents in  $R_{min}$  visit all the nodes in the torus in total to achieve  $g$ -partial gathering.

### 6.1 Observing phase

The behavior of agents in this phase is almost the same as that of the counting phase in Section 5.1. Concretely, each agent  $a_h$  first writes its ID on the current node  $v_{i,j}$ 's whiteboard. Thereafter, for  $3n$  rounds,  $a_h$  tries to move right in its row ring  $R_i$  and stores the observed IDs during the movement. By this behavior, by the similar discussion of the proof for Lemma 4, (i) each agent  $a_h$  in row ring  $R_i$  travels once around the row ring, or (ii) all the agents in  $R_i$  stay at the same node. Then, in either case, each agent  $a_h$  can get the list of IDs for all agents in  $R_i$ . Among the IDs,  $a_h$  stores the minimum ID to variable  $a_h.minIDrow$ , and it selects the semi-gathering node  $v_{sGather}$  as the node where the minimum ID is written in the row ring (the information of the semi-gathering node is used in the next subsection).

The pseudocode in the observing phase is described in Algorithm 5. Variables newly used from this section are given in Table 3. Notice that several variables are used in the following phases. Concerning the observing phase, by the same proof idea as that in Lemma 4, we have the following lemma.

► **Lemma 8.** *After finishing the observing phase, each agent  $a_h$  in row ring  $R_i$  stores the minimum agent ID among agents existing in  $R_i$  in the initial configuration to variable  $a_h.minIDrow$ .*

■ **Table 3** Global variables newly used in Section 6.

**Variables for agent  $a_h$ .**

Type	Name	Meaning	Initial value
int	$a_h.minIDrow$	the minimum agent ID among agents existing in the same row ring as $a_h$ in $c_0$	$\perp$
int	$a_h.minIDall$	the minimum agent ID among all the agents in $A_{row7}$	$\perp$

**Variables for node  $v_{i,j}$ .**

Type	Name	Meaning	Initial value
int	$v_{ij.ID}$	ID stored at $v_{i,j}$	$\perp$
int	$v_{ij.minIDrow}$	the minimum agent ID among agents existing in the row ring $R_i$ in $c_0$	$\perp$
int	$v_{ij.minIDall}$	the minimum agent ID among all the agents in $A_{row7}$	$\perp$

■ **Algorithm 5** The behavior of agent  $a_h$  in the observing phase ( $v_{i,j}$  is the current node of  $a_h$ ).

---

**Main Routine of Agent  $a_h$ :**

- 1  $v_{ij.ID} := a_h.id, v_{ij.nAgentsCurrent}++$
- 2  $a_h.nVisited := 1, a_h.rounds := 1, a_h.IDs[a_h.nAgentsRowRing] := a_h.id$   
 $a_h.nAgentsRowRing++$
- 3 **while**  $a_h.rounds \leq 3n$  **do**
- 4      $v_{ij.nAgentsCurrent}--$
- 5     Try to move from the current node  $v_{i,j}$  to the right neighboring node  $v_{i,j+1}$
- 6     **if**  $a_h$  reached  $v_{i,j+1}$  (that becomes new  $v_{i,j}$ ) **then**
- 7          $a_h.nVisited++$
- 8         **if**  $(v_{ij.ID} \neq \perp) \wedge (a_h.nVisited \leq n)$  **then**
- 9              $a_h.IDs[a_h.nAgentsRowRing] := v_{ij.ID}$
- 10             $a_h.nAgentsRowRing++$
- 11      $v_{ij.nAgentsCurrent}++, a_h.rounds++$
- 12 **if**  $a_h.nVisited \geq n$  **then**
- 13     //  $a_h$  traveled once around the row ring
- 14     Let  $min$  be the minimum ID among  $a_h.IDs[]$
- 15      $a_h.minIDrow := min$
- 16     Select the semi-gathering node  $v_{sGather}$  as a node where  $min$  (or  $a_h.minIDrow$ ) is written
- 17 **if**  $a_h.nVisited < n$  **then**
- 18     // all the agents in row ring  $R_i$  stay at the current node
- 19     Let  $min$  be the ID of agent  $a_{h'}$  with  $a_{h'}.rank = 1$
- 20      $a_h.minIDrow := min$
- 21     Select the current node as the semi-gathering node  $v_{sGather}$
- 22 Terminate the counting phase and enter the semi-gathering phase

---

## 6.2 Semi-gathering phase

In this phase, agents try to move vertically to visit a node in the row ring  $R$  so that there exist at least  $(n + 8)g$  agents in  $R$ . To this end, the semi-gathering phase comprises the following two sub-phases: the recognizing sub-phase and the moving sub-phase. In the recognizing sub-phase, several agents in each row ring  $R_i$  move horizontally to write the value of the minimum agent ID among agents in  $R_i$ , move vertically to collect information of minimum IDs written in each row ring, and share the information of the row ring  $R_{min}$  where, in the initial configuration, there exists the agent with the minimum ID among all agents that try to move in the torus in this sub-phase, with other agents in  $R_i$ . In the moving sub-phase, each agent moves vertically to visit a node in  $R_{min}$ .

### 6.2.1 Recognizing sub-phase

By Lemma 8, at the beginning of the recognizing sub-phase, each agent  $a_h$  in row ring  $R_i$  knows the list of IDs of all agents in  $R_i$ . When the number of IDs is less than 7 (i.e., there exist less than 7 agents in  $R_i$  in the initial configuration), agents in  $R_i$  do nothing in this sub-phase (and the next moving sub-phase) and wait for other agents' instructions, which is described in Section 6.3. Hence, in the following, we describe the behavior of agents in a row ring in which there exist at least 7 agents in the initial configuration. Notice that there exists at least one such a row ring with at least 7 agents because we assume  $k \geq 2gn + 6n + 16g - 11$  in this section and thus it never happens that there exist at most 6 agents in each row ring. We denote  $R_{row7}$  by the set of such row rings and  $A_{row7}$  by the set of agents existing in  $R_{row7}$  at the beginning of the recognizing sub-phase. Let  $a_1^i$  (resp.,  $a_2^i$ ) be the agent with the smallest (resp., the second smallest) ID among agents in  $R_i$ . Then, similar to the first behavior of the adjusting phase in Section 5.2 (Algorithm 3),  $a_1^i$  (resp.,  $a_2^i$ ) tries to move right (resp., left) for  $n$  rounds and then tries to move left (resp., right) for  $n$  rounds. During the movement,  $a_1^i$  and  $a_2^i$  write the value of ID of  $a_1^i$  ( $= a_1^i.minIDrow$  or  $a_2^i.minIDrow$ ) to variable  $v_{ij}.minIDrow$  of each node  $v_{i,j}$ 's whiteboard. By this behavior, by the same proof idea of Lemma 5, the value of  $a_1^i$ 's ID is stored to each node's whiteboard in row ring  $R_i$ .

Thereafter, for  $3n$  rounds, each agent  $a_h$  tries to move right until visiting  $v_{sGather}$  (where  $a_1^i$  exists in the initial configuration). After the movement, by the similar discussion of the proof of Lemma 4, all the agents in  $R_i$  that do not reach  $v_{sGather}$  stay at the same node  $v_i'$ . Between  $v_{sGather}$  and  $v_i'$ , we call the node with more (resp., less) agents  $v_{more}^i$  (resp.,  $v_{less}^i$ ) (tie is broken using agent IDs), and let  $na_{more}^i$  be the number of agents at  $v_{more}^i$ . Then, since there exist at least 7 agents in  $R_i$ ,  $na_{more}^i \geq 4$  holds and the four agents at  $v_{more}^i$  try to move vertically to collect the information of minimum IDs written in each row ring. We use the procedure called *Splitting()*, introduced in [18]. Concretely, let  $a_1^{iMore}$  (resp.,  $a_2^{iMore}$ ,  $a_3^{iMore}$ , and  $a_4^{iMore}$ ) be the agent with the smallest (resp., the second, third, and fourth smallest) ID at  $v_{more}^i$ . We call  $a_1^{iMore}$  and  $a_2^{iMore}$  (resp.,  $a_3^{iMore}$  and  $a_4^{iMore}$ ) the *up group* (resp., the *down group*). Then, each agent  $a_h^{iMore}$  ( $1 \leq h \leq 4$ ) tries to move up or down for  $12n$  rounds. Concretely, for the first  $3n$  rounds,  $a_h^{iMore}$  tries to move up regardless of whether it belongs to the up group or the down group. Next, for the second  $3n$  rounds, agents in the up (resp., down) group try to move up (resp., down). Thereafter, for the third  $3n$  rounds, each agent  $a_h^{iMore}$  tries to move up. Finally, for the last (or fourth)  $3n$  rounds, agents in the up (resp., down) group try to move up (resp., down). During the movement,  $a_h^{iMore}$  memorizes the value of the minimum ID that has ever observed to variable  $a_h^{iMore}.minIDall$ . By this behavior, we can show by [18] that either the up group or the down group travels once around the current column ring, which means the group can know the value of the minimum agent ID among  $A_{row7}$ . Without loss of generality, we assume that the up group ( $a_1^{iMore}$  and  $a_2^{iMore}$ ) traveled once around the column ring. Then, after *Splitting()*, for  $n$  rounds,  $a_1^{iMore}$  and  $a_3^{iMore}$  (resp.,  $a_2^{iMore}$  and  $a_4^{iMore}$ ) try to move up (resp., down) to visit a from where they started these movements (i.e.,  $v_{more}^i$ ). Since at most one link is missing in each of column rings, either  $a_1^{iMore}$  or  $a_2^{iMore}$  can visit  $v_{more}^i$  and either  $a_3^{iMore}$  or  $a_4^{iMore}$  can also visit  $v_{more}^i$ . Hence, after the movement, there exist at least two agents at  $v_{more}^i$ , and they can know the minimum agent ID among  $A_{row7}$ . Among the two agents, let  $a_{1'}^{iMore}$  (resp.,  $a_{2'}^{iMore}$ ) be the agent with a smaller (resp., larger) ID. Then, for  $n$  rounds,  $a_{1'}^{iMore}$  (resp.,  $a_{2'}^{iMore}$ ) tries to move right (resp., left) and writes the value of  $a_h^{iMore}.minIDall$  ( $h = 1', 2'$ ) to the variable  $v_{ij}.minIDall$  for each node  $v_{i,j}$ 's whiteboard in  $R_i$ . Then, since  $a_{1'}^{iMore}$  and  $a_{2'}^{iMore}$  can visit all the  $n$  nodes in  $R_i$  in total, through  $v_{ij}.minIDall$ , each agent  $a_h$  in  $R_i$  can store the correct value of the minimum agent ID among  $A_{row7}$  to  $a_h.minIDall$ .

The pseudocode of the recognizing sub-phase is described in Algorithm 6. In Algorithm 6, agents use Procedure *Splitting()*, whose pseudocode is described in Algorithm 7. For simplicity, in *Splitting()*, we omit the description of from which node  $v_{sGather}$  or  $v'$  agents try to move vertically using IDs in the case that there exist the same number of agents at both  $v_{sGather}$  and  $v'$ . In addition, for simplicity, we omit the detailed description of whether the up group traveled once around the column ring or the down group did so.

Concerning the recognizing sub-phase, we have the following lemma.

► **Lemma 9.** *Let  $A_{row7}$  be the set of agents such that there exist at least 7 agents in their initially belonging row ring in the initial configuration, and let  $\mathcal{R}_{row7}$  be the set of row rings having agents in  $A_{row7}$  in the initial configuration. Then, after finishing the recognizing sub-phase, each agent in  $\mathcal{R}_{row7}$  recognizes the minimum agent ID among  $A_{row7}$ .*

### 6.2.2 Moving sub-phase

In this sub-phase, agents try to visit a node in the row ring  $R_{min}$  where there exists the agent with the minimum ID among  $A_{row7}$  in the initial configuration. First, for  $3n$  rounds, each agent  $a_h$  in row ring  $R_i$  tries to move right until visiting a node  $v_{sGather}^i$  where there exists an agent with the minimum ID among all the agents in the current row ring  $R_i$  in the initial configuration. Then, by a similar discussion of the proof of Lemma 4, agents that do not reach  $v_{sGather}^i$  stay at the same node  $v_i'$ . Next,  $a_h$  calculates its rank among agents at the same node. If its rank is at most (resp., more than) the half of the number of agents at the current node,  $a_h$  belongs to an up (resp., a down) group. Then, for  $n$  rounds, until visiting a node in  $R_{min}$ ,  $a_h$  tries to move up (resp., down) if it is in an up (resp., a down) group. Since at most one link is missing in each column ring, either the up group or down group can visit a node in  $R_{min}$  after the movement. By this behavior, we show in Lemma 10 that there exist at least  $(n + 8)g$  agents in  $R_{min}$  after finishing the moving sub-phase.

The pseudocode of the moving sub-phase is described in Algorithm 8. Notice that, in the previous recognizing sub-phase, agents that belonged to an up group or a down group may stay at a node not in  $\mathcal{R}_{row7}$  at the beginning of the moving sub-phase due to link-missings. In this case, such agents do nothing in this sub-phase and wait for other agents' instructions in the next phase (lines 2 and 3).

Concerning the moving sub-phase, we have the following lemma.

► **Lemma 10.** *After finishing the moving sub-phase, there exist at least  $(n + 8)g$  agents in some row ring.*

**Proof.** By the behavior for the moving sub-phase (Algorithm 8), all agents existing in  $\mathcal{R}_{row7}$  try to visit a node in  $R_{min}$  and at least the half of such agents can visit there. Then, the initial configuration such that the number of agents that try to visit a node in  $R_{min}$  is the minimum is that (1) there exist 7 agents in  $R_{min}$ , (2) there exist 6 agents in each of  $n - 2$  row rings, and (3) there exist the remaining agents in one row ring  $R_i$ . In this case, each agent in the  $n - 2$  row rings does not move in this sub-phase. In addition, it is possible that two agents that, existed in  $R_{min}$  in the initial configuration and belonged to an up or a down group in the previous recognizing sub-phase, do not stay at a node in  $R_{min}$  or  $R_i$  after the movement. The same thing holds for agents in  $R_i$ . Thus, at the beginning of the moving sub-phase, since there exist at least 5 agents in  $R_{min}$  and they do not move in this sub-phase, and since there exist at least  $k - 7 - 6(n - 2) - 2 = k - 6n + 3$  agents in  $R_i$ , the number of agents in  $R_{min}$  at the end of the moving sub-phase is at least



■ **Algorithm 6** The behavior of agent  $a_h$  in the recognizing sub-phase ( $v_{i,j}$  is the current node of  $a_h$ ).

---

**Main Routine of Agent  $a_h$ :**

```

1  $a_h.rounds := 1, a_h.dir := 0$ 
2 if  $a_h.nAgentsRowRing < 7$  then
3   while  $a_h.rounds < 23n$  do  $a_h.rounds++$ 
4   Terminate the semi-gathering phase and enter the achievement phase
5 else
6   if  $a_h.id$  is smallest among  $a_h.IDs[]$  then  $a_h.dir := 1$ 
7   if  $a_h.id$  is the second smallest among  $a_h.IDs[]$  then  $a_h.dir := -1$ 
8   while  $a_h.rounds \leq n$  do
9      $v_{ij}.nAgentsCurrent--$ 
10    Try to move from the current node  $v_{i,j}$  to the neighboring node  $v_{i,j+a_h.dir}$ 
11    if  $a_h$  reached  $v_{i,j+a_h.dir}$  (that becomes new  $v_{i,j}$ ) then  $v_{ij}.minIDrow := a_h.minIDrow$ 
12     $v_{ij}.nAgentsCurrent++, a_h.rounds++$ 
13   $a_h.dir := a_h.dir \times (-1)$ 
14  while  $a_h.rounds \leq 2n$  do
15     $v_{ij}.nAgentsCurrent--$ 
16    Try to move from the current node  $v_{i,j}$  to the neighboring node  $v_{i,j+a_h.dir}$ 
17    if  $a_h$  reached  $v_{i,j+a_h.dir}$  (that becomes new  $v_{i,j}$ ) then  $v_{ij}.minIDrow := a_h.minIDrow$ 
18     $v_{ij}.nAgentsCurrent++, a_h.rounds++$ 
19  while  $a_h.rounds \leq 5n$  do
20    if  $v_{ij}.ID \neq v_{ij}.minIDrow$  then
21       $v_{ij}.nAgentsCurrent--$ 
22      Try to move from the current node  $v_{i,j}$  to the right neighboring node  $v_{i,j+1}$ 
23       $v_{ij}.nAgentsCurrent++$ 
24       $a_h.rounds++$ 
25   $a_h.minIDall := a_h.minIDrow$ 
26  Splitting()
27   $a_h.rounds := 1$ 
28  while  $a_h.rounds \leq n$  do
29    if  $v_{ij}.minIDrow \neq a_h.minIDrow$  then
30       $v_{ij}.nAgentsCurrent--$ 
31      Try to move from the current node  $v_{i,j}$  to the neighboring node  $v_{i+a_h.dir,j}$ 
32       $v_{ij}.nAgentsCurrent++$ 
33       $a_h.rounds++$ 
34  if  $v_{ij}.minIDrow = a_h.minIDrow$  then
35    if  $a_h$  visited all the nodes in the current column ring during Splitting() then
36       $v_{ij}.minIDall := a_h.minIDall$ 
37       $a_h.minIDall := v_{ij}.minIDall$ 
38      if  $a_h.rank = 1$  then  $a_h.dir := 1$ 
39      else if  $a_h.rank = 2$  then  $a_h.dir := -1$ 
40      while  $a_h.rounds \leq 2n$  do
41         $v_{ij}.nAgentsCurrent--$ 
42        Try to move from the current node  $v_{i,j}$  to the neighboring node  $v_{i,j+a_h.dir}$ 
43         $v_{ij}.minIDall := a_h.minIDall, v_{ij}.nAgentsCurrent++, a_h.rounds++$ 
44  Terminate the recognizing sub-phase and enter the moving sub-phase

```

---

■ **Algorithm 7** Procedure *Splitting()* ( $v_{i,j}$  is the current node of  $a_h$ .)

---

**Main Routine of Agent  $a_h$ :**

```

1  $a_h.rounds := 1, a_h.dir := 0$ 
2 if ( $v_{ij}.nAgentsCurrent \geq \lceil a_h.nAgentsRowRing/2 \rceil$ )  $\wedge$  ( $v_{ij}.nAgentsCurrent \geq 4$ ) then
3   if  $1 \leq a_h.rank \leq 2$  then  $a_h.dir := -1$ 
4   else if  $3 \leq a_h.rank \leq 4$  then  $a_h.dir := 1$ 
5 while  $a_h.rounds \leq 3n$  do
6    $v_{ij}.nAgentsCurrent--$ 
7   Try to move from the current node  $v_{i,j}$  to the up neighboring node  $v_{i-1,j}$ 
8    $v_{ij}.nAgentsCurrent++$ 
9   If  $v_{ij}.minIDrow < a_h.minIDall$  then  $a_h.minIDall := v_{ij}.minIDrow$ 
10   $a_h.rounds++$ 
11 while  $a_h.rounds \leq 6n$  do
12   $v_{ij}.nAgentsCurrent--$ 
13  Try to move from the current node  $v_{i,j}$  to the neighboring node  $v_{i+a_h.dir,j}$ 
14   $v_{ij}.nAgentsCurrent++$ 
15  If  $v_{ij}.minIDrow < a_h.minIDall$  then  $a_h.minIDall := v_{ij}.minIDrow$ 
16   $a_h.rounds++$ 
17 while  $a_h.rounds \leq 9n$  do
18   $v_{ij}.nAgentsCurrent--$ 
19  Try to move from the current node  $v_{i,j}$  to the up neighboring node  $v_{i-1,j}$ 
20   $v_{ij}.nAgentsCurrent++$ 
21  If  $v_{ij}.minIDrow < a_h.minIDall$  then  $a_h.minIDall := v_{ij}.minIDrow$ 
22   $a_h.rounds++$ 
23 while  $a_h.rounds \leq 12n$  do
24   $v_{ij}.nAgentsCurrent--$ 
25  Try to move from the current node  $v_{i,j}$  to the neighboring node  $v_{i+a_h.dir,j}$ 
26   $v_{ij}.nAgentsCurrent++$ 
27  If  $v_{ij}.minIDrow < a_h.minIDall$  then  $a_h.minIDall := v_{ij}.minIDrow$ 
28   $a_h.rounds++$ 
29 if ( $a_h.rank = 1$ )  $\vee$  ( $a_h.rank = 3$ ) then  $a_h.dir := -1$ 
30 else  $a_h.dir := 1$ 

```

---

$$\begin{aligned}
(7-2) + \lfloor (k-6n+3)/2 \rfloor &\geq 5 + (k-6n+3)/2 - 1 \\
&\geq 4 + (2gn+16g-8)/2 \\
&= 4 + gn + 8g - 4 \\
&= (n+8)g
\end{aligned}$$

The second inequality comes from the assumption of  $k \geq 2gn + 6n + 16g - 11$ . Therefore, the lemma follows. ◀

### 6.3 Achievement phase

In this phase, agents in  $R_{min}$  move in the torus to achieve  $g$ -partial gathering. Intuitively, from  $R_{min}$ , some  $2g$  agents visit a node in a row ring. Thereafter, the  $2g$  agents visit all the nodes in the row ring in a way such that an agent group  $A_r$  with some  $g$  agents tries to move right and another agent group  $A_l$  with the other  $g$  agents tries to move left. In addition, during the movement, when  $A_r$  or  $A_l$  visits a node with less than  $g$  agents, the less than  $g$  agents join the group's movement. By executing such a behavior in each of the  $n$  row rings, agents can achieve  $g$ -partial gathering.

■ **Algorithm 8** The behavior of agent  $a_h$  in the moving sub-phase ( $v_{i,j}$  is the current node of  $a_h$ .)

---

**Main Routine of Agent  $a_h$ :**

```

1  $a_h.rounds := 1, a_h.dir := 0$ 
2 if  $v_{ij}.minIDall = \perp$  then
3   while  $a_h.rounds \leq 4n$  do  $a_h.rounds++$ 
4 else
5   while  $a_h.rounds \leq 3n$  do
6     if  $v_{ij}.ID \neq v_{ij}.minIDrow$  then
7        $v_{ij}.nAgentsCurrent--$ 
8       Try to move from the current node  $v_{i,j}$  to the right neighboring node  $v_{i,j+1}$ 
9        $v_{ij}.nAgentsCurrent++$ ,
10     $a_h.rounds++$ 
11  if  $a_h.rank \leq \lceil v_{ij}.nAgentsCurrent/2 \rceil$  then  $a_h.dir := -1$ 
12  else  $a_h.dir := 1$ 
13  while  $a_h.rounds \leq 4n$  do
14    if  $v_{ij}.minIDrow \neq a_h.minIDall$  then
15       $v_{ij}.nAgentsCurrent--$ 
16      Try to move from the current node  $v_{i,j}$  to the neighboring node  $v_{i+a_h.dir,j}$ 
17       $v_{ij}.nAgentsCurrent++$ ,
18     $a_h.rounds++$ 
19 Terminate the moving sub-phase and enter the achievement phase

```

---

To this end, each agent  $a_h$  in  $R_{min}$  first executes procedure *Counting()* in Section 5.1 for  $3n$  rounds. Then,  $a_h$  can count the number of agents currently existing in  $R_{min}$ . Thereafter, for  $3n$  rounds,  $a_h$  tries to move right until visiting the node  $v_{i,j}^{min}$  with  $v_{ij}^{min}.ID = a_h.minIDall$ . Then, similarly to the previous discussion, all the agents in  $R_{min}$  that do not reach  $v_{i,j}^{min}$  stay at the same node  $v'_{min}$ . Between the two nodes, we call the node with more (resp., less) agents  $v_{min}^{more}$  (resp.,  $v_{min}^{less}$ ) (tie is broken using agent IDs). In addition, let  $R_{fromMin}^\ell$  ( $0 \leq \ell \leq n-1$ ) be the  $\ell$ -th up row ring from  $R_{min}$ . Here, we say a row ring  $R_i$  is  $\ell$ -th up from row ring  $R_{i'}$  when  $i' - i = \ell$  holds. Notice that  $R_{fromMin}^0$  is  $R_{min}$  itself. Then, agents execute the following sub-phases  $n$  times: in the  $\ell$ -th sub-phase,  $4g$  agents from  $v_{min}^{more}$  or  $v_{min}^{less}$  try to move vertically so that at least  $2g$  agents visit a node in  $R_{nomMin}^\ell$ , and then at least  $2g$  agents try to move horizontally to visit all the nodes in  $R_{fromMin}^\ell$ .

First, as a special case, we explain the behaviors of the 0-th sub-phase (i.e., movements in row ring  $R_{fromMin}^0$  or  $R_{min}$ ). Let  $na_{min}^{more}$  (resp.,  $na_{min}^{less}$ ) be the number of agents staying at  $v_{min}^{more}$ . When  $na_{min}^{less} \geq g$ , agents in  $R_{min}$  do nothing for  $2n$  rounds. Otherwise, several agents at  $v_{min}^{more}$  move horizontally to visit  $v_{min}^{less}$ . Concretely, each agent  $a_h$  at  $v_{min}^{more}$  with  $1 \leq a_h.rank \leq g$  (resp.,  $g+1 \leq a_h.rank \leq 2g$ ) belongs to a *right-left group*  $A_{rl}$  (resp., *left-right group*  $A_{lr}$ ). Then, for  $n$  rounds, each agent in  $A_{rl}$  (resp.,  $A_{lr}$ ) tries to move right (resp., left). By this behavior,  $A_{rl}$  and  $A_{lr}$  can visit all the  $n$  nodes in  $R_{min}$  in total. During the movement, if  $A_{rl}$  or  $A_{lr}$  visits  $v_{min}^{less}$ , the agents that stayed at  $v_{min}^{less}$  join the group's movement. Thereafter, for  $n$  rounds, each agent in  $A_{rl}$  (resp.,  $A_{lr}$ ) tries to move left (resp., right) until visiting  $v_{min}^{more}$ . By this behavior,  $A_{rl}$  or  $A_{lr}$  can return to  $v_{min}^{more}$ , there exist at most two nodes  $v_{min}^{more}$  and  $v_{min}^{less}$  with an agent, and both  $v_{min}^{more}$  and  $v_{min}^{less}$  have at least  $g$  agents. Notice that the position of  $v_{min}^{less}$  may change by these movements, but it does not affect the following explanations. In the following, we explain the behavior of each  $i$ -th sub-phase ( $i \geq 1$ ) with the updated numbers of  $na_{min}^{more}$  and  $na_{min}^{less}$ .

In the 1-st sub-phase, for  $n$  rounds, each agent  $a_h$  at  $v_{min}^{more}$  with  $1 \leq a_h.rank \leq 2g$  (resp.,  $2g + 1 \leq a_h.rank \leq 4g$ ) belongs to an up (resp., a down) group and tries to move up (resp., down) until visiting a node  $v_{fromMin}^1$  in  $R_{fromMin}^1$ . Since at most one link is missing in each column ring at each round, the up group or the down group can reach  $v_{fromMin}^1$ . Without loss of generality, we assume that the up group reached there. Thereafter, among agents in the up group, each agent  $a_h$  with  $1 \leq a_h.rank \leq g$  (resp.,  $g + 1 \leq a_h.rank \leq 2g$ ) belongs to a right-left group  $A_{rl}$  (resp., left-right group  $A_{lr}$ ). Then, for  $n$  rounds, each agent in  $A_{rl}$  (resp.,  $A_{lr}$ ) tries to move right (resp., left). By this behavior,  $A_{rl}$  and  $A_{lr}$  can visit all the  $n$  nodes in  $R_{fromMin}^1$  in total. During the movement, when  $A_{rl}$  or  $A_{lr}$  visits a node with less than  $g$  agents, the less than  $g$  agents join the group's movement. However, if the number  $na_{new}$  of agents in the updated group is at least  $2g$ , using IDs, only  $g$  agents continue their movements and the remaining  $na_{new} - g$  ( $\geq g$ ) agents stay at the current node to reduce the total number of agent moves. Thereafter, for  $n$  rounds, each agent in  $A_{rl}$  (resp.,  $A_{lr}$ ) tries to move left (resp., right) until returning to  $v_{fromMin}^1$ . By this behavior,  $A_{rl}$  or  $A_{lr}$  can reach  $v_{fromMin}^1$ .

Next, at the beginning of the 2-nd sub-phase, letting  $C_{min}^{more}$  be the column ring including  $v_{min}^{more}$ , there exist at least  $g$  agents at  $v_{fromMin}^1$  (i.e.,  $A_{rl}$  or  $A_{lr}$ ),  $2g$  agents at some node in  $C_{min}^{more}$  (i.e., the down group that may not have reached  $v_{fromMin}^1$  in the 1-st sub-phase), and  $na_{min}^{more} - 4g$  agents at  $v_{min}^{more}$ . From this situation, for  $n$  rounds, until visiting a node  $v_{fromMin}^2$  in  $R_{fromMin}^2$ , the  $2g$  agents that may not have reached  $v_{fromMin}^1$  in the 1-st sub-phase (the down group in this explanation) try to move down, the  $g$  agents at  $v_{fromMin}^1$  try to move up, and the  $g$  agents at  $v_{min}^{more}$  whose ID is either of the 1-st, 2nd,  $\dots$ , or  $g$ -th smallest newly try to move up. By this behavior, at least  $2g$  agents can reach  $v_{fromMin}^2$  by the similar discussion of the 1-st sub-phase. Thereafter, the  $2g$  agents try to move right or left and visit all the  $n$  nodes in  $R_{fromMin}^2$ , and some less than  $g$  agents at a node in  $R_{fromMin}^2$  join a group's movement. Agents repeat such sub-phases until the number of agents at  $v_{min}^{more}$  becomes less than  $2g$  at the end of some sub-phase  $\ell'$ .

In the  $(\ell' + 1)$ -th sub-phase, agents at  $v_{min}^{less}$  execute the exact same behavior as that of agents at  $v_{min}^{more}$  in the 1-st sub-phase. Thereafter, agents that existed at  $v_{min}^{less}$  complete the remaining sub-phases (i.e., until agents that existed in  $R_{min}$  execute the sub-phases  $n$  times in total). Then, agents achieve  $g$ -partial gathering. Notice that it is possible no agent at  $v_{min}^{less}$  moves in these sub-phases when there exist a large number of agents at  $v_{min}^{more}$  at the beginning of the 1-st sub-phase. Agents at  $v_{min}^{less}$  can determine whether or not they need to move in the torus and when they should start moving if they need to move, by using the information of  $na_{min}^{less}$  and the total number of agents existing in  $R_{min}$  calculated at the beginning of this achievement phase.

Concerning the achievement phase, we have the following lemma.

► **Lemma 11.** *After finishing the achievement phase, agents solve the  $g$ -partial gathering problem.*

**Proof.** First, in the 0-th sub-phase, when  $na_{min}^{less} < g$ , by the behavior for the achievement phase,  $2g$  agents at  $v_{min}^{more}$  are partitioned into a right-left group  $A_{rl}$  with at least  $g$  agents and a left-right group  $A_{lr}$  with at least  $g$  agents, and  $A_{rl}$  (resp.,  $A_{lr}$ ) tries to move right (resp., left) for  $n$  rounds and then tries to move left (resp., right) for  $n$  rounds until returning to  $v_{min}^{more}$ . In addition, during the movement, when  $A_{rl}$  or  $A_{lr}$  visits node  $v_{min}^{less}$  with less than  $g$  agents, the less than  $g$  agents join the group's movement. Hence, by this behavior, at least  $g$  agents or no agent exists at each node in  $R_{min}$ . Thereafter, in the  $\ell$ -th sub-phase ( $1 \leq \ell \leq n - 1$ ), since at least  $2g$  agents try to move up and another at least  $2g$  agents try to move down to visit the  $v_\ell$  in the  $\ell$ -th up ring  $R_{fromMin}^\ell$  from  $R_{min}$ , at least  $2g$  agents can

reach  $v_\ell$ . Then, by the same discussion in the case of the 0-th sub-phase, a right-left group  $A_{r_l}$  and a left-right group  $A_{l_r}$  visit all the nodes in  $R_{fromMin}^\ell$  in total, at least  $g$  agents or no agent exists at each node in  $R_{fromMin}^\ell$ , and  $A_{r_l}$  or  $A_{l_r}$  returns to  $v_\ell$ . Hence, by executing such a behavior in each row ring one by one, agents can achieve  $g$ -partial gathering.

In the following, we show that there exist the sufficient number of agents in  $R_{min}$  to solve the problem at the beginning of the achievement phase. To execute the above behaviors in  $r$  consecutive row rings, at least  $(r + 3)g$  agents are required ( $4g$  agents are required when agents from  $v_{min}^{more}$  or  $v_{min}^{less}$  start their vertical movements for the first time, and additional  $g$  agents are required otherwise). In addition, since agents start the above behavior from at most two nodes  $v_{min}^{more}$  and  $v_{min}^{less}$ , the situation where the required number of agents staying in  $R_{min}$  (i.e., agents staying at  $v_{min}^{more}$  or  $v_{min}^{less}$ ) is the maximum when agents starting from  $v_{min}^{more}$  visit  $r'$  ( $\lceil n/2 \rceil \leq r' \leq n - 1$ ) consecutive row rings in total is such that there exist  $(r' + 3)g + (g - 1)$  agents at  $v_{min}^{more}$  and there exist  $(n - r' + 3)g + (g - 1)$  agents at  $v_{min}^{less}$ . Thus,  $(r' + 3)g + (g - 1) + (n - r' + 3)g + (g - 1) < (n + 8)g$  agents are required in  $R_{min}$  at the beginning of the achievement phase to solve the problem. In Lemma 10, we already showed that such agents exist.

Therefore, the lemma follows.  $\blacktriangleleft$

We have the following theorem for the proposed algorithm.

► **Theorem 12.** *When  $k = O(gn)$  and  $k \geq 2gn + 6n + gn - 11$ , the proposed algorithm solves the  $g$ -partial gathering problem with  $O(n^2)$  rounds and the total number of  $O(gn^2)$  moves.*

## 7 Conclusion

In this paper, we considered the  $g$ -partial gathering problem of mobile agents in  $n \times n$  dynamic tori and considered the solvability of the problem and the time and move complexity, focusing on the relationship between values of  $k$ ,  $n$ , and  $g$ . First, we showed that agents cannot solve the problem when  $k = o(gn)$ , and showed that agents require a total number of  $\Omega(gn^2)$  moves to solve the problem when  $k = \Theta(gn)$ . Second, we showed that the problem can be solved with  $O(n^2)$  rounds and the total number of  $O(gn^3)$  moves when  $2gn + 2n - 1 \leq k \leq 2gn + 6n + 16g - 12$ . Finally, we showed that the problem can be solved with  $O(n^2)$  rounds and the total number of  $O(gn^2)$  moves when  $k = O(gn)$  and  $k \geq 2gn + 6n + 16g - 11$ . From these results, our algorithms can solve the partial gathering problem in dynamic tori with the asymptotically optimal number  $\Theta(gn)$  of agents and the second algorithm is also asymptotically optimal in terms of the total number of agent moves.

Future works are as follows. First, we consider the lower bound on the time complexity to solve the problem. Next, we consider whether or not agents can solve the problem with the asymptotically optimal total number of agent moves when  $k = \Omega(gn)$  but it is not in the range considered in Section 6. Finally, we will consider the solvability of the problem for agents with weaker capabilities. In this paper, as a first step to propose algorithms for solving the problem, we assumed that agents have distinct IDs, knowledge of  $k$  and  $n$ , common sense of direction, and they behave fully synchronously. In addition, we assumed that each node has a whiteboard. However, at this stage, we do not know whether or not agents with weaker capability can also solve the problem (e.g., agents without distinct IDs, without the common sense of direction, or agents that behave semi-synchronously or asynchronously). Hence, we plan to consider algorithms using agents with such weak capabilities. We conjecture that, in any of the above cases, agents cannot solve the problem or require more total number of moves than the proposed algorithms.

---

**References**

---

- 1 D. Baba, T. Izumi, F. Ooshita, H. Kakugawa, and T. Masuzawa. Linear time and space gathering of anonymous mobile agents in asynchronous trees. *Theoretical Computer Science*, 478:118–126, 2013.
- 2 S. Das, Di GA. Luna, D. Mazzei, and G. Prencipe. Compacting oblivious agents on dynamic rings. *PeerJ Computer Science*, 7:1–29, 2021.
- 3 Y. Dieudonné and A. Pelc. Anonymous meeting in networks. *Algorithmica*, 74(2):908–946, 2016.
- 4 Y. Dieudonné, A. Pelc, and V. Villain. How to meet asynchronously at polynomial cost. *SIAM Journal on Computing*, 44(3):844–867, 2015.
- 5 P. Fraigniaud and A. Pelc. Deterministic rendezvous in trees with little memory. *DISC*, pages 242–256, 2008.
- 6 T. Gotoh, Y. Sudo, F. Ooshita, H. Kakugawa, and T. Masuzawa. Group exploration of dynamic tori. *ICDCS*, pages 775–785, 2018.
- 7 R. S. Gray, D. Kotz, G. Cybenko, and D. Rus. D’agents: Applications and performance of a mobile-agent system. *Softw., Pract. Exper.*, 32(6):543–573, 2002.
- 8 E. Kranakis, D. Krizanc, and E. Markou. Mobile agent rendezvous in a synchronous torus. *LATIN*, pages 653–664, 2006.
- 9 E. Kranakis, D. Krozanc, and E. Markou. The Mobile Agent Rendezvous Problem in the Ring. *Synthesis Lectures on Distributed Computing Theory*, Vol. 1, 2010.
- 10 D.B. Lange and M. Oshima. Seven good reasons for mobile agents. *CACM*, 42(3):88–89, 1999.
- 11 Di GA. Luna, S. Dobrev, P. Flocchini, and N. Santoro. Distributed exploration of dynamic rings. *Distributed Computing*, 33(1):41–67, 2020.
- 12 Di GA. Luna, P. Flocchini, L. Pagli, G. Prencipe, N. Santoro, and G. Viglietta. Gathering in dynamic rings. *Theoretical Computer Science*, 811:79–98, 2018.
- 13 F. Ooshita, S. Kawai, H. Kakugawa, and T. Masuzawa. Randomized gathering of mobile agents in anonymous unidirectional ring networks. *IEEE Transactions on Parallel and Distributed Systems*, 25(5):1289–1296, 2014.
- 14 M. Shibata, S. Kawai, F. Ooshita, H. Kakugawa, and T. Masuzawa. Partial gathering of mobile agents in asynchronous unidirectional rings. *Theoretical Computer Science*, 617:1–11, 2016.
- 15 M. Shibata, N. Kawata, Y. Sudo, F. Ooshita, H. Kakugawa, and T. Masuzawa. Move-optimal partial gathering of mobile agents without identifiers or global knowledge in asynchronous unidirectional rings. *Theoretical Computer Science*, 822:92–109, 2020.
- 16 M. Shibata, D. Nakamura, F. Ooshita, H. Kakugawa, and T. Masuzawa. Partial gathering of mobile agents in arbitrary networks. *IEICE Transactions on Information and Systems*, 102(3):444–453, 2019.
- 17 M. Shibata, F. Ooshita, H. Kakugawa, and T. Masuzawa. Move-optimal partial gathering of mobile agents in asynchronous trees. *Theoretical Computer Science*, 705:9–30, 2018.
- 18 M. Shibata, Y. Sudo, J. Nakamura, and Y. Kim. Uniform deployment of mobile agents in dynamic rings. *SSS*, pages 248–263, 2020.
- 19 M. Shibata, Y. Sudo, J. Nakamura, and Y. Kim. Partial gathering of mobile agents in dynamic rings. *arXiv preprint*, 2022. [arXiv:2212.03457](https://arxiv.org/abs/2212.03457).
- 20 M. Shibata and S. Tixeuil. Partial gathering of mobile robots from multiplicity-allowed configurations in rings. *SSS*, pages 264–279, 2020.