

# Exploring a Dynamic Ring without Landmark

Archak Das

Department of Mathematics, Jadavpur University, India

*archakdas.math.rs@jadavpuruniversity.in*

Kaustav Bose

Department of Mathematics, Jadavpur University, India

*kaustavbose27@gmail.com*

Buddhadeb Sau

Department of Mathematics, Jadavpur University, India

*buddhadeb.sau@jadavpuruniversity.in*

---

## Abstract

Consider a group of autonomous mobile computational entities, called agents, arbitrarily placed at some nodes of a dynamic but always connected ring. The agents neither have any knowledge about the size of the ring nor have a common notion of orientation. We consider the EXPLORATION problem where the agents have to collaboratively to explore the graph and terminate, with the requirement that each node has to be visited by at least one agent. It has been shown by Di Luna et al. [Distrib. Comput. 2020] that the problem is solvable by two anonymous agents if there is a single observably different node in the ring called landmark node. The problem is unsolvable by any number of anonymous agents in absence of a landmark node. We consider the problem with non-anonymous agents (agents with distinct identifiers) in a ring with no landmark node. The assumption of agents with distinct identifiers is strictly weaker than having a landmark node as the problem is unsolvable by two agents with distinct identifiers in absence of a landmark node. This setting has been recently studied by Mandal et al. [ALGOSENSORS 2020]. There it is shown that the problem is solvable in this setting by three agents assuming that they have edge crossing detection capability. Edge crossing detection capability is a strong assumption which enables two agents moving in opposite directions through an edge in the same round to detect each other and also exchange information. In this paper we give an algorithm that solves the problem with three agents without the edge crossing detection capability.

## 1 Introduction

Consider a team of autonomous computational entities, usually called agents or robots, located at the nodes of a graph. The agents are able to move from a node to any neighboring node. The EXPLORATION problem asks for a distributed algorithm that allows the agents to explore the graph, with the requirement that each node has to be visited by at least one agent.

Being one of the fundamental problems in the field of autonomous multi-agent systems, the problem has been extensively studied in the literature. However, the majority of existing literature studies the problem for *static* graphs, i.e., the topology of the graph does not change over time. Recently within the distributed computing community, there has been a surge of interest in highly dynamic graphs: the topology of the graph changes continuously and unpredictably. In highly dynamic graphs, the topological changes are not seen as occasional anomalies (e.g., link failures, congestion, etc) but rather integral part of the nature of the system [19, 26]. We refer the readers to [4] for a compendium of different models of dynamic networks considered in the literature. If time is discrete, i.e., changes occur in rounds, then the evolution of a dynamic graph can be seen as a sequence of static graphs. A popular assumption in this context is *always connected* (Class 9 of [4]), i.e., the graph is connected in each round.

In the dynamic setting, the EXPLORATION problem was first studied in [20]. In particular, the authors studied the EXPLORATION problem in a dynamic but always connected ring by a set of autonomous agents. They showed that EXPLORATION is solvable by two anonymous agents (agents do not have unique identifiers) under fully synchronous setting (i.e., all agents are active in each round) if there is a single observably different node in the ring called *landmark* node. They also proved that in absence of a landmark node, two agents cannot solve EXPLORATION even if the agents are non-anonymous and they have chirality, i.e., they agree on clockwise and counterclockwise orientation of the ring. The impossibility result holds even if we relax the problem to EXPLORATION with partial termination. As opposed to the standard explicit termination setting where all agents are required to terminate, in the partial termination setting at least one agent is required to detect exploration and terminate. If the agents are anonymous, then EXPLORATION with partial termination with chirality remains unsolvable in absence of a landmark node even with arbitrary number of agents. Then in [22], the authors considered the EXPLORATION problem (without chirality and requiring explicit termination) with no landmark node. Since the problem cannot be solved even with arbitrary number of anonymous agents, they considered non-anonymous agents, in particular, agents with unique identifiers. Since the problem is unsolvable by two non-anonymous agents, they considered the question that whether the problem can be solved by three non-anonymous agents. They showed that the answer is yes if the agents are endowed with edge crossing detection capability. Edge crossing detection capability is a strong assumption which enables two agents moving in opposite directions through an edge in the same round to detect each other and also exchange information. In collaborative tasks like exploration, the agents are often required to meet at a node and exchange information. However, the edge crossing detection capability allows two agents to exchange information even without meeting at a node. The assumption is particularly helpful when the agents do not have chirality where it is more difficult to ensure meeting. Even if we do not allow exchange of information, simple detection of the swap can be useful in deducing important information about the progress of an algorithm. In [22], it was also shown that the assumption of edge crossing detection can be removed with the help of randomness. In particular, without assuming edge crossing detection capability, they gave a randomized algorithm that solves EXPLORATION with explicit termination with probability at least  $1 - \frac{1}{n}$  where  $n$  is the size of

the ring. Therefore this leaves the open question that whether the problem can be solved by a deterministic algorithm by three non-anonymous agents without edge crossing detection capability. In this paper, we answer this question affirmatively.

## 1.1 Related Work

The problem of EXPLORATION by mobile agents in static anonymous graph has been studied extensively in the literature [2, 5, 7, 8, 10, 12, 25]. Prior to [20], there have been a few works on EXPLORATION of dynamic graphs, but under assumptions such as complete a priori knowledge of location and timing of topological changes (i.e., *offline* setting) [9, 15, 17, 23] or periodic edges (edges appear periodically) [11, 16] or  $\delta$ -recurrent edges (each edge appears at least once every  $\delta$  rounds) [17] etc. In the *online* or *live* setting where the location and timing of the changes are unknown, distributed EXPLORATION of graphs without any assumption other than being always connected was first considered in [20]. In particular, they considered the problem on an always connected dynamic ring. They proved that without any knowledge of the size of the ring and without landmark node, EXPLORATION with partial termination is impossible by two agents even if the agents are non-anonymous and have chirality. They also proved that if the agents are anonymous, have no knowledge of size, and there is no landmark node then EXPLORATION with partial termination is impossible by any number of agents even in the presence of chirality. On the positive side the authors showed that under fully synchronous setting, if an upper bound  $N$  on the size of the ring is known to two anonymous agents, then EXPLORATION with explicit termination is possible within  $3N - 6$  rounds. They then showed that for two anonymous agents, if chirality and a landmark node is present, then exploration with explicit termination is possible within  $O(n)$  round, and in the absence of chirality with all other conditions remaining the same, EXPLORATION with explicit termination is possible within  $O(n \log n)$  rounds, where  $n$  is the size of the ring. They have also proved a number of results in the semi-synchronous setting (i.e., not all agents may be active in each round) under different assumptions. Then in [22], the authors considered agents with unique identifiers and edge crossing detection capability in a ring without any landmark node. They showed that EXPLORATION with explicit termination is impossible in the absence of landmark node and the knowledge of  $n$  by two agents with access to randomness, even in the presence of chirality, unique identifiers and edge-crossing detection capability. In the absence of randomness even EXPLORATION with partial termination is impossible in the same setting. With three agents under fully synchronous setting, the authors showed that EXPLORATION with explicit termination is possible by three non-anonymous agents with edge-crossing detection capability in absence of any landmark node. Removing the assumption of edge-crossing detection and replacing it with access to randomness, the authors gave a randomized algorithm for EXPLORATION with explicit termination with success probability at least  $1 - \frac{1}{n}$ . EXPLORATION of an always connected dynamic torus was considered in [14]. In [13] the problem of PERPETUAL EXPLORATION (i.e., every node is to be visited infinitely often) was studied in temporally connected (i.e., may not be always connected but connected over time) graphs. Other problems studied in dynamic graphs include GATHERING [3, 21, 24], DISPERSION [1, 18], PATROLLING [6] etc.

## 1.2 Our Results

We consider a dynamic but always connected ring of size  $n$ . A team of three agents are operating in the ring under a fully synchronous scheduler. Each agent has a  $k$  bit unique identifier. The agents do not have any knowledge of  $n$  and they do not have chirality. Furthermore, they do not have edge crossing detection capability. In this setting, we give a deterministic algorithm for EXPLORATION with explicit termination. The algorithm solves the problem in  $O(k^2 2^k n)$  rounds. As a subroutine, we also solve the problem MEETING where any two agents in the team are required to meet each other at a node. Another basic ingredient of our approach is an algorithm for the CONTIGUOUS AGREEMENT problem which requires that the agents have to agree on some common direction for some number of consecutive rounds. These problems may be of independent interest and useful for solving other problems in similar settings. A comparison of the results obtained in this paper with previous works is given in Table 1.

Paper	Number of agents	Agents	Landmark node	Edge cross. detection	Algorithm
[20]	2	Anonymous	Yes	No	Deterministic
[22]	3	Have unique identifiers	No	Yes	Deterministic
[22]	3	Have unique identifiers	No	No	Randomized
This paper	3	Have unique identifiers	No	No	Deterministic

Table 1: Comparison of our results with previous works.

## 1.3 Outline of the Paper

In Section 2, we describe the model and terminology used in the paper. In Section 3, we give an algorithm for EXPLORATION in the simpler setting where the agents have chirality. In Section 4, we use the techniques used in Section 3 to give an algorithm for EXPLORATION in the absence of chirality.

## 2 Model and Terminology

We consider a dynamic ring of size  $n$ . All nodes of the ring are identical. Each node is connected to its two neighbors via distinctly labeled ports. The labeling of the ports may not be globally consistent and thus might not provide an orientation. We consider a discrete temporal model i.e., time progresses in rounds. In each round at most one edge of the ring may be missing. Thus the ring is connected in each round. Such a network is known in the literature as a 1-interval connected ring.

We consider a team of three agents operating in the ring. The agents do not have any knowledge of the size of the ring. Each agent is provided with memory and computational capabilities. An agent can move from one node to a neighbouring node if the edge between them is not missing. Two agents moving in opposite direction on the same edge are not able to detect each other. An agent can only detect an active agent co-located at the same node i.e., if an agent terminates it becomes undetectable by any other agent. Two agents can communicate with each other only when they are present at the same node. Each agent has a unique identifier which is a bit string of length  $k > 1$ . The length  $k$  of the identifier is the same for each agent. For an agent  $r$ , its unique identifier will be denoted by  $r.ID$ . Also  $val(r.ID)$  will denote the numerical value of  $r.ID$ . For example  $val(00110) = 6$ ,  $val(10011) = 19$ , etc. Hence for any agent  $r$ ,  $val(r.ID) < 2^k$ .

Each agent has a consistent private orientation of the ring, i.e., a consistent notion of left or right. If the left and right of all three agents are the same then we say that the agents have chirality. By clockwise and counterclockwise we shall refer to the orientations of the ring in the usual sense. These terms will be used only for the purpose of description and the agents are unaware of any such global orientation if they do not have chirality. For two agents  $r_1$  and  $r_2$  on the ring,  $d^{\circlearrowright}(r_1, r_2)$  and  $d^{\circlearrowleft}(r_1, r_2)$  denotes respectively the clockwise and counterclockwise distance from  $r_1$  to  $r_2$ .

We consider a fully synchronous system, i.e., all three agents are active in each round. In each round, the agents perform the following sequence of operations:

**LOOK:** If other agents are present at the node, then the agent exchanges messages with them.

**COMPUTE:** Based on its local observation, memory and received messages, the agent performs some local computations and determines whether to move or not, and if yes, then in which direction.

**MOVE:** If the agent has decided to move in the COMPUTE phase, then the agent attempts to move in the corresponding direction. It will be able to move only if the corresponding edge is not missing. An agent can detect if it has failed to move.

During the execution of algorithm, two agents can meet each other in two possible ways: (1) two agents  $r_1$  and  $r_2$  moving in opposite direction come to the same node, or, (2) an agent  $r_1$  comes to a node where there is a stationary agent  $r_2$ . In the second case we say that  $r_1$  *catches*  $r_2$ . If two agents  $r_1, r_2$  are moving in opposite direction on the same edge in the same round, then we say that  $r_1$  and  $r_2$  *swaps over an edge*.

### 3 Exploration by Agents with Chirality

In this section, we shall assume that the agents have chirality. Since the agents have agreement in direction we shall use the terms clockwise and counterclockwise instead of right and left respectively. In Section 3.1 we present an algorithm for MEETING where at least two

agents are required to meet at a node. Then in Section 3.2 we shall use this algorithm as a subroutine to solve EXPLORATION.

### 3.1 Meeting by Agents with Chirality

We have three agents placed arbitrarily at distinct nodes of the ring. Our objective is that at least two of the agents should meet. The algorithm works in several phases. The lengths of the phases are  $2^{j+k}$ ,  $j = 0, 1, 2, \dots$ . In phase  $j$ , an agent  $r$  tries to move clockwise for the first  $\text{val}(r.ID)2^j$  rounds, and then remains stationary for  $(2^k - \text{val}(r.ID))2^j$  rounds.

We shall prove the correctness of the algorithm in Theorem 1. Before that we shall prove a lemma. This lemma will be used several times in the proofs throughout the paper.

**Lemma 1.** *Let  $r_1, r_2$  and  $r_3$  be three agents in the ring such that at round  $t$ ,  $0 \leq d^\circ(r_1, r_3) < d^\circ(r_1, r_2)$ . If  $r_1$  remains static and both  $r_2$  and  $r_3$  try to move clockwise for the next  $2n$  rounds, then within these  $2n$  rounds either  $r_2$  meets  $r_1$  or  $r_3$  meets  $r_2$ .*

*Proof.* Assume that at round  $t$ ,  $d^\circ(r_1, r_2) = x$  and  $d^\circ(r_2, r_3) = y$ . We have  $x + y \leq n$ . Within the next  $2n$  rounds, if  $r_2$  is able to move clockwise for at least  $x$  rounds, then it will meet  $r_1$  and we are done. So assume that  $r_2$  does not meet  $r_1$ . Suppose that  $r_2$  succeeds to make a move  $x' < x$  times in the next  $2n$  rounds. This means that  $r_2$  remains static for  $2n - x'$  rounds. Therefore,  $r_3$  moves clockwise in those  $2n - x'$  rounds when  $r_2$  is static. Hence,  $d^\circ(r_2, r_3)$  decreases in these rounds. Recall that initially we had  $d^\circ(r_2, r_3) = y$ . Also, in the  $x'$  rounds when  $r_2$  was able to move,  $d^\circ(r_2, r_3)$  may or may not have increased depending on whether  $r_3$  respectively failed or succeeded to move in those rounds. Now notice that

$$\begin{aligned} 2n - x' &\geq n + x + y - x' && \text{(since } n \geq x + y\text{)} \\ &> n + y && \text{(since } x > x'\text{)} \\ &> x' + y && \text{(since } n > x'\text{)} \end{aligned}$$

This implies that  $r_3$  will catch  $r_2$ . This completes the proof that within the  $2n$  rounds either  $r_2$  meets  $r_1$  or  $r_3$  meets  $r_2$ . □

**Theorem 1.** *The above algorithm solves MEETING for three agents with chirality. The algorithm ensures that the meeting takes place within  $2^{k+\lceil \log n \rceil + 2}$  rounds.*

*Proof.* Let  $p = \lceil \log 2n \rceil = \lceil \log n \rceil + 1$ . Hence  $p$  is the smallest positive integer such that  $2^p \geq 2n$ . If two agents have met before the  $p$ th phase, then we are done. If not, we show that a pair of agents is guaranteed to meet during the  $p$ th phase. Recall that in this phase, an agent  $r$  should (attempt to) rotate clockwise for  $\text{val}(r.ID)2^p$  rounds, and then remain stationary for the remaining  $(2^k - \text{val}(r.ID))2^p$  rounds. Let  $r_1$  be the first agent to come to rest in that phase, say at round  $t$ . Let  $r_2$  be the agent closest to  $r_1$  in the counterclockwise direction and  $r_3$  be the third agent. We have  $\text{val}(r_2.ID)2^p - \text{val}(r_1.ID)2^p \geq 2n$  and  $\text{val}(r_3.ID)2^p - \text{val}(r_1.ID)2^p \geq 2n$ . This implies that both  $r_2$  and  $r_3$  attempts to move for at least  $2n$  rounds after  $r_1$  comes to rest at round  $t$ . By Lemma 1, either  $r_2$  and  $r_1$  meets, or  $r_3$  and  $r_2$  meets. So the meeting takes place within  $2^k \sum_{i=0}^p 2^i < 2^{k+\lceil \log n \rceil + 2}$  rounds. □

## 3.2 Exploration with Termination by Agents with Chirality

### 3.2.1 Description of the Algorithm

We consider three agents in the ring having chirality. For simplicity assume that the agents are initially placed at distinct nodes of the ring. We shall later remove this assumption. Our plan is to first bring two of the agents at the same node using the MEETING algorithm described in Section 3.1. Then one of them will settle at that node and play the role of landmark node. Then the situation reduces to a setting similar to [20]. However we cannot use the same algorithm from [20] in our case. This is because unlike in [20] we have to ensure that the agent acting as landmark also terminates. However our algorithm uses some ideas from [20]. We now describe our algorithm in the following. The detailed pseudocode description of the algorithms are given in Algorithm 1, 2, 3, 4 and 5.

Initially all the agents start with their *state* variable set to *search*. Until an agent meets another agent, it executes the MEETING algorithm described in Section 3.1. Now according to Theorem 1, two agents are guaranteed to meet within  $2^{k+\lceil \log 2n \rceil + 2}$  rounds from the start of the algorithm. On meeting the agents compare their IDs and the one with smaller ID changes its *state* to *settled* and stops moving. The other agent changes its *winner* variable to *True* and henceforth abandons its phase-wise movement and attempts to move clockwise in each round.

Let us now describe the case when an agent with *state* *search* meets the *settled* agent. If an agent with *winner* = *False* encounters the *settled* agent it also abandons its phase-wise movement and henceforth tries to move in the clockwise direction in every round. If an agent with *winner* = *True* meets the *settled* agent  $r$ , then it indicates that it is meeting the *settled* agent for the second time and hence all nodes of the ring have been explored. The agent can also calculate the size of the ring as it is equal to the number of successful moves between the two meetings. The agent assigns this value to the variable *RSize* and also informs the *settled* agent about it. Then the agent will continue to move in the clockwise direction for  $2n$  more rounds. Both these agents will terminate after the completion of these  $2n$  rounds.

Now consider the case when an agent with *state* = *search* and *winner* = *True* meets an agent with *state* = *search* and *winner* = *False*. If the agent with *winner* = *True* already knows  $n$ , i.e., it has visited the *settled* agent twice, then both of them terminates immediately. If the agent with *winner* = *True* does not already know  $n$ , then it changes its *state* to *forward* and continues to move in the clockwise direction every round. On the other hand, the agent with *winner* = *False* changes its *state* to *bounce* and starts moving in the counterclockwise direction. This phenomenon is called the formation of *settled-forward-bounce* triplet. In this case, both the agents initiates a variable *TTime* to keep track of the number of rounds elapsed after triplet formation.

After the triplet is created, the agent with *state* *forward* will continue to move in clockwise direction. The agent with *state* *bounce* will move counterclockwise and then on fulfillment of certain conditions, it may change its *state* to *return* and start moving clockwise. Then it may again change its *state* to *bounce* and start moving counterclockwise. The period between any two such *state* changes will be called a *run*. While moving in the clock-

---

**Algorithm 1** The algorithm executed by an agent  $r$  with state search

---

1. Until another agent is found, execute the algorithm for MEETING described in Section 3.1.
  2. Upon the first meeting do the following.
    - 2.1 If two agents are encountered in the first meeting, then set  $r.state \leftarrow$  bounce and move counterclockwise. Also initiate the counter  $r.TTime$ . Otherwise if there is exactly one agent  $r'$  then do the following.
    - 2.2. If  $r'.state = search$  and  $r'.winner = False$  then do the following. If  $val(r.ID) < val(r'.ID)$ , then set  $r.state \leftarrow settled$  and remain at the current node. Otherwise if  $val(r.ID) > val(r'.ID)$ , then set  $r.winner \leftarrow True$  and keep moving clockwise until the next meeting. Also initiate a counter  $SCount$  to count the number of successful steps since the first meeting with  $r'$ .
    - 2.3. If  $r'.state = search$  and  $r'.winner = True$ , then set  $r.state \leftarrow$  bounce and move counterclockwise. Also initiate the counter  $r.TTime$ .
    - 2.4. If  $r'.state = settled$ , then keep moving clockwise until the next meeting.
  3. Upon any subsequent meeting do the following.
    - 3.1. If only one agent  $r'$  is encountered with  $r'.state = search$ , then do the following.
      - 3.1.1. If  $r'.winner = False$  and  $r.RSize = \emptyset$ , then set  $r.state \leftarrow$  forward and move clockwise. Also initiate the counter  $r.TTime$ .
      - 3.1.2. If  $r'.winner = True$  and  $r'.RSize = \emptyset$ , then set  $r.state \leftarrow$  bounce and move counterclockwise. Also initiate the counter  $r.TTime$ . Otherwise, if  $r'.winner = True$  and  $r'.RSize \neq \emptyset$ , then terminate.
    - 3.2. If only one agent  $r'$  is encountered with  $r'.state = settled$ , then do the following.
      - 3.2.1. If  $r.winner = True$ , then set  $r.RSize \leftarrow r.SCount = n$  (since  $r'$  is encountered for the second time) and inform  $r'$  about  $n$ . Keep moving clockwise for  $2n$  more rounds and then terminate.
      - 3.2.2. If  $r.winner = False$  and  $r'.RSize \neq \emptyset$ , then terminate.
    - 3.3 If two agents are encountered then there must be an agent  $r'$  with  $state$  search. Then execute 3.1.1. or 3.1.2. whichever is applicable.
-

---

**Algorithm 2** The algorithm executed by an agent  $r$  with state `settled`

---

1. Do not move. Terminate on one of the following conditions.
    - 1.1. If the other two agents are present at the same time and one of them has *state* `forward`, then terminate immediately.
    - 1.2. If an agent  $r'$  is encountered such that  $r'.SBound \neq \emptyset$ , then terminate immediately.
    - 1.3. If  $n$  is already known, i.e.,  $r.RSize \neq \emptyset$ , and an agent  $r'$  is encountered that does not know  $n$ , i.e.,  $r'.RSize = \emptyset$ , then terminate immediately.
    - 1.4. If an agent  $r'$  with state `search` informs  $r'.RSize = n$ , then terminate after  $2n$  more rounds.
    - 1.5. If an agent  $r'$  with state `forward` or `return` informs  $r'.RSize = n$  and  $r'.TTime$ , then initiate counter  $r.TTime$  with starting value  $r.TTime \leftarrow r'.TTime$  and terminate after the round when  $r.TTime = 16n$ .
- 

wise direction with *state* `forward`, the agent keeps count of the number of successful steps with *state* `forward` in the variable  $FSteps$ . The variable  $BSteps$  (resp.  $RSteps$ ) is used to keep count of the number of successful steps with *state* `bounce` (resp. `return`) in the current run. Also while moving in the counterclockwise direction with *state* `bounce`, the variable  $BBlocked$  counts the number of unsuccessful attempts to move in that run.

An agent  $r$  with *state* `bounce` will change its *state* to `return` if one of the following takes place: 1)  $r.BBlocked$  exceeds  $r.BSteps$  or 2) the agent  $r$  encounters the `settled` agent twice in the same run. An agent  $r$  with *state* `return` will change its *state* to `bounce` if  $r$  meets with the agent with *state* `forward` and  $r.RSteps > 2r.BSteps$ , where  $BSteps$  was counted in the last run with *state* `bounce`.

Here the main idea is that the agents will try to gauge the size of the ring. An agent may be able to find the size  $n$  exactly or calculate an upper bound of  $n$ . An agent can exactly find  $n$  only if it visits the static `settled` agent twice in the same direction. In this case it will also inform the `settled` agent about  $n$ . Clearly when this happens the ring has been explored completely. However the two agents cannot terminate immediately because the third agent is not aware of this. So the agents will remain active till  $TTime = 16n$ , i.e.,  $16n$  rounds from the time when the triplet was created. It should be noted here that the `settled` agent initially did not know the time when the triplet was created. It came to know about this from the  $TTime$  value of some agent that it met and initiated its own  $TTime$  counter accordingly. Now it can be shown that within these  $16n$  rounds the third agent will meet one of the two agents that already know  $n$ . These two agents will terminate immediately upon meeting. Now consider the case where an agent is able to find an upper bound of  $n$ . This happens when one of the following three takes place: 1) the forward agent meets the agent with *state* `bounce`, 2) the forward agent catches the agent with *state* `return`,

---

**Algorithm 3** The algorithm executed by an agent  $r$  with state forward

---

1. Until a new meeting, keep moving clockwise. Maintain the counters  $r.FSteps \leftarrow$  number of successful steps with state forward and  $r.TTime \leftarrow$  number of rounds since the triple was formed.
  2. If all three agents meet at a node then terminate immediately. Otherwise upon meeting an agent  $r'$ , do the following.
    - 2.1. If  $r.RSize = \emptyset \wedge r'.RSize \neq \emptyset$  or  $r.RSize \neq \emptyset \wedge r'.RSize = \emptyset$  then terminate immediately. Otherwise, do the following.
    - 2.2. If  $r'.state = settled$ , then infer  $r.RSize \leftarrow r.SCount = n$  (since  $r'$  is encountered for the second time) and inform  $r'$  about  $n$ . Keep moving clockwise and terminate after the round when  $r.TTime = 16n$ .
    - 2.3. If  $r'.state = bounce$ , set  $r.SBound \leftarrow r.FSteps + r'.BSteps$ . Then keep moving clockwise for  $r.SBound$  rounds and then terminate. If the settled agent is met meanwhile, then terminate immediately.
    - 2.4. If  $r'.state = return$ , then do the following.
      - 2.4.1. If  $r$  catches  $r'$ , then set  $r.SBound \leftarrow r.FSteps + r'.BSteps$ . Then keep moving clockwise for  $r.SBound$  rounds and then terminate. If the settled agent is met meanwhile, then terminate immediately.
      - 2.4.2. If  $r'$  catches  $r$  and  $r'.Rsteps > 2r'.Bsteps$ , then keep moving clockwise until the next meeting. Otherwise if  $r'.Rsteps \leq 2r'.Bsteps$ , then set  $r.SBound \leftarrow r.FSteps + r'.BSteps + 1$ . Then keep moving clockwise for  $r.SBound$  rounds and then terminate. If the settled agent is met meanwhile, then terminate immediately.
-

---

**Algorithm 4** The algorithm executed by an agent  $r$  with state bounce

---

1. Maintain the counters  $r.BSteps$   $\leftarrow$  number of successful steps with state bounce in the current run,  $r.BBlocked$   $\leftarrow$  number of unsuccessful attempts with state bounce in the current run and  $r.TTime$   $\leftarrow$  number of rounds since the triple was formed. Until a new meeting or fulfillment of the condition  $r.BBlocked > r.BSteps$ , keep moving in the counterclockwise direction. If  $r.BBlocked > r.BSteps$  is satisfied, change  $r.state$   $\leftarrow$  return and move clockwise.
  2. If all three agents meet at a node then terminate immediately. Otherwise upon meeting any agent  $r'$ , do the following.
    - 2.1. If  $r.RSize = \emptyset \wedge r'.RSize \neq \emptyset$  or  $r.RSize \neq \emptyset \wedge r'.RSize = \emptyset$  then terminate immediately. Otherwise, do the following.
      - 2.2. If  $r'.state = settled$  then do the following.
        - 2.2.1. If it is the first meeting with  $r'$  in the same run, then keep moving counterclockwise until a new meeting or fulfillment of the condition  $r.BBlocked > r.BSteps$ . Also initiate a counter  $SCount$  to count the number of successful steps since the first meeting with  $r'$  in the current run.
        - 2.2.2. If it is the second meeting with  $r'$  in the same run, then set  $r.RSize$   $\leftarrow$   $r.SCount = n$ . Change  $r.state$   $\leftarrow$  return and move clockwise.
      - 2.3. If  $r'.state = forward$ , then set  $r.SBound$   $\leftarrow$   $r'.FSteps + r.BSteps$ . Then keep moving counterclockwise for  $r.SBound$  rounds and then terminate. If the settled agent is met meanwhile, then terminate immediately.
-

---

**Algorithm 5** The algorithm executed by an agent  $r$  with state return

---

1. If the size of the ring is already known, i.e.,  $r.RSize = n$ , then keep moving clockwise and terminate after the round when  $r.TTime = 16n$ . Otherwise, keep moving in the clockwise direction until a new meeting. Maintain the counters  $r.RSteps \leftarrow$  number of successful steps with state return in the current run and  $r.TTime \leftarrow$  number of rounds since the triple was formed.
  2. If all three agents meet at a node then terminate immediately. Otherwise upon meeting any agent  $r'$ , do the following.
    - 2.1. If  $r.RSize = \emptyset \wedge r'.RSize \neq \emptyset$  or  $r.RSize \neq \emptyset \wedge r'.RSize = \emptyset$  then terminate immediately. Otherwise, do the following.
      - 2.2. If  $r'.state = \text{forward}$ , then do the following.
        - 2.2.1. If  $r'$  catches  $r$ , then set  $r.SBound \leftarrow r.FSteps + r'.BSteps$ . Then move counterclockwise for  $r.SBound$  rounds and then terminate. If the settled agent is met meanwhile, then terminate immediately.
        - 2.2.2. If  $r$  catches  $r'$  then do the following.
          - 2.2.2.1. If  $r.RSteps \leq 2r.BSteps$  then set  $r.SBound \leftarrow r.FSteps + r'.BSteps + 1$ . Then move counterclockwise for  $r.SBound$  rounds and then terminate. If the settled agent is met meanwhile, then terminate immediately.
          - 2.2.2.2. Otherwise, if  $r.RSteps > 2r.BSteps$ , then change  $r.state \leftarrow \text{bounce}$  and move counterclockwise.
-

3) the agent with *state* return catches the forward agent with  $RSteps \leq 2BSteps$ . It can be shown in each of the cases, these two agents will be able to correctly calculate an upper bound  $SBound$  of  $n$ . Furthermore these cases imply that the ring has been already explored completely. However the two agents cannot terminate immediately because the settled agent is not aware of this. Therefore in order to acknowledge the settled agent, these two agents will start moving in opposite directions for  $SBound$  more rounds and then terminate. Clearly one of them will be able to meet the settled agent.

### 3.2.2 Correctness

**Lemma 2.** *There exists a round  $T_1 \leq 2^{k+\lceil \log n \rceil + 2}$  when two of the agents with state search meet and a settled agent is created.*

*Proof.* This follows from Theorem 1 since the agents execute the algorithm from Section 3.1 until they meet another agent.  $\square$

**Lemma 3.** *Suppose that  $r_1$  and  $r_2$  meet at round  $T_1$  and  $r_1$  becomes settled. There is a round  $T_2$ , with  $T_1 < T_2 < 2^{k+\lceil \log n \rceil + 3}$ , when the third agent  $r_3$  meets either  $r_1$  or  $r_2$ .*

*Proof.* At round  $T_1$ ,  $r_1$  and  $r_2$  meet and  $r_1$  becomes settled. After  $T_1$ ,  $r_2$  is trying to move clockwise in each round. On the other hand, since  $r_3$  is still executing the MEETING algorithm, it will try to move clockwise on some rounds and on other rounds, will not try to move at all. Now if both  $r_2$  and  $r_3$  try to move clockwise for some  $2n$  consecutive rounds together, then by Lemma 1 either  $r_2$  meets  $r_3$  or  $r_3$  meets  $r_1$ . Clearly, this is guaranteed to happen in any phase  $l$ , where  $l \geq p = \lceil \log 2n \rceil$ . Now suppose that  $T_1$  belongs to the  $j$ th phase. By Lemma 2,  $j \leq p$ . If  $j < p$ , then the required meeting should take place in or before the  $p$ th phase and if  $j = p$ , then the required meeting will take place in  $p$ th or  $(p+1)$ th phase. Therefore, we have  $T_2 \leq \sum_{i=0}^{p+1} 2^{i+k} < 2^{k+\lceil \log n \rceil + 3}$ .  $\square$

**Lemma 4.** *Within  $T_2 + 4n$  rounds either all three agents terminate or a settled-forward-bounce triple is created.*

*Proof.* Recall that by Lemma 3, at round  $T_2$ , either  $r_3$  meets  $r_1$  or  $r_3$  meets  $r_2$ . In the latter case, a settled-forward-bounce triple is created and we are done. So consider the other case where  $r_3$  meets the settled agent  $r_1$ . Before this meeting,  $r_3$  was trying to move clockwise in some rounds, while in other rounds, it was not trying to move at all. After meeting  $r_1$ ,  $r_3$  will try moving clockwise in each round. Then by Lemma 1 within  $2n$  rounds (i.e., within  $T_2 + 2n$  rounds from the beginning), either  $r_2$  meets  $r_1$  again, or  $r_3$  meets  $r_2$ . In the latter case, a settled-forward-bounce triple is created, and we are done. So consider the other case where  $r_2$  meets  $r_1$  for the second time. Thus  $r_2$  finds out  $n$  (the size of the ring) and also informs  $r_1$  about it. Then  $r_2$  will keep moving clockwise for  $2n$  more rounds and then terminate. Also  $r_1$  will remain active for  $2n$  more rounds and then terminate. Now within these  $2n$  rounds,  $r_3$  will meet either  $r_1$  or  $r_2$  and terminate immediately. Therefore, within  $T_2 + 4n$  rounds either all three agents terminate or a settled-forward-bounce triple is created.  $\square$

Suppose that at round  $T_3 \leq T_2 + 4n$  a settled-forward-bounce triple is formed. We shall denote by  $r_1$ ,  $r_2$  and  $r_3$  the agents with states settled, forward and bounce respectively. We shall say that the agents  $r_2$  and  $r_3$  *agree on an upper bound of  $n$*  if one of the following events occur at any round after  $T_3$ . We show in Lemma 5 that indeed if one of the following events take place then the agents can find an upper bound of  $n$ .

**Event 1.** The agent  $r_2$  with state forward and  $r_3$  with state bounce meet each other at a node and neither of them know  $n$ .

**Event 2.** The agent  $r_2$  with state forward catches  $r_3$  with state return at a node and neither of them know  $n$ .

**Event 3.** The agent  $r_3$  with state return catches  $r_2$  with state forward at a node and  $r_3.Rsteps \leq 2r_3.Bsteps$  and neither of them know  $n$ .

**Lemma 5.** *If one of Event 1-3 takes place, then*

1. *exploration is complete,*
2.  *$r_2$  and  $r_3$  can infer an upper bound  $N$  of  $n$  such that  $n \leq N \leq 3n$  and will set it as  $SBound$ .*

*Proof.* **Event 1.** Consider a run of  $r_3$ , starting when it met  $r_2$  and changed its state to bounce and ending when it met  $r_2$  again (Event 1). Then it is easy to see that all nodes of the ring have been explored by  $r_2$  and  $r_3$ . Upon meeting, both  $r_2$  and  $r_3$  set  $SBound = r_2.FSteps + r_3.BSteps$ . Clearly  $SBound \geq n$ . Also since both  $r_2$  and  $r_3$  do not know  $n$ ,  $r_2.FSteps < n$  and  $r_3.BSteps < 2n$ . To see the first inequality observe that if  $r_2.FSteps \geq n$  then it implies that  $r_2$  with *state* forward has met  $r_1$ . But recall that  $r_2$  has already met  $r_1$  before when it was in *state* search and moving in the same direction. This implies that  $r_2$  can infer  $n$  by counting the number of successful steps since the first meeting. For the second inequality observe that if  $r_3.BSteps \geq 2n$  then it implies  $r_3$  has met  $r_1$  twice in the same run and therefore can infer  $n$  by counting the number of successful steps from the first and second meeting. Hence  $SBound < 3n$ .

**Event 2.** Suppose that at some node  $u$ ,  $r_2$  and  $r_3$  meet each other and continue moving in clockwise (with state forward) and counterclockwise (with state bounce) direction respectively. Then suppose that at node  $v$ ,  $r_3$  changes its state to return and its direction to clockwise. Then after sometime  $r_2$  catches  $r_3$  (Event 2). Clearly all nodes in the counterclockwise path from  $u$  to  $v$  have been visited by  $r_3$  and all nodes in the clockwise path from  $u$  to  $v$  have been visited by  $r_2$ . Hence all nodes in the ring have been together explored by  $r_2$  and  $r_3$ . Upon meeting, they both set  $SBound = r_2.FSteps + r_3.BSteps$ . Clearly  $r_3.BSteps = d^\circ(u, v)$  and  $r_2.FSteps \geq d^\circ(u, v)$ . Hence  $SBound \geq d^\circ(u, v) + d^\circ(u, v) = n$ . Also by previous argument  $SBound < 3n$ .

**Event 3.** Suppose that at some time  $r_2$  and  $r_3$  meet each other at node  $u$  and continue moving in clockwise (with state forward) and counterclockwise (with state bounce) direction respectively. Then at some time  $r_3$  changes its state to return (when we have

$r_3.BBlocked = r_3.BSteps + 1$ ) and its direction to clockwise. Then after sometime it catches  $r_2$  and finds that  $r_3.RSteps \leq 2r_3.BSteps$  (Event 3). We show that this implies that exploration is complete. If  $r_2$  and  $r_3$  swapped over an edge at some round in between, then it implies that all nodes of the ring have been explored and we are done. Otherwise we have  $r_3.RSteps = r_3.BSteps + r_2.FSteps$  at the time of meeting. Since  $r_3.RSteps \leq 2r_3.BSteps$ , we have  $r_2.FSteps \leq r_3.BSteps$ . Also recall that  $r_3.BBlocked = r_3.BSteps + 1$ . Now if  $r_2$  had been able to successfully execute a move during each of those rounds when  $r_3$  was blocked with state bounce, we must have had  $r_2.FSteps > r_3.BSteps$ . But since  $r_2.FSteps \leq r_3.BSteps$ ,  $r_2$  with state forward and  $r_3$  with state bounce must have been blocked at the same round. This can only happen if  $r_2$  and  $r_3$  are blocked on two ends of the same missing edge. This implies that the ring has been explored. Upon meeting both  $r_2$  and  $r_3$  set  $SBound = r_2.FSteps + r_3.BSteps + 1$ . If they had swapped over an edge then clearly  $SBound \geq n$ . Otherwise we showed that  $r_2$  and  $r_3$  were blocked at two adjacent nodes of the ring, say  $v$  and  $w$  respectively. Then  $r_3.BSteps \geq d^\circ(u, w)$  and  $r_2.FSteps \geq d^\circ(u, v)$ . Hence  $SBound \geq d^\circ(u, v) + d^\circ(u, w) + 1 = n$ . Also by previous argument  $SBound < 3n + 1 \leq 3n$ .  $\square$

**Lemma 6.** *Suppose that  $r_2$  and  $r_3$  meet at some round  $T$  and  $r_3$  changes its state to bounce. If they do not meet again for  $10n$  rounds then all three agents are guaranteed to find out  $n$*

*Proof.* Assume that  $r_2$  and  $r_3$  do not meet for  $10n$  rounds from  $T$ . Now at round  $T$ ,  $r_3$  changes its state to bounce. By round  $T + 4n$  either  $r_3$  has made  $2n$  successful moves in the counter-clockwise direction or the condition  $BBlocked > BSteps$  is fulfilled. In the former case  $r_3$  finds out  $n$  and changes its state to return. In the latter case also  $r_3$  changes its state to return. Therefore within  $4n$  rounds  $r_3$  is guaranteed to start moving in clockwise direction, say at round  $T' \leq T + 4n$  i.e., from  $T'$  onwards both  $r_2$  and  $r_3$  are moving in the same direction. Therefore within  $2n$  rounds either  $r_2$  and  $r_3$  meet each other or one of them meets  $r_1$ . Since we assumed that  $r_2$  and  $r_3$  do not meet, the latter takes place.

**Case 1:** Suppose that  $r_1$  meets  $r_2$ . Then both  $r_1$  and  $r_2$  find out  $n$  (if not already known). Within  $2n$  rounds  $r_3$  comes to  $r_1$  and also find out  $n$  (if not already known). So within  $4n$  rounds from  $T'$ , i.e.  $8n$  rounds from  $T$  all three agents find out  $n$ .

**Case 2:** Suppose that  $r_1$  meets  $r_3$ . Then within next  $2n$  rounds  $r_2$  meets  $r_1$ . Then by the arguments of Case 1, all three agents will find out  $n$  within next  $2n$  rounds. Therefore, within  $6n$  rounds from  $T'$ , i.e.,  $10n$  rounds from  $T$ , all three agents find out  $n$ .  $\square$

**Lemma 7.** *Within  $16n$  rounds after  $T_3$ , either all three agents find out  $n$  or  $r_2, r_3$  agree on an upper bound of  $n$ .*

*Proof.* Let us refer to the meeting of  $r_2$  and  $r_3$  at round  $T_3$  as their 1st meeting. If  $r_2$  and  $r_3$  do not meet each other again for  $16n$  rounds, then by Lemma 6 all three agents will find out  $n$ , and we are done. Therefore we assume that  $r_2$  and  $r_3$  meet again at least once after  $T_3$  within  $16n$  rounds. Let  $t_i$  denote the time between the  $i$ th and  $(i + 1)$ th meeting. In view of Lemma 6 we can assume that  $t_i \leq 10n$  because otherwise all three agents will find out  $n$ , and we are done. Also if any one of these meetings is of the type Event 1-3, then we are done as such meetings allow  $r_2$  and  $r_3$  to agree on an upper bound of  $n$  (c.f. Lemma 5). So let us

assume that such meetings do not occur. Therefore in each meeting after  $T_3$ ,  $r_3$  with state return catches  $r_2$  with state forward. If at the time of any meeting after  $T_3$ ,  $r_3$  is aware of  $n$  (by meeting  $r_1$  twice in a run with *state bounce*), then we are done. This is because  $r_2$  finds out  $n$  from  $r_3$  at the time of the meeting and  $r_1$  also had already found out  $n$  from  $r_3$ . So assume that this does not happen at any of the meetings. Notice that if we can show that one such meeting takes place after  $r_2$  visits  $r_1$ , then we are done. To see this observe that if  $r_2$  visits  $r_1$  then  $r_2$  finds out  $n$  and also informs  $r_1$  about it. Then  $r_3$  also comes to know about  $n$  in the next meeting between  $r_2$  and  $r_3$ . So we complete the proof by showing that a meeting between  $r_2$  and  $r_3$  takes place within  $16n$  rounds from  $T_3$  with  $r_2$  having already met  $r_1$ .

Recall that  $t_i$  denotes the time between the  $i$ th and  $(i+1)$ th meeting. Let  $fs_i$ ,  $bs_i$  and  $rs_i$  denote the number of successful steps made during this time by  $r_2$  with state forward,  $r_3$  with state bounce and  $r_3$  with state return respectively. First we claim that if  $r_2$  and  $r_3$  swap over an edge between their  $i$ th and  $(i+1)$ th meeting, then  $r_2$  will find out  $n$  before the  $(i+1)$ th meeting. Since the  $(i+1)$ th meeting is not of the type Event 3, we have  $rs_i > 2bs_i$ . If  $bs_i > n$ , then  $rs_i > 2n > n$ . This means that  $r_3$  with state return meet  $r_1$ . But then  $r_2$  must have met  $r_1$  before this meeting and found out  $n$ . So let  $bs_i \leq n$ . Then it is not difficult to see that  $bs_i + fs_i = n + rs_i$ . Again using  $rs_i > 2bs_i$ , we have  $fs_i > n + bs_i > n$ . This implies that  $r_2$  have met  $r_1$  and found out  $n$ . So assume that  $r_2$  and  $r_3$  do not swap in between the  $i$ th and  $(i+1)$ th meeting. Then we have  $rs_i = bs_i + fs_i$ . Furthermore we have  $t_i \leq 2bs_i + 1 + fs_i + rs_i$ . Here  $2bs_i + 1$  is an upper bound on the time needed by  $r_3$  to switch state from bounce to return. To see this, recall that  $r_3$  changes its state to return if one of the following takes place: (1) it finds out  $n$  by meeting  $r_1$  twice, (2)  $r_3.BBlocked > r_3.BSteps$  is satisfied. Since we assumed that (1) does not take place,  $r_3$  must have changed its state to return because of (2). Since this happens when the number of failed attempts to move exceeds the number of successful moves, we can conclude that  $2bs_i + 1$  is an upper bound on the time needed by  $r_3$  to switch state from bounce to return. Also,  $fs_i + rs_i$  is an upper bound on the time needed for  $r_3$  with state return to catch  $r_2$ . This is because the number of successful moves by  $r_3$  is  $rs_i$  and the number of failed attempts to move by  $r_3$  is bounded by  $fs_i$  since each failed move by  $r_3$  implies a successful move by  $r_2$ . Substituting the value of  $rs_i$  in the inequality we get,  $t_i \leq 3bs_i + 2fs_i + 1$ . Since this meeting is not of the type Event 3, we have  $rs_i > 2bs_i \implies fs_i > bs_i$ . Therefore  $t_i < 6fs_i$ . Let  $k$  be the smallest integer such that  $t_1 + \dots + t_k \geq 6n$ . So we have  $6n \leq t_1 + t_2 + \dots + t_k < 6(fs_1 + fs_2 + \dots + fs_k) \implies n < fs_1 + fs_2 + \dots + fs_k$ . This implies that  $r_2$  makes enough progress to meet  $r_1$  before the  $(k+1)$ th meeting. Therefore after the  $(k+1)$ th meeting all three robots have found out  $n$ . It remains to show that this meeting takes place within  $16n$  rounds from  $T_3$ . For this observe the following. It follows from the definition of  $k$  (which implies that  $t_1 + \dots + t_{k-1} < 6n$ ) and the fact that  $t_k \leq 10n$  that  $t_1 + \dots + t_k < 16n$ .  $\square$

**Lemma 8.** *Within  $19n$  rounds after  $T_3$  exploration of the ring is complete and all three agents terminate.*

*Proof.* By Lemma 7 within  $16n$  rounds after  $T_3$  either all three agents find out  $n$  or  $r_2, r_3$  agree on an upper bound of  $n$ . In the former case the agents will terminate when  $TTime = 16n$ , i.e., after  $16n$  rounds from  $T_3$ . Now in the latter case when  $r_2$  and  $r_3$  meet at a node to agree

on an upper bound  $N$  of  $n$ , they will move in opposite direction for  $N$  more rounds. Within these  $N$  rounds one of them will meet  $r_1$  and both will terminate. The other agents will terminate after the completion of these  $N$  rounds. Recall that by Lemma 5  $N < 3n$ . Hence in this case all the three agents terminate within  $19n$  rounds after  $T_3$ .  $\square$

**Theorem 2.** EXPLORATION with explicit termination is solvable by three agents with chirality in  $2^{k+\lceil \log n \rceil+3} + 23n = O(2^k n)$  rounds.

*Proof.* By Lemma 8 the exploration is complete and all three agents terminate within  $T_3 + 19n$  rounds. By Lemma 4  $T_3 \leq T_2 + 4n$  and by Lemma 3  $T_2 < 2^{k+\lceil \log n \rceil+3}$ . Therefore our algorithm solves EXPLORATION with explicit termination by three agents with chirality in  $2^{k+\lceil \log n \rceil+3} + 23n = O(2^k n)$  rounds.  $\square$

## 4 Exploration by Agents without Chirality

### 4.1 Contiguous Agreement

In this section we define a new problem called CONTIGUOUS AGREEMENT. Three agents with unique identifiers are placed at three different nodes in the ring. In each round, each agent chooses a direction according to a deterministic algorithm based on its ID and current round. The requirement of the problem is that the agents have to choose the same direction for some  $N$  consecutive rounds where  $N$  is a constant unknown to the agents.

#### 4.1.1 The Algorithm

Before presenting the algorithm, we describe the construction of modified identifiers which will be used in the algorithm. Recall that  $r.ID$  is a binary string of length  $k$ . We now describe the construction of the corresponding modified identifier  $r.MID$  which is a binary string of length  $\frac{k(k-1)}{2} + k + 1$ . We shall first concatenate a string of length  $\frac{k(k-1)}{2}$  at end of  $r.ID$ . Let us write  $\frac{k(k-1)}{2} = l$ . To define the string, we shall identify each position of the string as, instead of an integer from  $[l] = \{1, \dots, l\}$ , a 2-tuple from the set  $S = \{(u, v) \in [k] \times [k] \mid u < v\}$ . In order to formally describe this, let us define a bijection  $\phi : S \rightarrow [l]$  in the following way. Notice that  $|S| = l$ . Arrange the elements of  $S$  in lexicographic order. For any  $(u, v) \in S$ , we define  $\phi((u, v))$  to be the position of  $(u, v)$  in this arrangement. For example, if  $k = 4$ , then the elements of  $S$ , arranged in lexicographic order, are  $(1,2), (1,3), (1,4), (2,3), (2,4), (3,4)$ . Therefore, we have  $\phi((1,2)) = 1$ ,  $\phi((1,3)) = 2$ ,  $\phi((1,4)) = 3$ ,  $\phi((2,3)) = 4$ ,  $\phi((2,4)) = 5$  and  $\phi((3,4)) = 6$ . Now we define the string of length  $l$  that will be concatenated with  $r.ID$ . The  $i$ th bit of the string is the  $\mathbb{Z}_2$  sum of the  $u$ th and  $v$ th bit of  $r.ID$  where  $(u, v) = \phi^{-1}(i)$ . After the concatenation, we get a string of length  $k + l = k + \frac{k(k-1)}{2}$ . Finally we append 0 at the beginning of this string to obtain  $r.MID$  of length  $\frac{k(k-1)}{2} + k + 1$  (c.f. Fig. 1).

We now present the algorithm that solves CONTIGUOUS AGREEMENT. The algorithm works in phases with the length of the phases being  $2^j \left( \frac{k(k-1)}{2} + k + 1 \right)$ ,  $j = 0, 1, 2, \dots$ . For

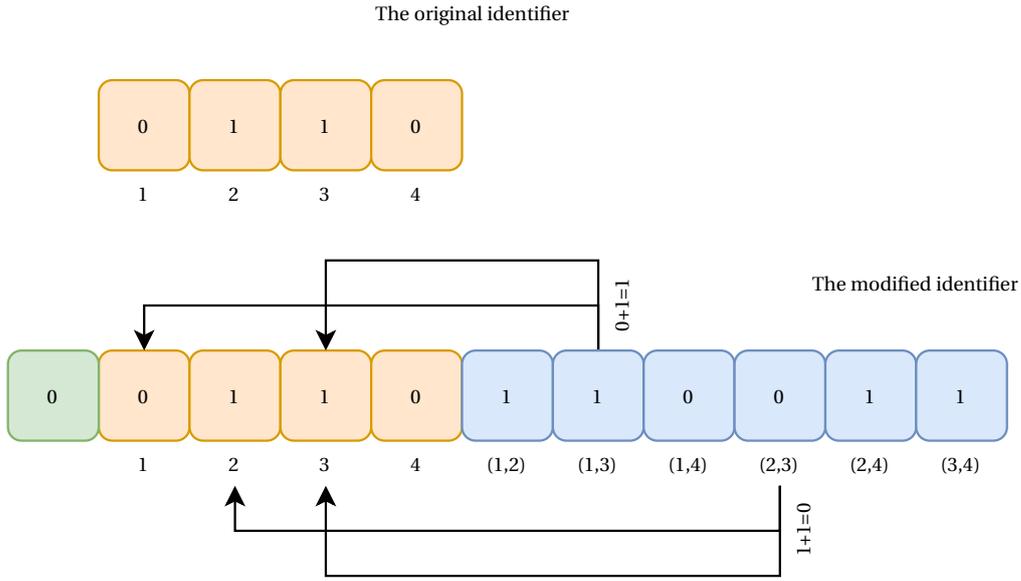


Figure 1: The construction of the modified identifier.

a string  $S$  and a positive integer  $t$ , let  $Dup(S, t)$  denote the string obtained by repeating each bit of string  $S$   $t$  times. For example,  $Dup(101, 3) = 111000111$ . For the  $j$ th phase, the agent  $r$  computes  $Dup(r.MID, 2^j)$ . Notice that the length of the  $j$ th phase is equal to the length of  $Dup(r.MID, 2^j)$ . In the  $i$ th round of the  $j$ th phase,  $r$  moves left if the  $i$ th bit of  $Dup(r.MID, 2^j)$  is 0 and otherwise moves right.

#### 4.1.2 Correctness

**Theorem 3.** *The algorithm described above solves CONTIGUOUS AGREEMENT.*

*Proof.* Let us denote the three agents by  $r_1, r_2$  and  $r_3$ . For a binary string  $S$  and an index  $1 \leq \alpha \leq |S|$ , we denote by  $S[\alpha]$  the  $\alpha$ th bit of  $S$ . We first show that in the 0th phase, there is a round in which all three agents choose the same direction. In the 0th phase, each agent  $r_i$  uses the string  $Dup(r_i.MID, 2^0) = r_i.MID$  to set its direction. First consider the case where all three agree on the orientation, say the left according to each of the agents is the counterclockwise direction. Clearly if there is an index  $\alpha$  where the strings  $r_1.MID$ ,  $r_2.MID$  and  $r_3.MID$  have the same bit, i.e.,  $r_1.MID[\alpha] = r_2.MID[\alpha] = r_3.MID[\alpha]$ , then the agents will decide the same the direction in the  $\alpha$ th round of the 0th phase. Now suppose that the agents do not agree on orientation. Consider the case where left according to  $r_1, r_2$  is the counterclockwise direction (i.e.,  $r_1$  and  $r_2$  have agreement on orientation), while left according to  $r_3$  is the clockwise direction. Clearly in this case, if there is an index  $\alpha$  such that  $r_1.MID[\alpha] = r_2.MID[\alpha] \neq r_3.MID[\alpha]$ , then the agents will decide the same the direction in the  $\alpha$ th round of the 0th phase. Therefore, it follows from the above discussion that it suffices to show that  $\exists$  indices  $\alpha, \beta, \gamma$  and  $\eta$  such that,

1.  $r_1.MID[\alpha] = r_2.MID[\alpha] = r_3.MID[\alpha]$

2.  $r_1.MID[\beta] = r_2.MID[\beta] \neq r_3.MID[\beta]$
3.  $r_1.MID[\gamma] = r_3.MID[\gamma] \neq r_2.MID[\gamma]$
4.  $r_2.MID[\eta] = r_3.MID[\eta] \neq r_1.MID[\eta]$

Recall that each of the strings start with 0. Hence, we have  $\alpha = 1$ . Since  $r_1.ID \neq r_2.ID$ ,  $\exists$  an index  $a$  so that  $r_1.ID[a] \neq r_2.ID[a]$ . Without loss of generality let  $r_3.ID[a] = r_1.ID[a]$ , i.e.,  $r_1.ID[a] = r_3.ID[a] \neq r_2.ID[a]$ . So we let  $\gamma = a + 1$  which fulfills the requirement that  $r_1.MID[\gamma] = r_3.MID[\gamma] \neq r_2.MID[\gamma]$ . We add 1 here because of the 0 appended at the beginning of the MIDs. Similarly since  $r_1.ID \neq r_3.ID$ ,  $\exists$  an index  $b \neq a$  so that  $r_1.ID[b] \neq r_3.ID[b]$ . Without loss of generality let  $r_2.ID[b] = r_1.ID[b]$ , i.e.,  $r_1.ID[b] = r_2.ID[b] \neq r_3.ID[b]$ . So we let  $\beta = b + 1$ . So now it remains to find the index  $\eta$ . We take  $\eta$  to be the index of MID where we put the sum of the  $\beta$ th bit of MID ( $\beta - 1$ th bit of ID) and the  $\gamma$ th bit of MID ( $\gamma - 1$ th bit of ID). We have to show that  $r_2.MID[\eta] = r_3.MID[\eta] \neq r_1.MID[\eta]$ . For the equality, observe the following.

$$\begin{aligned}
r_2.MID[\eta] &= r_2.MID[\beta] + r_2.MID[\gamma] \\
&= (1 + r_3.MID[\beta]) + (1 + r_3.MID[\gamma]) \quad (\text{for } x, y \in \mathbb{Z}_2, x \neq y \iff x = y + 1) \\
&= r_3.MID[\beta] + r_3.MID[\gamma] \\
&= r_3.MID[\eta]
\end{aligned}$$

To prove the inequality, we assume for the sake of contradiction that  $r_1.MID[\eta] = r_2.MID[\eta]$ . This leads to a contradiction as shown in the following.

$$\begin{aligned}
r_1.MID[\eta] &= r_2.MID[\eta] \\
\implies r_1.MID[\beta] + r_1.MID[\gamma] &= r_2.MID[\beta] + r_2.MID[\gamma] \\
\implies r_2.MID[\beta] + (1 + r_2.MID[\gamma]) &= r_2.MID[\beta] + r_2.MID[\gamma] \\
\implies 1 &= 0
\end{aligned}$$

Hence, we show that there is round in the 0th phase where all three agents will decide the same direction. It immediately follows from the proof that there are  $2^j$  consecutive rounds in the  $j$ th phase where all three agents will choose the same direction. Hence, for  $j = \lceil \log N \rceil$ , the agents will choose the same direction for  $N$  consecutive rounds in the  $j$ th phase.  $\square$

## 4.2 Meeting by Agents without Chirality

In Section 3.1, we describe an algorithm that solves MEETING by agents with chirality. In the current setting where agents do not have chirality, the main idea is to use the strategy

of CONTIGUOUS AGREEMENT so that the agents can implicitly agree on a common direction and solve MEETING by employing the strategy from section 3.1. Similar to the algorithm for CONTIGUOUS AGREEMENT, our algorithm for MEETING also works in phases. In the algorithm for CONTIGUOUS AGREEMENT the length of the phases were  $2^j \left( \frac{k(k-1)}{2} + k + 1 \right)$ ,  $j = 0, 1, 2, \dots$ . For MEETING, the phases will be of length  $2^{j+k} \left( \frac{k(k-1)}{2} + k + 1 \right)$ ,  $j = 0, 1, 2, \dots$ . In the  $j$ th phase of the algorithm an agent  $r$  uses the string  $Dup(r.MID, 2^{j+k})$  to decide its movement. Notice that the length of  $Dup(r.MID, 2^{j+k})$  is equal to the length of the  $j$ th phase. The string  $Dup(r.MID, 2^{j+k})$  is a concatenation of  $\left( \frac{k(k-1)}{2} + k + 1 \right)$  blocks of length  $2^{j+k}$  where each block consists of all 0's or all 1's. Our plan is to simulate the MEETING algorithm from section 3.1. So, in the  $2^{j+k}$  rounds corresponding to each block, the agent  $r$  will (try to) move in the first  $val(r.ID)2^j$  rounds and will be stationary for the remaining  $(2^k - val(r.ID))2^j$  rounds. If the block consists of 0's, then the movement will be towards left and otherwise towards right.

**Theorem 4.** *The above algorithm solves MEETING for three agents without chirality. The algorithm ensures that the meeting takes place within  $k^2 2^{k+\lceil \log n \rceil + 3}$  rounds.*

*Proof.* Let  $p = \lceil \log 2n \rceil$ . If two agents have met before the  $p$ th phase, then we are done. Otherwise, we show that a pair of agents must meet during the  $p$ th phase. Recall that in the  $p$ th phase, an agent  $r$  uses the bits of the string  $Dup(r.MID, 2^{p+k})$  to decide its movement in each round. Each block of  $Dup(r.MID, 2^{p+k})$  is  $2^{p+k}$  bits long. From the proof of Theorem 3 it follows that there is a block in  $Dup(r.MID, 2^{p+k})$  so that the agents have an agreement in direction in the rounds corresponding to that block. Recall that the agents simulate the MEETING algorithm from section 3.1 in the rounds corresponding to each block. Hence it follows from Theorem 1 that two agents are guaranteed to meet during the rounds corresponding to the aforesaid block. Therefore, the algorithm ensures that the meeting takes place within  $2^k \left( \frac{k(k-1)}{2} + k + 1 \right) \sum_{i=0}^p 2^i < k^2 2^{k+\lceil \log n \rceil + 3}$  rounds.  $\square$

### 4.3 Exploration with Termination by Agents without Chirality

For simplicity assume that the agents are placed arbitrarily at distinct nodes of the ring. We shall remove this assumption at the end of this section. Initially the *state* variable of all three agents are set to search. We shall adopt a strategy similar to the one used in Section 3. In fact, we only need to make some modifications in the algorithms to be executed by the agents with *state* search and settled.

The agents will execute the MEETING algorithm described in Section 4.2 until another agent is encountered. The agents will keep count of the number of rounds since the beginning in the variable *STime*. Now by Theorem 4 two of the agents are guaranteed to meet within  $k^2 2^{k+\lceil \log n \rceil + 3}$  rounds. Upon meeting the agents will agree on a common direction, say the right direction of the agent with larger ID. Without loss of generality assume that the agreed direction is the clockwise direction. The agent with smaller ID, say  $r_1$ , will become the settled agent and the one with larger ID, say  $r_2$ , will continue moving in the clockwise direction. The agent  $r_1$  will save the port number leading to the agreed direction, i.e., clockwise.

Let  $r_3$  denote the third agent which is still executing the MEETING algorithm. Using similar arguments as in Lemma 3 we can show that a second meeting is guaranteed to take place on or before  $\lceil \log 2n \rceil + 1$ th phase, i.e., within  $k^2 2^{k+\lceil \log n \rceil+4}$  rounds from the start of the algorithm. However unlike in Section 3 where the agents had chirality, here the second meeting may also take place between  $r_1$  and  $r_2$ . This is because  $r_3$  is moving in clockwise direction in some rounds and counterclockwise in other rounds. Hence there is a possibility that  $r_2$  and  $r_3$  may swap over an edge and  $r_1$  meets  $r_2$  first. However even then it is not difficult to see that  $r_3$  is guaranteed to meet  $r_1$  or  $r_2$  on or before  $(\lceil \log 2n \rceil + 1)$ th phase i.e., within  $k^2 2^{k+\lceil \log n \rceil+4}$  rounds from the start of the algorithm. To see this, observe that  $r_2$  and  $r_3$  will try to move in the same direction for some  $4n$  consecutive rounds in the  $(\lceil \log 2n \rceil + 1) = \lceil \log 4n \rceil$ th phase. Within the first  $2n$  rounds a meeting should take place by Lemma 1. If this meeting involves  $r_3$  then we are done. Otherwise  $r_1$  and  $r_2$  meet each other and then by again applying by Lemma 1,  $r_3$  will meet one of them within the following  $2n$  rounds.

Now consider the following cases depending on which of the two robots meet on the second meeting.

1. Suppose that the second meeting takes place between  $r_1$  and  $r_2$ . In this case the ring has been explored and  $r_2$  finds out  $n$  and informs  $r_1$  about it. Then  $r_2$  will continue moving in the clockwise direction. Both agents will terminate after the round when  $S\text{Time} = k^2 2^{k+\lceil \log n \rceil+4}$ . Recall that  $r_3$  is guaranteed to meet one of them in the meantime and will terminate immediately.
2. Suppose that the second meeting takes place between  $r_2$  and  $r_3$ . Then  $r_2$  informs  $r_3$  about the agreed direction. Hence the case reduces to the setting of Section 3. Therefore  $r_2$  and  $r_3$  will change their *state* to forward and bounce respectively and execute the algorithms as before.
3. Now suppose that the second meeting takes place between  $r_1$  and  $r_3$ . In this case  $r_3$  will come to know about the agreed direction and again the case reduces to the setting of Section 3. So  $r_3$  will continue to move in the agreed direction i.e., clockwise.

**Theorem 5.** EXPLORATION *with explicit termination is solvable by three agents without chirality in  $k^2 2^{k+\lceil \log n \rceil+4} + 23n = O(k^2 2^k n)$  rounds.*

*Proof.* It follows from the above discussions and the results proved in Section 3 that the agents will terminate after exploring the ring within  $k^2 2^{k+\lceil \log n \rceil+4} + 23n = O(k^2 2^k n)$  rounds.  $\square$

Recall that we assumed that the agents are placed initially at distinct nodes of the ring. We now show that this assumption is not necessary if the initial configuration has two agents  $r_1, r_2$  at the same node and the third agent  $r_3$  at a different node. Then the case reduces to the situation when the first meeting takes place. Then  $r_1$  and  $r_2$  will change their *state* to settled and forward while  $r_3$  will execute the MEETING algorithm with *state* search. The algorithm will progress as before and achieve exploration with termination.

If all three agents are in the same node in the initial configuration then the three agents will compare their identifiers and will change their *state* to settled, forward and bounce accordingly. Again, the algorithm will progress as before and achieve exploration with termination.

## 5 Concluding Remarks

We showed that EXPLORATION (with explicit termination) in a dynamic always connected ring without any landmark node is solvable by three non-anonymous agents without chirality. This is optimal in terms of the number of agents used as the problem is known to be unsolvable by two agents. Our algorithm takes  $O(k^2 2^k n)$  rounds to solve the problem where  $n$  is the size of the ring and  $k$  is the length of the identifiers of the agents. An interesting question is whether the problem can be solved in  $O(\text{poly}(k)n)$  rounds. However with  $k = O(1)$  the round complexity is  $O(n)$ . This is asymptotically optimal as there are  $n$  nodes to be explored and in each round three agents can visit at most three nodes.

A challenging problem that remains open is EXPLORATION in a dynamic network of arbitrary underlying topology. Except for some bounds on the number of agents obtained in the recent work [13], almost nothing is known. As for other special topologies only the case of torus has been studied [14]. Comparison of what can be achieved by anonymous and non-anonymous agents in these settings is an direction to explore.

## References

- [1] Ankush Agarwalla, John Augustine, William K. Moses Jr., Sankar Madhav K., and Arvind Krishna Sridhar. Deterministic dispersion of mobile robots in dynamic rings. In *Proceedings of the 19th International Conference on Distributed Computing and Networking, ICDCN 2018, Varanasi, India, January 4-7, 2018*, pages 19:1–19:4. ACM, 2018. [doi:10.1145/3154273.3154294](https://doi.org/10.1145/3154273.3154294). 3
- [2] Susanne Albers and Monika Rauch Henzinger. Exploring unknown environments. *SIAM J. Comput.*, 29(4):1164–1188, 2000. [doi:10.1137/S009753979732428X](https://doi.org/10.1137/S009753979732428X). 3
- [3] Marjorie Bournat, Swan Dubois, and Franck Petit. Gracefully degrading gathering in dynamic rings. In *20th International Symposium on Stabilization, Safety, and Security of Distributed Systems, SSS 2018, Tokyo, Japan, November 4-7, 2018, Proceedings*, volume 11201 of *Lecture Notes in Computer Science*, pages 349–364. Springer, 2018. [doi:10.1007/978-3-030-03232-6\\_23](https://doi.org/10.1007/978-3-030-03232-6_23). 3
- [4] Arnaud Casteigts, Paola Flocchini, Walter Quattrociocchi, and Nicola Santoro. Time-varying graphs and dynamic networks. *Int. J. Parallel Emergent Distributed Syst.*, 27(5):387–408, 2012. [doi:10.1080/17445760.2012.668546](https://doi.org/10.1080/17445760.2012.668546). 2

- [5] Shantanu Das, Paola Flocchini, Shay Kutten, Amiya Nayak, and Nicola Santoro. Map construction of unknown graphs by multiple agents. *Theor. Comput. Sci.*, 385(1-3):34–48, 2007. [doi:10.1016/j.tcs.2007.05.011](https://doi.org/10.1016/j.tcs.2007.05.011). 3
- [6] Shantanu Das, Giuseppe Antonio Di Luna, and Leszek Antoni Gasieniec. Patrolling on dynamic ring networks. In *45th International Conference on Current Trends in Theory and Practice of Computer Science, SOFSEM 2019, Nový Smokovec, Slovakia, January 27-30, 2019, Proceedings*, volume 11376 of *Lecture Notes in Computer Science*, pages 150–163. Springer, 2019. [doi:10.1007/978-3-030-10801-4\\_13](https://doi.org/10.1007/978-3-030-10801-4_13). 3
- [7] Xiaotie Deng and Christos H. Papadimitriou. Exploring an unknown graph. *J. Graph Theory*, 32(3):265–297, 1999. 3
- [8] Yoann Dieudonné and Andrzej Pelc. Deterministic network exploration by anonymous silent agents with local traffic reports. *ACM Trans. Algorithms*, 11(2):10:1–10:29, 2014. [doi:10.1145/2594581](https://doi.org/10.1145/2594581). 3
- [9] Thomas Erlebach, Michael Hoffmann, and Frank Kammer. On temporal graph exploration. In *42nd International Colloquium on Automata, Languages, and Programming, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part I*, volume 9134 of *Lecture Notes in Computer Science*, pages 444–455. Springer, 2015. [doi:10.1007/978-3-662-47672-7\\_36](https://doi.org/10.1007/978-3-662-47672-7_36). 3
- [10] Paola Flocchini, Matthew Kellett, Peter C. Mason, and Nicola Santoro. Map construction and exploration by mobile agents scattered in a dangerous network. In *23rd IEEE International Symposium on Parallel and Distributed Processing, IPDPS 2009, Rome, Italy, May 23-29, 2009*, pages 1–10. IEEE, 2009. [doi:10.1109/IPDPS.2009.5161080](https://doi.org/10.1109/IPDPS.2009.5161080). 3
- [11] Paola Flocchini, Bernard Mans, and Nicola Santoro. On the exploration of time-varying networks. *Theor. Comput. Sci.*, 469:53–68, 2013. [doi:10.1016/j.tcs.2012.10.029](https://doi.org/10.1016/j.tcs.2012.10.029). 3
- [12] Pierre Fraigniaud, David Ilcinkas, Guy Peer, Andrzej Pelc, and David Peleg. Graph exploration by a finite automaton. *Theor. Comput. Sci.*, 345(2-3):331–344, 2005. [doi:10.1016/j.tcs.2005.07.014](https://doi.org/10.1016/j.tcs.2005.07.014). 3
- [13] Tsuyoshi Gotoh, Paola Flocchini, Toshimitsu Masuzawa, and Nicola Santoro. Exploration of dynamic networks: Tight bounds on the number of agents. *Journal of Computer and System Sciences*, 122:1–18, 2021. [doi:10.1016/j.jcss.2021.04.003](https://doi.org/10.1016/j.jcss.2021.04.003). 3, 22
- [14] Tsuyoshi Gotoh, Yuichi Sudo, Fukuhito Ooshita, Hirotsugu Kakugawa, and Toshimitsu Masuzawa. Group exploration of dynamic tori. In *38th IEEE International Conference on Distributed Computing Systems, ICDCS 2018, Vienna, Austria, July 2-6, 2018*, pages 775–785. IEEE Computer Society, 2018. [doi:10.1109/ICDCS.2018.00080](https://doi.org/10.1109/ICDCS.2018.00080). 3, 22
- [15] David Ilcinkas, Ralf Klasing, and Ahmed Mouhamadou Wade. Exploration of constantly connected dynamic graphs based on cactuses. In *21st International Colloquium on*

- Structural Information and Communication Complexity, SIROCCO 2014, Takayama, Japan, July 23-25, 2014. Proceedings*, volume 8576 of *Lecture Notes in Computer Science*, pages 250–262. Springer, 2014. doi:[10.1007/978-3-319-09620-9\\_20](https://doi.org/10.1007/978-3-319-09620-9_20). 3
- [16] David Ilcinkas and Ahmed Mouhamadou Wade. On the power of waiting when exploring public transportation systems. In *15th International Conference on Principles of Distributed Systems, OPODIS 2011, Toulouse, France, December 13-16, 2011. Proceedings*, volume 7109 of *Lecture Notes in Computer Science*, pages 451–464. Springer, 2011. doi:[10.1007/978-3-642-25873-2\\_31](https://doi.org/10.1007/978-3-642-25873-2_31). 3
- [17] David Ilcinkas and Ahmed Mouhamadou Wade. Exploration of the T-interval-connected dynamic graphs: The case of the ring. In *20th International Colloquium on Structural Information and Communication Complexity, SIROCCO 2013, Ischia, Italy, July 1-3, 2013*, volume 8179 of *Lecture Notes in Computer Science*, pages 13–23. Springer, 2013. doi:[10.1007/978-3-319-03578-9\\_2](https://doi.org/10.1007/978-3-319-03578-9_2). 3
- [18] Ajay D. Kshemkalyani, Anisur Rahaman Molla, and Gokarna Sharma. Efficient dispersion of mobile robots on dynamic graphs. In *40th IEEE International Conference on Distributed Computing Systems, ICDCS 2020, Singapore, November 29 - December 1, 2020*, pages 732–742. IEEE, 2020. doi:[10.1109/ICDCS47774.2020.00100](https://doi.org/10.1109/ICDCS47774.2020.00100). 3
- [19] Fabian Kuhn and Rotem Oshman. Dynamic networks: models and algorithms. *SIGACT News*, 42(1):82–96, 2011. doi:[10.1145/1959045.1959064](https://doi.org/10.1145/1959045.1959064). 2
- [20] Giuseppe Antonio Di Luna, Stefan Dobrev, Paola Flocchini, and Nicola Santoro. Distributed exploration of dynamic rings. *Distributed Comput.*, 33(1):41–67, 2020. doi:[10.1007/s00446-018-0339-1](https://doi.org/10.1007/s00446-018-0339-1). 2, 3, 4, 7
- [21] Giuseppe Antonio Di Luna, Paola Flocchini, Linda Pagli, Giuseppe Prencipe, Nicola Santoro, and Giovanni Viglietta. Gathering in dynamic rings. *Theor. Comput. Sci.*, 811:79–98, 2020. doi:[10.1016/j.tcs.2018.10.018](https://doi.org/10.1016/j.tcs.2018.10.018). 3
- [22] Subhrangsu Mandal, Anisur Rahaman Molla, and William K. Moses Jr. Live exploration with mobile robots in a dynamic ring, revisited. In *16th International Symposium on Algorithms and Experiments for Wireless Sensor Networks, ALGOSENSORS 2020, Pisa, Italy, September 9-10, 2020*, volume 12503 of *Lecture Notes in Computer Science*, pages 92–107. Springer, 2020. doi:[10.1007/978-3-030-62401-9\\_7](https://doi.org/10.1007/978-3-030-62401-9_7). 2, 3, 4
- [23] Othon Michail and Paul G. Spirakis. Traveling salesman problems in temporal graphs. *Theor. Comput. Sci.*, 634:1–23, 2016. doi:[10.1016/j.tcs.2016.04.006](https://doi.org/10.1016/j.tcs.2016.04.006). 3
- [24] Fukuhito Ooshita and Ajoy K. Datta. Brief announcement: Feasibility of weak gathering in connected-over-time dynamic rings. In *20th International Symposium on Stabilization, Safety, and Security of Distributed Systems, SSS 2018, Tokyo, Japan, November 4-7, 2018, Proceedings*, volume 11201 of *Lecture Notes in Computer Science*, pages 393–397. Springer, 2018. doi:[10.1007/978-3-030-03232-6\\_27](https://doi.org/10.1007/978-3-030-03232-6_27). 3

- [25] Petrisor Panaite and Andrzej Pelc. Exploring unknown undirected graphs. *J. Algorithms*, 33(2):281–295, 1999. [doi:10.1006/jagm.1999.1043](https://doi.org/10.1006/jagm.1999.1043). 3
- [26] Nicola Santoro. Time to change: On distributed computing in dynamic networks (Keynote). In *19th International Conference on Principles of Distributed Systems, OPODIS 2015, December 14-17, 2015, Rennes, France*, volume 46 of *LIPICs*, pages 3:1–3:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2015. [doi:10.4230/LIPICs.OPODIS.2015.3.2](https://doi.org/10.4230/LIPICs.OPODIS.2015.3.2)