Efficient Live Exploration of a Dynamic Ring with Mobile Robots[⋆]

Subhrangsu Mandal[a,1,*], Anisur Rahaman Molla[b,2], William K. Moses Jr.[c,3]

[a]*Department of Computer Science and Engineering, Indian Institute of Technology Indore, India.*
[b]*Computer and Communication Sciences Division, Indian Statistical Institute, Kolkata, India.*
[c]*Department of Computer Science, Durham University, Durham, UK.*

Abstract

The graph exploration problem requires a group of mobile robots, initially placed arbitrarily on the nodes of a graph, to work collaboratively to explore the graph such that each node is eventually visited by at least one robot. One important requirement of exploration is the *termination* condition, i.e., the robots must know that exploration is completed. The problem of live exploration of a dynamic ring using mobile robots was recently introduced by Di Luna et al. [DC 2020]. More specifically, they studied the problem of live exploration in a dynamic ring and proposed multiple algorithms to solve this problem in the fully synchronous and semi-synchronous settings with various guarantees when 2 robots were involved. They also showed that, under certain assumptions, exploration of the ring using two robots was impossible. An important question left open was how the presence of 3 robots would affect the results. In this paper, we try to settle this question in a fully synchronous setting and also show how to extend our results to a semi-synchronous setting.

In particular, we present algorithms for exploration with explicit termination using 3 robots in conjunction with either (i) unique IDs of the robots and edge crossing detection capability (i.e., two robots moving in opposite directions through an edge in the same round can detect each other), or (ii) access to randomness. The time complexity of our deterministic algorithm is asymptotically optimal. We also provide complementary impossibility results showing that there does not exist any explicit termination algorithm for 2 robots even when each robot has a unique ID, edge crossing detection capability, and access to randomness. The theoretical analysis and comprehensive simulations of our algorithm show the effectiveness and efficiency of the algorithm in dynamic rings. We also present an algorithm to achieve exploration with partial termination using 3 robots with unique IDs in the semi-synchronous

[⋆]A preliminary version of this article appears in the Proceedings of ALGOSENSORS 2020 [1].

[*]Corresponding author.

Email addresses: `santu.cst@gmail.com` (Subhrangsu Mandal), `molla@isical.ac.in` (Anisur Rahaman Molla), `wkmjr3@gmail.com` (William K. Moses Jr.)

[1]S. Mandal was supported, in part, by YFRSG research grant IITI/YFRSG/2022-23/25, Indian Institute of Technology Indore.

[2]A. R. Molla was supported, in part, by DST Inspire Faculty research grant DST/INSPIRE/04/2015/002801, Govt. of India.

[3]Part of the work was done while W. K. Moses Jr. was a post doctoral fellow at the University of Houston in Houston, USA and a post doctoral fellow at the Technion - Israel Institute of Technology in Haifa, Israel. This work was supported in part by a Technion fellowship, in part by NSF grants CCF-1540512, IIS-1633720, and CCF-1717075, and in part by BSF grant 2016419.

setting, when robots have access to edge crossing detection capability and randomness but do not know a bound on the size of the ring or have access to a landmark or are guaranteed that robots have common chirality. Our algorithms are fully decentralized, lightweight, and easily implementable.

Keywords: multi-agent systems, mobile robots, exploration, uniform deployment, distributed algorithms, dynamic graph, ring graph

1. Introduction

The research area of autonomous mobile robots in a graph setting has been well studied over the years. Many fundamental problems have been studied in this area, such as exploration [2, 3], dispersion [4, 5], gathering [6, 7], and scattering [8, 9] among others. In particular, the problem of exploration has attracted much interest [10, 11, 12, 13, 14, 15, 16]. In this problem, multiple robots are placed on nodes in the graph and the goal is to design an algorithm, run by each robot, such that all robots collectively visit each node at least once as quickly as possible. As this fundamental problem has been solved to a large degree in most vanilla settings [17, 18, 19, 20, 21], its study has been extended to more exotic, but realistic, settings.

One such setting is a dynamic network. In the real world, dynamism is seen fairly regularly in networks. Like most things in real life, the dynamism that appears in the real world is quite complex. In order to work towards a deeper understanding of this complexity, we first start with a simple model of dynamism, which is a restricted version of *1-interval connectivity* applied to a ring in the synchronous setting. We now describe this version of dynamism. Consider n vertices, $v_0, v_1, \ldots, v_{n-1}$, with an undirected edge between every node v_i and $v_{(i+1) \bmod n}$, $\forall i : 0 \leq i \leq n-1$. In each round, the adversary can choose to remove at most one edge of the ring. Note that if the adversary removes an edge (v_i, v_{i+1}) from the ring in some round, the edge is removed only for that round. In the subsequent round, all edges are considered to be present and the adversary chooses from among all edges at most one to remove (the adversary may choose to remove the same edge again).

In this setting, Di Luna et al. [22] were the first to study the problem of graph exploration when robots do not know what the adversary will do next (*live* or *online* dynamism). This is contrasted with another scenario, called *post-mortem* dynamism, where robots have complete knowledge of how the adversary will control dynamism in every round. In [22], Di Luna et al. studied both fully synchronous systems and semi-synchronous systems where nodes are anonymous, i.e., nodes do not have unique IDs. In the fully synchronous setting, they have shown that by using just 2 robots without unique IDs, subject to some assumptions, deterministic exploration of a ring in the presence of 1-interval connectivity is possible with termination detection. These assumptions include a mix of the following ideas: (i) robots have knowledge of the value of n, (ii) there exists a *landmark* (a unique node that can be identified by robots as being unique), (iii) robots have common chirality (a common sense of clockwise/counterclockwise). They differentiate between *explicit termination* detection where all robots can detect the completion of exploration and subsequently terminate, and *partial termination* detection where at least one of the robots (but not necessarily all of them) detects completion and terminates. They have shown that deterministic ring exploration with explicit termination is possible with 2 robots with the aforementioned assumptions. They have also provided matching impossibility results that deterministic exploration with partial termination is impossible with 2 robots when n is unknown and no landmark is available, even in the presence of robots with unique IDs and common chirality. They have also shown that if n is unknown, no landmark is available and the robots are anonymous, then regardless of the number of robots initially deployed on the ring, deterministic exploration with partial termination is impossible. This impossibility holds even if those robots have common chirality. An important question left unanswered is if exploration with ≥ 3 robots is possible when no knowledge of n is known and no landmark is available but robots may have IDs. In this paper, we further extended the exploration problem in the dynamic ring and try to settle this question in the fully synchronous setting and provide partial results in the semi-synchronous setting.

1.1. Our Contributions

In this paper, we look into exploration of a dynamic ring with 3 robots and show various positive results when certain assumptions are made.

We show that deterministic exploration of a dynamic ring of size n with explicit termination detection is indeed possible with 3 robots when n is unknown and no landmark is present. In fact, not only is exploration possible, but the running time of our algorithm (which is linear in the size of the ring) is asymptotically optimal. We require robots to have unique IDs and have the capability of edge crossing detection, i.e., two robots passing through the same edge in a given round in opposite directions can detect that they passed each other in that round. We also implement our algorithm and show that it outperforms the theoretical time bound for different parameter ranges. Further, we show that the algorithm also works for variable length IDs of the robots (see Section 5.3).

We subsequently remove the need for the edge crossing detection assumption with the help of randomness. We also show how to use randomness to remove the need for robots to have unique IDs. Note that this result – we achieve explicit termination with 3 anonymous robots, no landmark, no knowledge of n, but access to randomness – is in sharp contrast to the impossibility result of [22] where even partial termination with any number of robots is impossible under the same setting but without access to randomness. We also show how to modify our algorithm to achieve partial termination with better runtime. Our positive results are summarized in Table 1.

One may wonder if either the use of edge crossing detection or the use of randomness is sufficient for 2 robots to bypass the impossibility result from [22]. We show that when robots only have access to edge crossing detection, exploration with partial termination of 2 robots is impossible. We further show that even when the use of randomness is also allowed, exploration with explicit termination of 2 robots is impossible. Thus, we see that only with the use of 3 robots do either of these capabilities provide sufficient power to overcome the impossibility of exploration with explicit termination. Our impossibility results are summarized in Table 2 along with a comparison to the impossibility result from [22].

Finally, we show how to use the ideas we built up throughout the paper to achieve partial termination in the semi-synchronous setting when robots neither know an upper bound on the value of n nor have access to a landmark. The algorithm uses 3 robots with unique IDs and access to the edge crossing detection capability.

Table 1: Fully synchronous setting, possibility with 3 robots.

Assumptions	Running time (for explicit termination)
Non-anonymous robots, Edge crossing detection	$O(n)$ rounds
Non-anonymous robots, Access to randomness	$O(n \log n)$ rounds on expectation*
Access to randomness	$O((n + n \cdot 2^l) \log n)$ rounds on expectation**
* Explicit termination with probability $\geq 1 - 1/n$.	
** Explicit termination with probability $\geq (1 - O(1/2^l))(1 - 1/n)$, where l is an input to the algorithm.	

Table 2: Fully synchronous setting, impossibility results.

Paper	# Robots	Assumptions	Even with Assumptions	Which termination impossible
[22]	2	No knowledge of n, No landmark	Non-anonymous robots, Chirality	Partial
[22]	Any	No knowledge of n, No landmark, Anonymous robots	Chirality	Partial
This paper	2	No knowledge of n, No landmark	Non-anonymous robots, Chirality, Edge crossing detection	Partial
This paper	2	No knowledge of n, No landmark	Non-anonymous robots, Chirality, Edge crossing detection, Access to randomness	Explicit

1.2. Organization of Paper

In Section 2, we discuss related work in the area. In Section 3, we elaborate on the exact model of the system. In Section 4, we present our impossibility results for termination with just 2 robots. In Section 5, we develop our algorithm to achieve exploration with explicit termination using 3 robots and provide experimental data to illustrate its efficiency. In Section 6, we show how to remove the requirement of edge crossing detection and unique IDs for robots in our algorithm through the creative application of access to randomness. In Section 7, we extend our deterministic algorithm to achieve exploration with partial termination using 3 robots in the semi-synchronous model. Finally, we conclude with the future research directions in Section 8.

2. Related Work

Exploration of static anonymous graphs using mobile robots has been studied for a very long time. A good survey on the topic is presented in [23, 24]. Exploration on anonymous graphs with 1-interval connected dynamism is relatively new and the first paper to study it in the current model is [22]. In the paper, they look at exploration problem in a ring under 1-interval connected dynamism and provide various deterministic algorithms to solve the problem using 2 robots for various assumptions. It should be noted that the way 1-interval connectivity is defined in their paper and also in the current paper is different from the original definition proposed in [25, 26]. Specifically, the original definition of 1-interval connectivity allows for permutations of the nodes of the graph, whereas in [22], the nodes remain stationary and the adversary can only choose whether to temporarily remove at most one of a fixed set of edges per round.

A randomized approach to graph exploration is presented in [27] via random walk, however the model of dynamism they look at is slightly different. Their approach is that of a lazy random walk, but when the rate of change of the graph is very fast, i.e., every round the adversary changes the graph, then things become complicated. Essentially their approach may take $\Omega(n^2)$ time to explore a dynamic ring of size n, however, it cannot guarantee any termination.

There are other works in literature which have addressed the problem of exploration on dynamic graphs. The exploration problem on dynamic rings for the T-interval connected case is addressed in [28]. They have addressed the problem in two scenarios. In one scenario, the robot knows about all changes in the dynamic ring. In another case, the robot has no knowledge about the changes but the edges are δ-recurrent. They have extended their work in [29] and addressed the exploration problem on cactus graph when changes in the graph topology are known to the robot. There are other works like [30, 31] which address the exploration problem for general graphs in centralized environment when the change in the graph topology is already known. There are works [32, 33] which address the live or online version of the exploration problem in a distributed environment for periodically varying graphs. In this case, there are a finite number of carriers in the graph and an edge between two nodes exists only when a carrier moves from one node to the other. They have assumed that the movement of each carrier is periodic. A very recent work [34] studies exploration in time-varying graphs (including 1-interval connectivity) of arbitrary topology and investigates the number of robots that are necessary and sufficient to explore such graphs. There have been other papers that look at different problems such as gathering [35] and dispersion [36, 37] on dynamic graphs under 1-interval connectivity.

3. Network Model and Assumptions

Graph. We consider an undirected, 1-interval connected dynamic ring $\mathcal{R}$ of size n as considered in [22, 24]. As $\mathcal{R}$ is a ring, each node in $\mathcal{R}$ has two neighbours connected via two ports. The ring is anonymous, i.e., nodes are indistinguishable. We assume that the nodes are fixed, but the edges of $\mathcal{R}$ may change over time. More precisely, in any round at most one of the edges might be missing from $\mathcal{R}$. An adversary decides which edge is to be removed in a round. Such a dynamic ring is called a 1-interval connected ring [22, 38]. The adversary controls the edge deletion with the knowledge of the algorithm and current states and positions of the robots. However, for the randomized algorithms, the outcome of coin tosses made by each robot is not known to the adversary.

Robots. There are three robots R_1, R_2, and R_3 which explore $\mathcal{R}$. Each robot is equipped with a finite memory, say $O(\log n)$ bits, and computational capabilities. Each robot has a unique identifier (ID) and initially a robot only knows its own ID. Furthermore, we assume that the IDs are k-bit strings such that the length k is $O(1)$. It is sufficient to represent 3 distinct IDs with a constant number of bits. The ID of a robot is represented as $b_{k-1}b_{k-2}\cdots b_1b_0$. Initially, robots do not know the size of the ring (not even a bound on it). The robots do not share any common chirality, i.e., the *clockwise* and *anti-clockwise* directions for the robots may differ. During movement, at any node a robot can differentiate between the port through which it enters the node and the other port. All the robots execute the same protocol. Multiple robots can reside at a single node at the same time. When multiple robots are on the same node they can sense each other's presence and exchange information. The robots can move from one node to a neighboring node in some round if the corresponding edge is available in that round. A robot can *successfully move* towards a *fixed* direction if the corresponding adjacent edge is available in the dynamic ring; otherwise, if the edge is missing, the robot waits until the edge is available. At this waiting period, the robot can sense presence of other robots and can exchange information with them. For some of our algorithms, we make use of the *edge crossing detection* capability, i.e., two robots moving in opposite directions on the same edge in the same round can detect that they passed each other in that round and exchange information.

Time. For the majority of our work, we consider a synchronous system which progresses in time steps, called as rounds. In a single round, the sequence of operations executed is as follows: (i) each robot performs some local computation and decides whether or not to move from the current node and, in case it moves, the direction of that movement, (ii) the adversary removes at most one edge from the ring for this round, (iii) the robots execute their movements, if any, as long as the edge they wish to move over is present.

We also consider a semi-synchronous system in Section 7. In this setting, in every round, the adversary chooses a subset of robots to be active in that round and the remaining robots are put to sleep. The adversary is restricted to ensure that every robot is activated infinitely often. We consider the scenario when the *passive transport* of robots is allowed.[4] Informally, passive transport allows a robot to move along an edge even when it is asleep. In more detail, consider a round i where an awake robot wants to move through an edge e that the adversary removed in that round. Now, suppose the adversary subsequently puts the robot to sleep from round $i+1$ to some round j. If e is present again for the first time in some round $k : i+1 \leq k \leq j$, then the robot moves along the edge in round k even though it is asleep. In short, when a robot is awake, it can perform all the tasks like moving, deciding direction, communicating with co-located robots, etc. When a robot is asleep, it can only perform movement if it was already slated to move on that edge earlier while it was awake. Each robot does not have knowledge of whether it was asleep in the previous round or not.

4. Impossibility of Exploration with 2 Robots

In this section, we extend the impossibility results from [22] to the scenario where robots also have the edge crossing detection capability and access to randomness. First, we make a similar observation to Observation 2 from [22].

Observation 1. *The adversary can prevent two robots starting at different locations from meeting each other even if they have unlimited memory, common chirality, distinct known IDs, and the edge crossing detection capability when $n \geq 3$. When robots also have access to randomness, then it is required that $n \geq 5$.*

This observation is clear to see when robots do not have access to randomness by considering the following strategy of an adversary. If the two robots are on adjacent nodes, then the adversary removes the edge between those two nodes. If the robots are at distance two from each other but will move into the same node, then the adversary can remove the edge between the center node and one of the nodes with a robot on it. However, when robots have access to randomness, it is not clear how the robots will move. Thus, when robots have

[4]There exist other types of transport models not considered here; see [22] for details.

access to randomness, the adversary has the same above strategy for robots on adjacent nodes. If the robots are at a distance two from each other, the adversary removes one of the edges adjacent to the centre node between the nodes the robots are on. This is the reason why the minimum sizes of the ring are different for when randomness is involved and when it is not. If robots have access to randomness, then in order to apply the previous strategy, the ring should be of size at least 5. Now, using this observation, we are able to prove the following impossibility result, which is a more general version of that seen in [22]. Note that, the nature of the proof is similar to that of Theorem 1 in [22].

Theorem 4.1. *There does not exist any exploration algorithm with partial termination of anonymous rings of unknown size (≥ 3) by two robots, even when robots have distinct IDs, common chirality, the edge crossing detection capability, and when the scheduler is fully synchronous.*

Proof. Let there exist an algorithm $\mathcal{A}$ that achieves exploration with partial termination with two robots, say R_1 and R_2. Run this algorithm on a ring of size n and consider any adversary strategy that prevents R_1 and R_2 from ever meeting. From Observation 1, we know such a strategy is possible. Let us assume that, without loss of generality, R_1 terminates first after $T(n)$ rounds. We now construct a ring and adversary strategy such that $\mathcal{A}$ will fail, i.e., never achieve exploration.

Consider the ring of size $4T(n) + 4$ and place R_1 and R_2 on nodes that are at distance $2T(n)+2$ from each other, i.e., at opposite "ends" of the ring. Have each robot run $\mathcal{A}$ and let the adversary act such that R_1's local view at each time step is similar to its view when R_1 ran $\mathcal{A}$ on the ring of size n.[5] Thus, after $T(n)$ time steps (rounds), R_1 will terminate and by Observation 1, R_1 never came into contact with R_2. Subsequently, in each future round the adversary will remove any edge that R_2 may want to traverse and thus ensure that R_2 does not explore any more nodes. After $T(n)$ rounds, R_1 and R_2 would have collectively explored at most $2T(n)+2$ nodes and thus $\mathcal{A}$ fails to achieve exploration, which is a contradiction. □

We now provide a similar impossibility result when robots have access to randomness. Note that for this result, we are showing the impossibility of explicit termination and not partial termination. Also note that the proof is similar to that of the previous theorem with a few subtle but significant changes.

Theorem 4.2. *There does not exist any exploration algorithm with explicit termination of anonymous rings of unknown size (≥ 5) by two robots, even when robots have distinct IDs, common chirality, the edge crossing detection capability, access to randomness, and when the scheduler is fully synchronous.*

Proof. Let there exist an algorithm $\mathcal{A}$ that achieves exploration with explicit termination with two robots, say R_1 and R_2. Run this algorithm on a ring of size n. Define $T(n)$ to be the running time of $\mathcal{A}$ with high probability, i.e., with probability $\geq 1 - 1/n$, over all choices of randomness and all adversarial strategies that prevented R_1 and R_2 from meeting. By

[5]Here, the local view of robot R_1 is the set of all other robots currently co-located with it, which may be empty, and the set of ports R_1 sees on the node.

Observation 1, we know that such strategies exist. Define an execution j of $\mathcal{A}$ as a vector of all the random choices, information communicated, local computations, and movements performed by both robots until termination. Define $V_{R_1}(i,j)$ as the vector of local views of R_1 for all rounds up to round i for some execution j of $\mathcal{A}$. Define $V_{R_1}(j)$ as the vector of local views of R_1 for all rounds up to termination for some execution j of $\mathcal{A}$. Define $\mathcal{V}_{\mathcal{R}_\infty}$ as the set of all $V_{R_1}(j)$ across all executions j of $\mathcal{A}$ for all possible choices of randomness and all possible adversarial strategies subject to the condition that R_1 and R_2 never meet. We now construct a ring and adversary strategy such that $\mathcal{A}$ will fail, i.e., never achieve exploration.

Consider the ring of size $4T(n)+4$ and place R_1 and R_2 on nodes that are at distance $2T(n)+2$ from each other. Now, the adversary focuses on R_1 and acts so that the local view of R_1 in round i, $V_{R_1}(i,j)$ will always belong to $\mathcal{V}_{\mathcal{R}_\infty}$ for the current execution j of $\mathcal{A}$. This is possible because R_1 and R_2 never meet in any of the executions we considered and so the local view of R_1 in a given round is influenced only by which edge has been removed in that round. Furthermore, since R_1 and R_2 are located at distance $2T(n)+2$ away from each other, R_2 will never meet R_1 within $T(n)$ rounds, so we can safely ignore how R_2 behaves. Now, after $T(n)$ rounds, R_1 will terminate.

We subsequently have the adversary focus on R_2 and trap the robot within a strip of two nodes. Consider two adjacent nodes u and v and let R_2 be present on one of them. The adversary always removes the edge from the node that is not the edge between u and v. Thus, R_2 will either terminate or indefinitely move between these two nodes.

In the course of the execution, R_1 could explore at most $T(n)+1$ nodes before it terminates and R_2 could similarly explore at most those many nodes before being trapped. Thus, no more than $2T(n)+2$ nodes could ever be explored, resulting in $\mathcal{A}$ failing. This is a contradiction and thus we see that no such $\mathcal{A}$ can exist. $\square$

The reason the above impossibility result works for explicit termination but not for partial termination is that when we allow robots to use randomness to make choices, it no longer becomes clear which robot terminates first. This is not an issue for explicit termination because we leverage the fact that both robots eventually terminate and consider that running time. However, for partial termination, when utilizing the adversary to mimic the local view of one of the robots, it is unclear which robot we should focus on initially. And since we cannot focus on both simultaneously, if we pick the incorrect robot initially, we cannot guarantee that it will eventually terminate, and thus cannot move on to focus on the other robot.

Note that, the above proof strategy and observation can be extended to multiple robots when the adversary is made more powerful. For this, let us define an adversarial strategy, called *t-adversary* which can remove at most t edges in the graph in a given round. Note that, Observation 1 holds for $t+1$ robots starting at unique positions on a ring of size at least $t+2$ when robots do not have access to randomness and of size at least $2t+3$ when they do. Thus, we can use a similar proof strategy to prove the following theorems.[6]

[6] We briefly recap the strategy. First, run a supposed exploration algorithm $\mathcal{A}$ on a ring of size n that terminates in $T(n)$ rounds. Subsequently, construct a ring of size $(t+1)(2T(n)+2)$ and place the $t+1$ robots equidistant from each other. Now, for partial termination (explicit termination), run $\mathcal{A}$ on this larger ring and simulate the execution on the smaller ring for 1 robot (t robots) until it settles down and subsequently

Theorem 4.3. *There does not exist any exploration algorithm with partial termination of anonymous rings of unknown size at least $t+2$ by $t+1$ robots in the presence of a t-adversary, even when robots have distinct IDs, common chirality, the edge crossing detection capability, and when the scheduler is fully synchronous.*

Theorem 4.4. *There does not exist any exploration algorithm with explicit termination of anonymous rings of unknown size at least $2t + 3$ by $t + 1$ robots in the presence of a t-adversary, even when robots have distinct IDs, common chirality, the edge crossing detection capability, access to randomness, and when the scheduler is fully synchronous.*

5. Deterministic Exploration with 3 Robots

In Section 4, we have shown that it is impossible to explore an anonymous dynamic ring of unknown size with two robots and achieve explicit termination. In this section, we present a deterministic solution for this exploration problem using three robots. We assume that each robot has a unique ID which is not known to the other robots unless they meet. For simplicity of the algorithm, let us first assume that the length of the IDs of the robots are same. Later we show how the algorithm works for variable length IDs in Section 5.3. We further assume that when two robots cross an edge (from opposite directions) in the same round, they *sense* each other and that the meeting happened.[7] Note that this edge crossing detection assumption does not help two robots (with unique IDs) solve the exploration problem (see Section 4).

Let us now describe the algorithm. It works in four stages: (Stage 1) first meeting of (any) two robots, (Stage 2) second meeting of two robots, (Stage 3) exploration detection, and (Stage 4) termination.

Stage 1 ensures the first meeting of any two robots at some node or via edge crossing in the ring. For this, we need to make sure that at least two of them move in opposite directions; otherwise, if all the three robots move in the same direction at the same speed, they may never meet even if the adversary never deletes any edge. Thus, we have to break this symmetry deterministically. For this, each robot moves based on the bit string of its ID. Each robot moves in phases and each phase consists of several rounds. More precisely, the number of rounds in the i-th phase is 2^i. Without loss of generality, say that a robot moves in what it considers the clockwise (right) direction in phase i when $b_{i \bmod k} = 0$. When $b_{i \bmod k} = 1$, the robot moves in the other (left) direction. The first stage ends when at least two robots meet. Let us mark or name these two robots A and B, where the larger ID one is A and the other one is B. Note that, the third robot may not know about this meeting and hence is unaware of the end of the first stage. Let us call the third robot as C.[8] If these three robots are present at the same time (at some node) then the smallest ID robot gets named C.[9] Notice that if two or three robots are positioned at the same node initially then

trap the remaining t robots (1 robot) on already explored nodes. The total explored number of nodes will fall short of the total size of the ring and hence $\mathcal{A}$ is incorrect.

[7]Here by 'sense' we mean the two robots can detect the edge crossing and can exchange information including IDs.

[8]The third robot gets named C only after it meets either A or B at the end of Stage 2.

[9]Note that this situation can only occur initially.

the algorithm starts from Stage 2.

Then **Stage 2** starts (which is known to at least A and B). The robots A and B start moving in opposite directions from the meeting point (node or nodes adjacent to the edge they crossed by each other) from Stage 1 and never change their directions until they terminate the algorithm. A and B each maintain a counter which counts the number of steps the robot successfully moves. Furthermore, each robot stores the ID of the other. Note that a robot cannot move in a particular direction in a round if the corresponding edge is missing (i.e., deleted by the adversary). Each of the robots continues to move until one of them meets the third robot. Stage 2 ends when either of A and B meets the third robot, which subsequently gets named C. Without loss of generality, assume that A and C meet. Then A shares the following stored information with C: ID of B, the direction of B's movement and the number of steps A has successfully moved after Stage 1.[10] C stores all this information. A and C also store each other's IDs.

Then **Stage 3** starts, which ensures the completion of the ring exploration by at least two robots. This can happen in two ways. (I) if A and B meet (again) then it is guaranteed that exploration of the ring is complete. This scenario is depicted in Figure 1. (II) The adversary

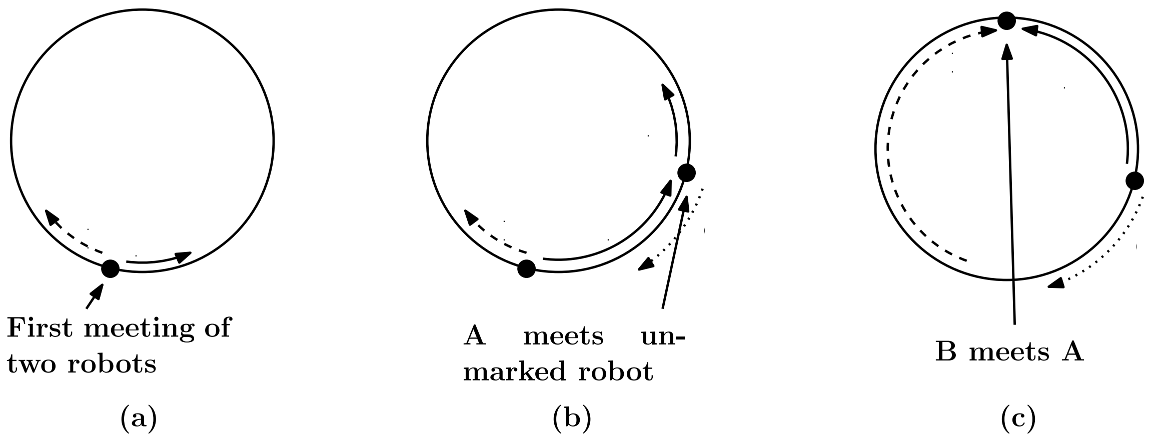


Figure 1: Continuous, dashed and dotted lines shows the movement of robot A, B and C respectively. (a) Shows the scenario when two unmarked robot meet and get marked as A and B, i.e. Stage 1 ends and Stage 2 starts. (b) Shows the scenario when A and the unmarked robot meet and the unmarked robot gets marked as C, i.e. Stage 2 ends Stage 3 starts, (c) Shows the scenario when A and B meets, i.e. Stage 3 ends.

can prevent the meeting of A and B by removing an edge between them. Recall that A and B are moving in opposite directions. Eventually these two robots will reach two adjacent nodes and may wait for the (missing) edge to move. In this scenario, exploration is completed but A and B do not know this as n is unknown. If the adversary does not remove the edge in one round, then A and B will meet. Therefore, the adversary will need to remove the edge indefinitely. In this situation, robot C is used to determine the completion of exploration.

[10]Note that even if A and C do not have shared chirality, the direction of B can be conveyed as follows. Depending on how A and C meet, C will immediately know the direction A moves in or can take a round to understand this based on how A and C both move in their "clockwise" direction and see if they moved to the same node or not. Once C determines the naming mechanism A uses for directions, C can understand exactly which direction B is moving in.

From the meeting point of A and C in Stage 2, robot C starts moving towards the opposite direction of A (i.e., in the same direction of B) and A continues moving in its fixed direction.

Robot B does not know that A and C met in Stage 2, and continues to move in its fixed direction. Robot C moves towards B until it catches B. Subsequently, C changes its direction and move towards A until it catches A. C then repeats this process and moves back to B. Essentially, C performs a zig-zag movement between A and B, and checks if the distance (i.e., the hop distance) from A to B, and B to A are the same. For this, the robot C maintains two variables $AtoB$ and $BtoA$. $AtoB$ stores the number of successful steps (moves) towards B, starting from A until it meets B, and $BtoA$ stores a similar number.

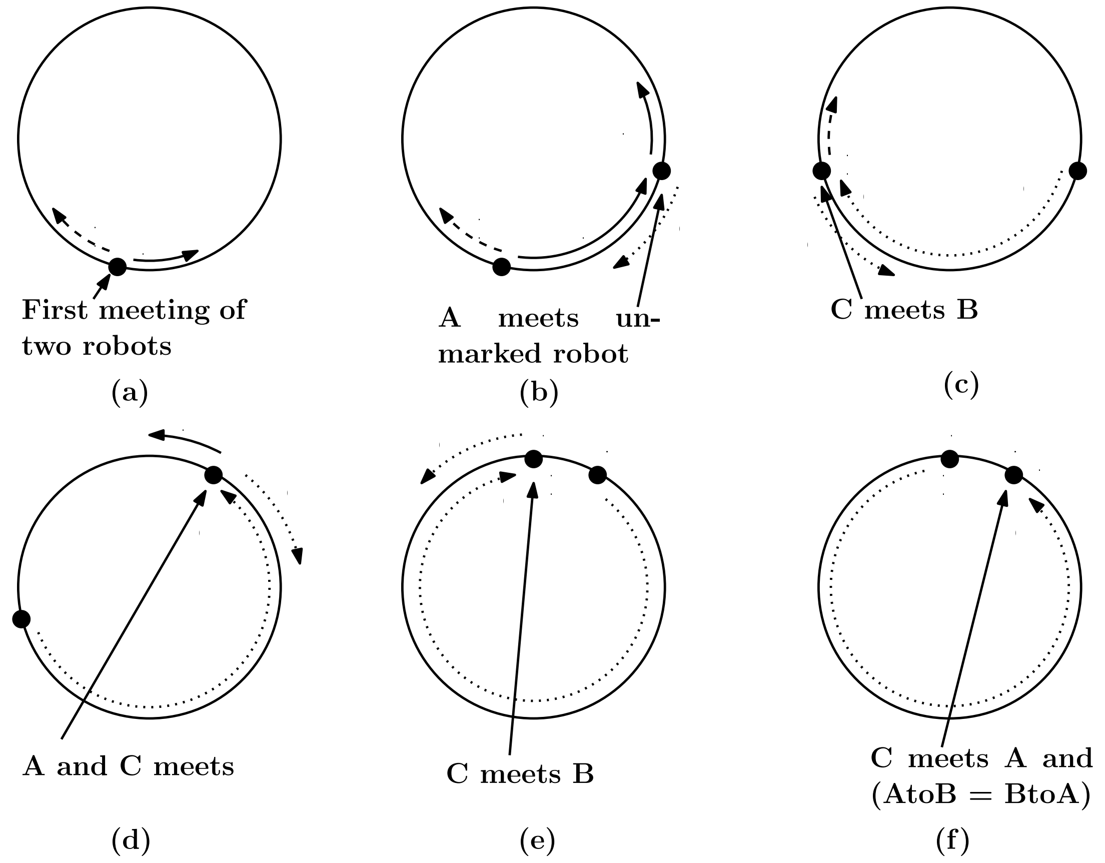


Figure 2: Continuous, dashed, and dotted lines show the movements of robots A, B, and C respectively. (a) Shows the scenario when two robots meet for the first time and get marked as A and B, i.e., Stage 1 ends and Stage 2 starts. (b) Shows the scenario when A and the third unmarked robot meet and the unmarked robot gets marked as C, i.e., Stage 2 ends Stage 3 starts. (c) Shows the scenario when C meets B and finds $AtoB \neq BtoA$. (d) Shows the scenario when C again meets A at node v and finds $AtoB \neq BtoA$. (e) Shows the scenario when C again meets B at node u and finds $AtoB \neq BtoA$. (f) Shows the scaenario when C again meets A at node v and finds $AtoB = BtoA$ and Stage 3 ends. u and v are two consecutive nodes in $\mathcal{R}$. As nodes in $\mathcal{R}$ are not identifiable, the names of nodes are used here only for the ease of understanding.

When these two distances are equal, i.e., $AtoB = BtoA$, the algorithm determines that exploration is complete, as this condition implies that A and B lie on two adjacent nodes and the edge between those two nodes has been removed by the adversary. Thus, C has explored the entire graph. This scenario is depicted in Figure 2.

Therefore, either A and B meet and detect that exploration is completed, or C deduces the completion of exploration from the hop distance counts. In the latter case, C would be co-located in a node with either A or B and can thus inform that robot about the completion of exploration. Thus, at least two robots detect the completion of exploration, but the third robot may be unaware of this. Then we begin Stage 4 to ensure that all robots are made aware of the completion of exploration and can terminate.

In **Stage 4**, the two robots which detected the completion of exploration move in order to inform the third robot about this, and all robots terminate (to guarantee the explicit termination). Recall that the robots A and B maintained a counter of their successful moves starting from their meeting in Stage 2. If A and B meet at some node again, then the sum of their counters is exactly n. If they meet by crossing each other, then the sum of their counters is exactly $n+1$. If they do not meet but the scenario from Stage 3 plays out, then C and one of the two robots knows the value of $n-1$. In any case, there are two robots which know that exploration is completed and know the value of n at some node. Then these two robots start moving in opposite directions to each other for at most n rounds. When one of them meets the third robot, it informs the third robot about the completion of exploration and they both terminate. The other robot also terminates after n rounds (if it does not meet the third robot). After n rounds, at least one of the robots (which detects the completion of exploration) meets the third robot and informs it about the completion of exploration.

The formal pseudocode of the algorithm is given in Algorithm 1, which is run by all robots in parallel. Each robot maintains the following variables:

- *size*: current estimate on size of the ring (initialized to ∞).
- *step*: stores the number of successful steps or moves of a robot.
- *mark*: takes value in $\{A, B, C\}$. Initialized to NIL for all the robots.

Different events are defined below, which may occur when two or more robots meet during the execution of the algorithm.

- *meetSmall:* when a robot meets with a smaller ID robot and *mark* $= NIL$ for both the robots.
- *meetLarge:* when a robot meets with a larger ID robot and *mark* $= NIL$ for both the robots.
- *meetMark:* when a robot, whose *mark* $= NIL$, meets another robot whose *mark* $\neq NIL$.
- *meetUMark:* when a robot, whose *mark* $\neq NIL$, meets another robot whose *mark* $= NIL$.
- *meetX:* when a robot, whose *mark* $\neq NIL$, meets another robot whose *mark* $= X$ for $X \in \{A, B, C\}$.
- *meetTer:* when a robot meets another robot which is executing startTermination procedure.

Notice that when two robots meet and the *meetSmall* event occurs for one robot, the *meet-Large* event occurs for the other robot at the same time. In the pseudocode, the following functions are used:

- MOVE(dir): By executing this function a robot moves one step towards the direction specified in dir. Here value of dir can be $left$ or $right$.
- GETMARK(): When two robots meet, this function returns the value stored in variable $mark$ of the other robot.
- GETSTEPS(): When two robots meet, this function returns the value stored in variable *step* of the other robot.
- ASKTERMINATE(): When two robots meet, using this function one robot sends a signal to the other robot to start termination.
- RECTERMINATE(): When two robots meet, this function detects whether a robot has received a signal to start termination or not. This function returns 1 if signal to start termination is received.

In a ring there are two possible directions to move in at each node. A robot chooses a direction arbitrarily and calls that direction as *left (anti-clockwise)* and the other direction as *right (clockwise)*. Since the robots do not share common chirality, the left and right directions of a robot may differ from the other robots (initially). When two robots meet, they can decide upon a common set of directions and then share a common chirality.

Each robot starts executing the algorithm EXPLORE-DYNAMIC-RING-3-ROBOTS by executing the pseudocode described in Algorithm 1. Lines 1 to 7 of Algorithm 1 handle the scenario when all three robots are co-located on a single node. In this case, the robot with the largest ID, second largest ID, and smallest ID become robot A, B, and C respectively. Lines 8 to 12 of Algorithm 1 handle the scenario when two robots are co-located on the same node. In this case, the robot with the larger ID becomes robot A and the other one becomes robot B. The case when all three robots are placed in different nodes is handled in lines 13 to 20. In this scenario, each robot moves following the bit pattern in its respective ID by executing the EXPLORE procedure.

The procedure EXPLORE is described in Algorithm 2. This procedure takes the direction and an integer i as inputs. While executing this procedure, a robot tries to move for 2^i steps in the inputted direction unless a meeting event occurs. The robot becomes either robot A, B, or C depending on if the event is *meetSmall*, *meetLarge*, or *meetMark* respectively.

Whenever two robots meet or two robots are co-located, Algorithm 1 or Procedure 2 calls the procedure BEROBOTAB or BEROBOTC. Procedure 3 gives the pseudocode of BEROBOTAB, which is followed by a robot when it is either robot A or robot B. The input parameter indicates whether it is robot A or robot B. In lines 4 to 10 of Algorithm 3, robots A and B move in directions *left* and *right* respectively and increases the number of steps by one for each successful movement. In lines 11 to 25 of Algorithm 3, when a *meetB* or *meetA* event occurs, the robots exchange the number of steps they have moved and determine the size of the ring and start the termination stage through Procedure STARTTERMINATION. Then, lines 26 to 33 handle the scenarios when a *meetC* or *meetTer* event occurs. When *meetC* with recTerminate() = 1 occurs, both robots start the termination stage. Note that when event *meetC* with recTerminate() = 0 or *meetUMark* occurs, neither robot A nor robot B needs to do anything. If event *meetTer* occurs, the robot terminates.

BeRobotC described in Procedure 4 gives the pseudocode to be followed by a robot when it is robot C. Robot C does the zigzag traversal between robot A and robot B until the number of steps in two consecutive A to B and B to A traversals are equal, or it meets a robot executing the termination stage. Robot C moves in one direction and whenever a *meetB* or *meetA* event occurs, it compares the number of steps covered in this traversal with the previous traversal. Lines 1 to 28 of Procedure 4 describes these steps.

startTermination described in Procedure 4 describes the pseudocode to be followed by a robot when it is executing termination stage. In this stage the robot tries to move for n (size of the ring) steps. If it meets any other robot while moving, it terminates after asking the other robot to terminate.

Algorithm 1 Explore-Dynamic-Ring-3-Robots

```
 1: if all three robots are on the same node then
 2:     if robot's ID is largest then
 3:         BeRobotAB($A$)
 4:     else if robot's ID is 2nd largest then
 5:         BeRobotAB($B$)
 6:     else
 7:         BeRobotC()
 8: else if any two robots are on the same node then
 9:     if robot's ID is larger then
10:         BeRobotAB($A$)
11:     else
12:         BeRobotAB($B$)
13: else
14:     $i := 0$
15:     while (1) do
16:         if $b_{i \bmod k} = 0$ then
17:             Explore($left$, i)
18:         else if $b_{i \bmod k} = 1$ then
19:             Explore($right$, i)
20:         $i := i + 1$
```

Procedure 2 Explore(dir, i)

```
1: for $tstep = 0$ to $(2^i - 1)$ do
2:     if meetSmall then
3:         BeRobotAB($A$)
4:     else if meetLarge then
5:         BeRobotAB($B$)
6:     else if meetMark then
7:         BeRobotC()
8:     Move($dir$)
```

Procedure 3 BeRobotAB($recMark$)

```
step := 0
if $mark = NIL$ then
    $mark := recMark$
while (1) do
    if $mark = A$ then
        Move($left$)
    else
        Move($right$)
    if move successful then
        step := step+1
    if $meetB$ or $meetA$ then
        if $mark = A$ then
            recStep := getSetps()
            if meeting on a node then
                size := step + recStep
            else if meeting by edge crossing then
                size := step + recStep - 1
            startTermination($A$, $left$, size)
        else
            recStep := getSetps()
            if meeting on a node then
                size := step + recStep
            else if meeting by edge crossing then
                size := step + recStep - 1
            startTermination($B$, $right$, size)
    else if $meetC$ and recTerminate() = 1 then
        size := getSteps() + 1
        if $mark = A$ then
            startTermination($A$, $left$, size)
        else if $mark = B$ then
            startTermination($B$, $right$, size)
    else if $meetTer$ then
        $terminate$
```

Procedure 4 BeRobotC()

```
AtoB := 0
BtoA := 0
recMark := getMark()
mark := C
if recMark = A then
    dir := right
else if recMark = B then
    dir := left
while (1) do
    Move(dir)
    if move successful then
        step := step+1
    if meetB then
        AtoB := step
        if AtoB ≠ BtoA then
            step := 0
        else
            size := step + 1
            askTerminate()
            startTermination(C, left, size)
    else if meetA then
        BtoA := step
        if AtoB ≠ BtoA then
            step := 0
        else
            size := step + 1
            askTerminate()
            startTermination(C, right, size)
    else if meetTer then
        terminate
```

Procedure 5 startTermination(*mark*, *dir*, size)

```
Ttime := size
while (1) do
    Move(dir)
    Ttime := Ttime-1
    if Ttime = 0 then
        terminate
    else if (meetA or meetB or meetC) then
        terminate
```

5.1. Correctness and Time Complexity Analysis

We first discuss the correctness of the algorithm in the following lemma.

Lemma 5.1. *The proposed algorithm correctly explores the dynamic ring and guarantees explicit termination.*

Proof. We first show that by the end of Stage 3, the ring is explored (by at least two robots). Stage 3 ends when one of the following two cases occurs: (I) robots A and B meet again or (II) robot C meets A and finds $AtoB = BtoA$ (after zig-zag movement) or C meets B and finds $AtoB = BtoA$.

Case I: Robots A and B meet again after their first meeting in Stage 1. Since they move in opposite directions and never change directions after their first meeting, it is obvious that when they meet again, exploration is completed. The robots can also calculate the value of n when they meet again.

Case II: Robot C meets either of A and B, and learns that $AtoB = BtoA$. This implies that robot C has traversed the same number of steps in two consecutive zig-zag movements. This scenario is only possible if both A and B are trying to traverse the same edge from adjacent nodes, since the adversary can remove only one edge at a time. Thus, when robot C determines that $AtoB = BtoA$ after two consecutive zig-zag movements, it is guaranteed that C has explored all nodes in the ring and the size of the ring is $AtoB + 1$ (equivalently, $BtoA + 1$).

Thus, in both cases, at least two robots detect that exploration is completed by the end of Stage 3. Moreover, the robots which detect this also learn the size of the ring n. Thus, in the termination stage (Stage 4), these two robots, which detected the completion of exploration, start moving in two opposite directions for at most n rounds and terminate. As these two robots start moving in opposite directions from the same meeting point, it follows from the proof of Lemma 5.2 (below) that after n rounds, at least one of them meets the third robot. So the third robot also gets the information about the completion of exploration and terminates. Thus, explicit termination is guaranteed at the end of Stage 4. □

Lemma 5.2. *If two among the three robots move in opposite directions in a dynamic ring of size n, then at least two of them meet in at most $n - 2$ rounds.*

Proof. If any two among these three robots are initially located at the same node in the ring, then this lemma holds trivially.

We assume that the three robots are initially located on three different nodes in the ring. Recall that a robot always tries to move in some specified direction as long as the corresponding edge is available, i.e., it does not voluntarily remain stationary.

As we are considering a dynamic ring, there are two robots which are moving in the same direction. Let us assume two robots R_1 and R_2 are moving in the same direction. The other robot R_3 is moving in a different direction. Recall that the adversary can remove only one edge in each round. Thus, the distance between R_1, R_3 and the distance between R_2, R_3 decrease by at least one in each round. All three robots are located on three different nodes and size of the ring is n. Thus, after $(n-2)$ rounds, if R_3 does not meet either of R_1, R_2, the distance between them is 1 and R_1, R_2 are co-located on the same node. Hence the lemma holds. □

Let us now analyze the time complexity of the exploration algorithm. We calculate the time taken in each stage of the algorithm.

Lemma 5.3. *In Stage 1 of the proposed algorithm, there exists a phase $i \in [(j-1)k, jk-1]$ for $j \geq 1$ when at least one robot moves in the direction opposite to the direction followed by other two robots, where k is the length of the ID bit-string of the robots.*

Proof. Recall that in the i^{th} phase a robot moves in some direction for 2^i rounds. It then changes its direction of movement in the next phase iff the next bit in its ID is different. Consider the scenario where all robots move in the same direction starting from the first phase (otherwise the lemma is trivially true). There are two scenarios to consider. Either all robots share the same chirality or they do not.

Consider the scenario where all robots share the same chirality. Since the IDs of the robots are different, at least one bit in the ID of each pair of the robots are different. Hence, there will be at least two phases in $(j-1)k$ to $jk-1$ for $j \geq 1$ when one of the robots moves in a direction opposite to the direction followed by the other two robots.

Consider the scenario where robots do not share the same chirality. Since chiralities can be different, two robots with different chiralities can have IDs that are complementary (e.g., 000 and 111) and thus move in the same direction in all phases. However, since all IDs are different, the third robot's ID will be such that there will exist at least one phase where one of the robots moves in a different direction from the other two. □

Lemma 5.4. *Stage 1 of the proposed algorithm finishes in at most $n + n \cdot 2^k$ rounds, where k is the length of the ID bit-string of the robots.*

Proof. We show that there exists a phase $i \in [0, j(k-1)]$ when at least two robots meet in Stage 1, where k is the length of the IDs of the robots and j is some positive integer. It follows from Lemma 5.3 that there exists a phase $i \in [0, k-1]$ when at least two robots move in the opposite directions to each other. The number of rounds in that phase is 2^i. If $2^i \geq n-2$ then it follows from Lemma 5.2 that any two robots meet. However, it might be the case that $2^i < n-2$ for that i in $[0, k-1]$. Then according to our algorithm (see Stage 1), these two robots (again) move in the opposite directions in each of the phases $j(k-1)+i$ for $j = 1, 2, \ldots,$ and hence Stage 1 finishes when $2^{j(k-1)} \leq n-3$ and $2^{j(k-1)+i} \geq n-2$ for some positive integer j. Thus, Stage 1 takes at most $\sum_{t=0}^{j(k-1)+i} 2^t$ rounds. The sum is bounded above by $(n + n \cdot 2^i)$, since $2^{j(k-1)} \leq n-3$. Therefore, Stage 1 of the proposed algorithm finishes in at most $n + n \cdot 2^k$ rounds, since $i < k$. □

Lemma 5.5. *Stage 2 of the proposed algorithm finishes in at most n rounds.*

Proof. Stage 2 finishes when any of the two robots A and B meets C after their (A and B) first meeting in Stage 1. Note that A and B start moving in opposite directions after Stage 1 ends. Thus it follows from the proof of Lemma 5.2 that one of them meets C in at most $n-2$ rounds. □

Lemma 5.6. *Stage 3 of the proposed algorithm finishes in at most $4n$ rounds.*

Proof. Stage 3 finishes when either (I) A and B meet again (which ensures that exploration is completed), or (II) robot C detects the completion of exploration from the step-counts of its zig-zag movement, i.e., when $AtoB = BtoA$. Since robot A and robot B move in opposite directions over the ring, they can meet in at most n rounds if there are no edge deletions. However, the adversary may remove edges to prevent or delay their meeting. Despite that, the two robots reach two adjacent nodes of an edge in $n-1$ rounds (since they are moving in opposite directions and the adversary can remove at most one edge at a time).

Now we claim that if A and B do not meet in $4n$ rounds, then (II) happens, i.e., the robot C detects the completion of exploration within this $4n$ rounds. The robot C performs zig-zag movement between A and B until $AtoB = BtoA$. Notice that $AtoB$ will be equal to $BtoA$ when A and B are at the adjacent nodes of an edge. This takes at most $n-1$ rounds, as explained before. Suppose the robot C changes its direction of movement just before A and B reach the two adjacent nodes. Suppose C was at the side of B at this point. Then it will take another $n-1$ rounds (at most) for C to reach the other endpoint i.e., the adjacent node where A is waiting. Then C takes another $n-1$ rounds to reach the other endpoint B (and then the count $AtoB$ becomes $n-1$). Finally, it requires another $n-1$ rounds for C to reach A and get the condition $AtoB = BtoA$ satisfied. Therefore, C detects the completion of exploration from the step-counts of its zig-zag movement in $4n$ rounds.

Thus, Stage 3 finishes in at most $4n$ rounds. □

Lemma 5.7. *Stage 4 of the proposed algorithm finishes in at most n rounds.*

Proof. In Stage 4, two robots, which have detected the completion of exploration, move for at most n rounds and terminate. Since these two robots move in opposite directions, it follows from Lemma 5.2 that at least one of them meets the third robot in $n-2$ rounds and they both terminate. Hence Stage 4 takes at most n rounds. □

Now we state the main result of this section.

Theorem 5.8. *The proposed algorithm correctly explores a 1-interval connected dynamic (anonymous) ring of size n in $O(n + n \cdot 2^k)$ rounds with 3 robots where each robot has a unique ID of length k bits, the robots have no knowledge of n, and the robots do not have common chirality.*

Proof. The correctness of the algorithm follows from Lemma 5.1.

The running time of the algorithm follows from the time complexity analysis of the four stages in Lemmas 5.4, 5.5, 5.6 and 5.7. Thus by summing up the individual runtimes, we get the time complexity as $(n + n \cdot 2^k) + n + 4n + n = 7n + n \cdot 2^k$.

Hence, the proposed algorithm explores a dynamic ring of size n with three robots in $(7n + n \cdot 2^k)$ rounds. □

Corollary 5.8.1. *There exists an algorithm which explores a 1-interval connected dynamic (anonymous) ring of size n in $O(n)$ rounds with 3 robots having unique IDs of length $O(1)$ bits, edge crossing detection, and without the knowledge of n and without common chirality.*

5.2. Simulation Results

We perform an experimental evaluation and highlight the effectiveness and efficiency of our algorithm in dynamic rings for different parameter ranges. In particular, we evaluate the performance of our algorithm by computing the running time for different sizes of the dynamic ring (i.e., number of nodes in the ring) and also for different ID lengths of the robots. In the simulations, we assume that the robots are placed at random nodes in the beginning. Furthermore, each robot decides its initial direction of movement randomly, i.e., each robot decides clockwise (right) or anti-clockwise (left) direction randomly.

We assume an adversary determines the dynamic ring in each round. In particular, the following four different adversarial strategies are considered for the simulations.

- *Random Edge Deleted (RED):* In each round, the adversary randomly selects an edge in the ring and deletes it. The previously deleted edge gets added back to the ring.
- *Same Edge Deleted (SED):* The adversary randomly selects an edge in the ring and keeps the edge deleted throughout the execution.
- *Random Robot Blocking (RRB):* In each round, the adversary targets a random robot, and blocks the movement of the robot in that round. This is done by deleting the edge through which the robot decides to move in that particular round.
- *Same Robot Blocking (SRB):* The adversary randomly selects a robot and blocks the movement of the robot throughout the execution by deleting appropriate edges. That is, the robot is not allowed to move from its initial position.

The robots have no knowledge about the adversarial strategies, but the adversary knows the robots' current positions including the edges through which the robots decide to move. Thus the above adversarial strategies are adaptive. In all the cases, the dynamic ring remains connected throughout the execution.

Varying the Size of the Ring (Figure 3): In this experiment, we consider a dynamic ring of five different sizes, i.e., $n = 20000, 40000, 60000$, 80000, and 100000. The robots have IDs of length 3; in fact, the ID-bits are 100, 101 and 111 for the 3 robots. We run the algorithm 5 times for each value of n and count the average number of rounds taken to explore the ring. We plot the results in Figure 3, where the x-axis represents the ring size n and the y-axis represents the actual number of rounds taken to explore the ring in the experimental set up. Observe that the time or the number of rounds to explore the ring increases when the size of the ring n increases. However, in all the cases, the running time of our algorithm is bounded by $5n$ for all the different adversarial strategies. The running time is less than $3n$ for the strategies RED and RRB. This shows that the simulation results outperform our theoretically proven time bound– $7n + n \cdot 2^k$, where k is the ID length of a robot.

Varying the ID Length of the Robots (Figure 4): In this experiment, we consider a ring of size 32768. The length of the IDs of each robot varies from 3 to 15 (notice that $\log n = 15$ as $32768 = 2^{15}$). In particular, the ID lengths of the robots considered for the simulations are 3, 6, 9, 12, and 15. The ID-bits are generated randomly. In this case also, we run the algorithm 5 times for each different ID length and count the average number of rounds taken to explore the ring. Again we perform the simulations for the four adversarial strategies. We plot these simulation results in Figure 4, where the x-axis represents the ID length and the y-axis represents the rounds taken by the algorithm to explore the ring. Observe that, in all the cases, the running time of the algorithm is bounded above by $5n$,

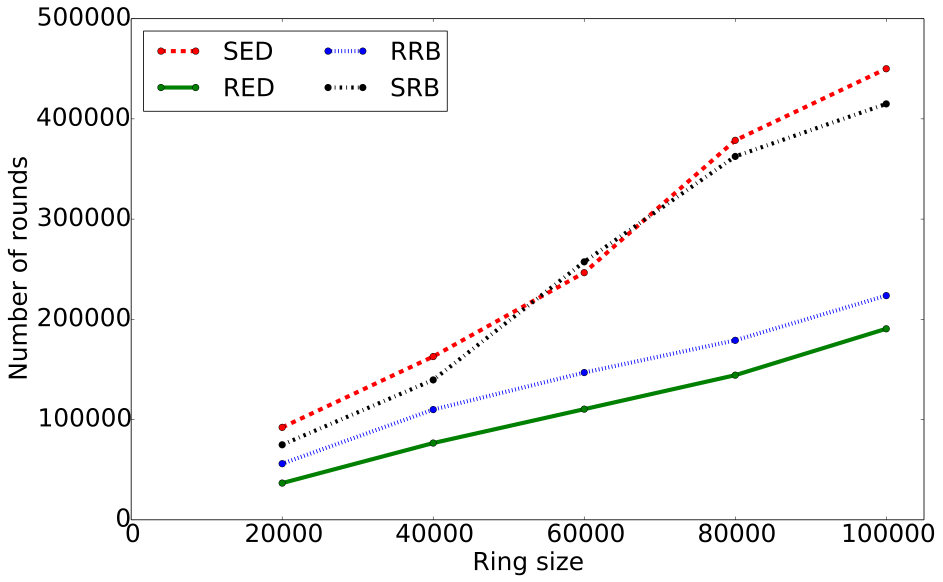


Figure 3: Varying n, the size of the ring.

where n is the ring size. In fact, the running time is less than $3n$ for the strategies RED and RRB. In this case also, the simulation results outperform our theoretically proven time bound.

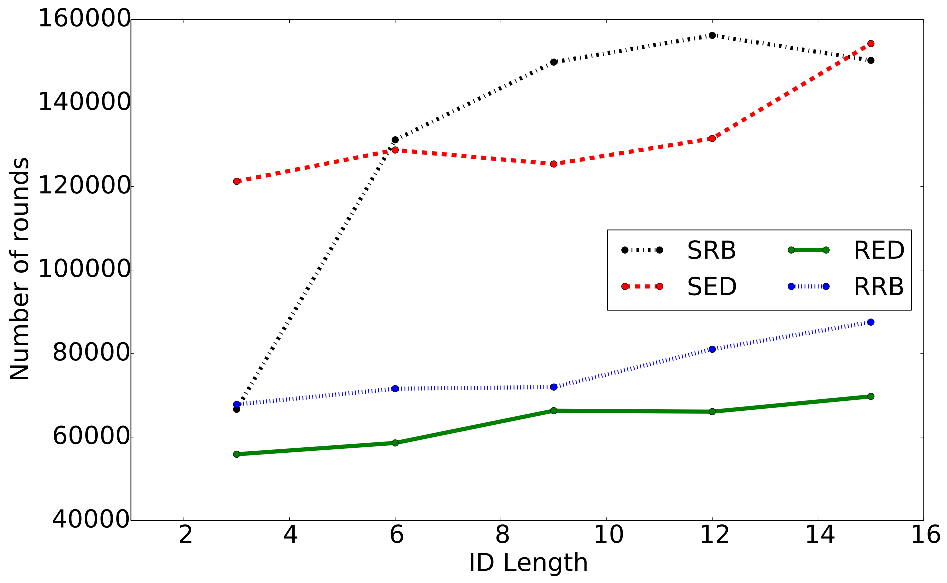


Figure 4: Varying the ID length of the robots.

5.3. Variable Length Identifier

The proposed algorithm assumes that the ID-length of the robots are the same. We now discuss how to remove the assumption. Recall that in Stage 1 of the algorithm, two robots are required to meet in order to move to the next stage. Lemmas 5.3 and 5.4 show that to ensure this meeting, there should be one phase when two robots need to move in opposite directions for $(n-2)$ rounds, where n is the size of the ring. When the IDs are distinct but with the same length for all the robots, the scheme where each robot moves following the bit patterns generated by repeated concatenation of the IDs works for our purpose. If the IDs are of different lengths, then this strategy may not work. For example, consider the scenario when the IDs of the robots are 101, 101101, and 101101101. Then the bit-strings generated by concatenating the IDs repeatedly are identical (although the IDs are distinct). In this case, if the robots are located at different nodes and have common chirality, then they will never meet even in a static ring. Thus we need to make sure that the generated ID-strings are distinct for the robots.

One modification in the above bit-string generation process overcomes this issue. As the length of IDs are variable, we assume that the length of the ID of the j^{th} robot is k_j $(j = 1, 2, 3)$. In the algorithm, in Stage 1, each robot moves in phases. In phase i, if the i^{th} bit of k_j is 0 robot j moves towards the clockwise (right) direction for 2^i rounds. It moves towards the anti-clockwise (left) direction if the i^{th} bit of k_j is 1. After processing the k_j^{th} bit, the robot repeats the same bit pattern, i.e., at $i = k_j + 1$ the j^{th} robot moves according to the 0^{th} bit in its ID and so on. In other words, the robots concatenate their ID bits to generate bit-strings which are different from each other.

We modify the bit-string generation process as follows. The j^{th} robot appends k_j number of 0 bits with its ID on the right side. Let us call this combined ID consisting of the original ID followed by the k_j number of 0s as the padded ID of the j^{th} robot R_j, denoted by $padID(R_j)$[11]. Then each robot generates a bit-string by repeatedly concatenating this $padID(R_j)$ and follows that generated bit-string to decide the direction of movement in Stage 1. We name this 0-padding scheme as Variable-ID-Padding. Notice that if the length of two IDs are different, then the ID-lengths will break the symmetry in the generated bit-string of the robots, since the number of padded 0s are different. If ID lengths of two robots are the same, then we show that the generated IDs will differ in at least one bit position. This ensures that the bit patterns followed by two robots differ in at least one bit. Below we prove this claim.

Lemma 5.9. *Let the two robots with IDs R_1, R_2 have the generated padded IDs $padID(R_1)$, $padID(R_2)$ respectively using the* Variable-ID-Padding *scheme. Let $l_1 = len(padID(R_1))$ and $l_2 = len(padID(R_2))$ be the length of the padded IDs. Let X and Y be two bit-strings of the same length l^*, where $l^* = l.c.m.(l_1, l_2)$[12], generated by l^*/l_1 and l^*/l_2 number of concatenations of $padID(R_1)$ and $padID(R_2)$ respectively.*[13] *Then, X and Y differ in at least one bit.*

Proof. Let $k_1 = len(R_1), k_2 = len(R_2)$. When $k_1 = k_2$, the lemma directly follows from the assumption that each robot has a unique identifier. Consider the case when $k_1 \neq k_2$.

[11]W.l.o.g, we assume that the IDs with all 0s are the same irrespective of their length.

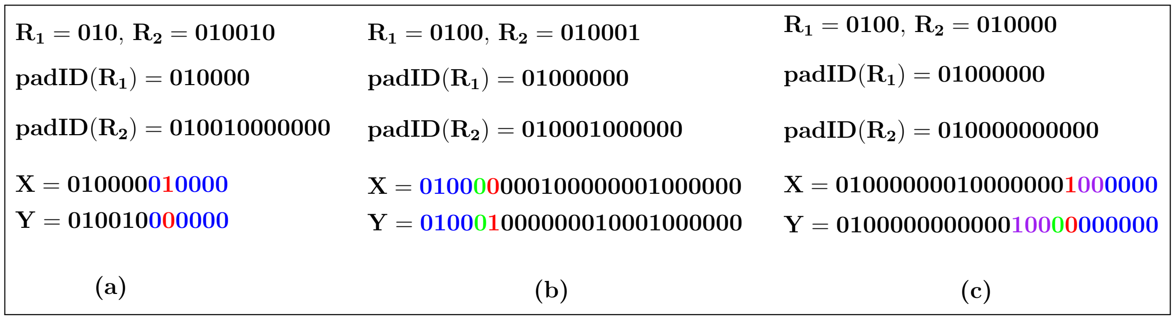
[12]$l.c.m.(l_1, l_2)$ is the least common multiple of integers l_1 and l_2.

[13]This concatenation is done from the left to the right, i.e., the *padID* is concatenated at the right end of the already generated bit pattern.

W.l.o.g., assume that $k_1 < k_2$. In this case, when R_1 is not a substring of R_2, the lemma holds straightforwardly. Thus, we need to consider the case when R_1 is a substring of R_2.

As $k_1 < k_2$, the number of padded 0s in $padID(R_1)$ is less than the number of padded 0s in $padID(R_2)$. To generate X, $padID(R_1)$ is concatenated l^*/l_1 times. Similarly, to generate Y, $padID(R_2)$ is concatenated l^*/l_2 times. There are two possible scenarios when R_1 is a substring of R_2: (i) either $k_2 \geq 2k_1$ or (ii) $k_2 < 2k_1$. Let us assume that the string X is $x_{l^*-1}x_{l^*-2}\cdots x_1x_0$ and the string Y is $y_{l^*-1}y_{l^*-2}\cdots y_1y_0$.

Case 1 ($k_2 \geq 2k_1$): In this scenario, $l_2 \geq 2l_1$ and $k_2 \geq l_1$. As the lengths of X and Y are equal, the substring $x_{l_1-1}x_{l_1-2}\ldots x_0$ of X and the substring $y_{l_2-1}y_{l_2-2}\ldots y_0$ of Y are exactly equal to $padID(R_1)$ and $padID(R_2)$ respectively. Thus, all the bits from y_{k_2-1} to y_0 of Y are 0s. Since $k_2 \geq l_1$ and there is at least one 1 bit in the substring $x_{l_1-1}x_{l_1-2}\ldots x_0$ of X, X and Y differ in at least one bit. Figure 5(a) shows one example of this scenario.


R₁ = 010, R₂ = 010010
padID(R₁) = 010000
padID(R₂) = 010010000000
X = 010000010000
Y = 010010000000
(a)

R₁ = 0100, R₂ = 010001
padID(R₁) = 01000000
padID(R₂) = 010001000000
X = 010000000100000001000000
Y = 010001000000010001000000
(b)

R₁ = 0100, R₂ = 010000
padID(R₁) = 01000000
padID(R₂) = 010000000000
X = 010000000100000001000000
Y = 010000000000010000000000
(c)

Figure 5: Example of different cases of Lemma 5.9 when R_1 is a substring of R_2 and $k_1 < k_2$. Coloured bits are the compared bits of X and Y to find the mismatched bit. The mismatched bit is shown in red. (a) Shows an example of Case 1, when $k_2 = 2k_1$. (b) Shows an example of Case 2a. The first k_1 bits (from MSB) in Y and X are shown in blue. Green bits are the next d bits in X and Y (The mismatched bit among those $d = 2$ bits is shown in red). (c) Shows an example of Case 2b. Here $d = 2, k_1 = 4, k_2 = 6$. Blue bits are the 0 bits among the compared bits due to padding in $padID(R_1)$ and $padID(R_2)$. Green bits are the d 0 bits for R_2 from LSB. Purple bits are the bits that are the 0 bits along with the first 1 bit from the right side (LSB) of R_1. As R_1 is a substring of R_2, those purple bits are present in R_2 also.

Case 2 ($k_2 < 2k_1$): Let $k_2 - k_1 = d > 0$, since $k_1 < k_2$ by our assumption. Then $k_1 > d$, since $k_2 < 2k_1$. For simplicity, let us break this case into two subcases.

Case 2a: Suppose there is at least one 1 in the first d bits of R_2 (from LSB). In this scenario, if we compare the bits from x_{l^*-1} to $x_{l^*-k_2}$ of X with the bits from y_{l^*-1} to $y_{l^*-k_2}$ of Y, then x_{l^*-1} to $x_{l^*-k_1}$ bits of X and y_{l^*-1} to $y_{l^*-k_1}$ bits of Y are the same with R_1 because R_1 is a substring of R_2. There are d number of 0s in between $x_{l^*-k_1}$ to $x_{l^*-k_2}$ of X because $d < k_1$. According to our assumption, there is at least one bit with 1 in between $y_{l^*-k_1}$ to $y_{l^*-k_2}$ of Y. Hence the lemma holds. Figure 5(b) shows one example of this scenario.

Case 2b: Suppose there is no 1 in the first d bits of R_2 (from LSB), i.e., the first d bits are all 0 (from LSB). Let for R_1 (starting from b_0) b_{d_1-1} be the first bit which is 1. As R_1 is a substring of R_2, in this scenario for R_2, b_{d+d_1-1} is the first bit which is 1 in R_2 starting from b_0. Now, if we compare X and Y starting from x_0 and y_0, then for X, $x_{k_1+d_1-1}$ is the first bit which is 1 but for Y, $y_{k_2+d+d_1-1}$ is the first bit which is 1. As $d > 0$, the lemma holds. Figure 5(c) shows one example of this scenario. □

From Lemma 5.9 and the proof of Lemma 5.4, it ensures the meeting of any two robots in Stage 1 of the Algorithm 1 when the ID-lengths of the robots are different. Thus the algorithm works with the remaining stages unchanged. The following corollary shows the running time of Stage 1 of the algorithm.

Corollary 5.9.1. *If the length of the IDs of the robots are different, then Stage 1 of the proposed algorithm (Algorithm 1) takes $n + n.2^{l^*}$ rounds using the* Variable-ID-Padding *scheme, where n is the size of the ring and $l^* = 2 \cdot l.c.m.(k_1, k_2, k_3)$ where k_1, k_2, k_3 are the ID-lengths of robots.*

Therefore, in this case the algorithm finishes in $O(n+n.2^{l^*})$ rounds, since the time bound of Stage 1 dominates the time bounds of the other stages.

6. Exploration with Randomness

We can remove the need for edge crossing detection through the creative use of randomness, resulting in an algorithm that is both Las Vegas and Monte Carlo in nature. In Section 6.1, we bound the algorithm's expected running time and success probability. Subsequently, in Section 6.2, we show how to further use randomness to remove the requirement of having unique IDs initially assigned to each robot.

6.1. Removing Edge Crossing Detection

In this section, we first discuss the changes that are required to be made to the algorithm in Section 5. We then provide bounds for the running time and correctness of the modified algorithm. Note that we assume that the length of the robot IDs, k, is a constant.

Notice that edge crossing detection is used when two robots are located at adjacent nodes and must move in opposite directions along the same edge in the same round. A simple way to get the robots to meet is to force one to be stationary while the other moves. This can easily be achieved by having each robot flip a fair coin to decide if it should move or not. If the result of the coin toss is heads, then the robot performs the movement it initially planned to do. If the result is tails, then the robot does not move. Call this subroutine Random-Movement.

Random-Movement can be run as a subroutine by every robot in every round, after deciding to move (and where to) but before the actual movement. We now describe further modifications to the algorithm in Section 5, in addition to using the subroutine, that are required so that we may remove the need for edge crossing detection.

In Stage 1, notice that if the algorithm required a robot to move for s steps in r rounds, then directly using Random-Movement in the algorithm may cause that robot to move for less than s steps in r rounds, even when the adversary does not block any movement of the robot. This is slightly problematic as we require that there exists a phase in which robots can move for at least n steps, and we do not want to drastically increase the running time. An easy fix is to extend the number of rounds of each phase i by a constant factor, say 8, in order to ensure that on expectation and with high probability, the number of steps a robot moves through is at least 2^i when $2^i \geq n$.[14]

For Stage 2 to occur, we need two robots to come into contact with each other and be marked A and B. Notice that the probability that two robots that were supposed to move through the same edge meet instead is $1/2$. Thus, on expectation, at least one such meeting occurs in two opportunities to meet within k phases once $2^i \geq n$. Note that, in a phase j each robot attempts to move for $8 \cdot 2^j$ rounds in one particular direction. This ensures that in phase j there will be at least 2 meeting opportunities between two robots on expectation when $2^j \geq n$ and the robots are moving in opposite directions. Thus, on expectation after $O(n)$ rounds, Stage 1 is over and Stage 2 begins.

Now, in Stage 2, we require either A or B to come into contact with the third robot. Notice that this third robot is not constrained to only move in one direction, but may move in both directions. Thus, it may only come into contact with either A or B as a result of crossing an edge. Again, through the use of Random-Movement, it takes 2 such attempts on expectation at edge crossing between the third robot and either A or B before contact is made and the third robot gets marked as C. Then the algorithm moves to Stage 3. Notice that before the third robot becomes marked, it is possible that A and B meet again, thus sending the algorithm directly into Stage 4 (since A or B may miss to meet C as there is no edge crossing detection).

If the algorithm is in Stage 3, then the third robot has been marked C. Now, either A and B meet again or the adversary blocks A and B at adjacent nodes and C moves back and forth between them. In the latter case, it takes $O(n)$ rounds with high probability[15] to move to Stage 4 and C, and one of A and B will know the exact value of n.[16] In the former case, after some cn rounds, where c is a positive constant, A and B will meet on expectation. Thus, A and B will know an upper bound cn of n.

In Stage 4, let us assume that without loss of generality, two of the robots A and B learn an upper bound N on the value of n, i.e., $N = cn$ for some constant c. Now, either the third robot is marked or it is not. Either way, our goal in this stage is to inform this third robot that exploration is complete. It is possible that every interaction of A or B with the third robot is a situation where edge crossing would normally occur. In the event of one such interaction, the probability of the third robot being informed is $1/2$. After $2 \log N$ such interactions, the probability of the third robot being informed is at least $1 - 1/n$. Recall that in this stage, robots A and B use a counter and will stop after N rounds. If we change the counter to end at $16N \log N$ instead, then the third robot has $2 \log N$ opportunities with high probability to interact with either robot. Thus, with probability at least $1 - 1/n$ the third robot will interact with at least one of the robots and terminate, eventually resulting in explicit termination.

[14]The expectation is easy to see. The high probability bound can be seen by applying a simple Chernoff bound [39].

[15]The high probability is a result of the use of Random-Movement.

[16]It is possible that A and B may have crossed each other several times before the adversary blocks them at adjacent nodes and the latter case occurs. In this case, C and the robot it finally interacts with will know an upper bound on n. Note that if A and B cross each other at least twice, then on expectation they will meet, leading to the former case.

Construct the new algorithm Modified-Explore-Dynamic-Ring-3-Robots using the above mentioned modifications to the stages and the use of Random-Movement.

Theorem 6.1. *When three robots with constant length IDs run* Modified-Explore-Dynamic-Ring-3-Robots*, exploration of the ring with explicit termination occurs with probability at least* $1 - 1/n$ *in* $O(n \log n)$ *rounds on expectation.*

Proof. The increase in running time is a result of the modifications to Stage 4 by adding $O(n \log n)$ rounds. Note that the increase in the number of rounds to each phase only results in an increase of a constant factor to the running time. The change from exact running time to expected running time is a result of using the Random-Movement subroutine.

The change from explicit termination to explicit termination with high probability is a result of the modification to Stage 4. As we have two robots surely terminating after a certain number of rounds, we can only guarantee with probability at least $1 - 1/n$ that the third robot will terminate as well. □

Remark 1. *If we relax our termination condition to partial termination, we can eliminate Stage 4 of* Modified-Explore-Dynamic-Ring-3-Robots *entirely and simply have robots terminate instead of moving to Stage 4. Then the exploration of the ring with partial termination occurs with high probability in* $O(n)$ *rounds on expectation.*

6.2. Assigning Unique IDs

Throughout this paper, we made the assumption that each robot was initially assigned a unique ID from the range $[1, 2^k]$ prior to the start of the algorithm, where k is the length of the ID bits. This assumption can be removed by having each robot pick an ID uniformly at random from a range of numbers $[1, 2^l]$, where l is a parameter to the algorithm.[17] It is easy to see that the probability that all robots have unique IDs is $(1-1/2^l)(1-2/2^l) = 1-O(1/2^l)$. It should be noted that although l can be made arbitrarily large to improve the probability that each robot has a different ID, a larger value of l possibly results in a longer runtime of the algorithm (refer to Lemma 5.4).

Theorem 6.2. *When three robots run* Modified-Explore-Dynamic-Ring-3-Robots *and choose IDs uniformly at random from the range* $[1, 2^l]$*, exploration of the ring with explicit termination occurs with probability at least* $(1 - 1/n)(1 - O(1/2^l))$ *in* $O((n + n \cdot 2^l) \log n)$ *rounds on expectation.*

7. Exploration in Semi-Synchronous Setting (SSYNC)

In this section, we show how to extend our ideas to the passive transport semi-synchronous model proposed in [22] in order to achieve exploration with partial termination using 3 robots even in the absence of a landmark or the knowledge of n. Recall that in the semi-synchronous setting, in every round, an adversary keeps awake a subset of the robots while the rest are put to sleep with the restriction that every robot is active infinitely often. Furthermore,

[17]Since the total number of robots 3 is known to the robots, they can use it to set a value for l, e.g., $l = 2^3 + 12 = 20$.

recall that we consider the passive transport scenario where a robot that tries to move over a missing edge in some round may complete that move while asleep in a future round subject to some conditions.

In this setting, Di Luna et al. [22] have shown that with 3 robots and either the knowledge of an upper bound on n or the presence of a landmark, they were able to achieve exploration with partial termination. We show that it is possible for 3 robots to achieve exploration with partial termination without either of the above two requirements, so long as the robots have the ability of edge crossing detection. We assume that robots have unique IDs, but that requirement can be removed through the use of randomness as described in Section 6.2.

We consider the four stage algorithm presented in Section 5 and show how to modify it to achieve exploration with partial termination in this passive transport semi-synchronous model. Stage 1 proceeds is essentially the same, but instead of having each robot count the number of rounds per phase, it instead counts the number of successful moves per phase. Stage 2 proceeds as described in the original algorithm. Stage 3 is modified as follows. Robot C will check for an additional condition before determining that the ring has been explored. If $AtoB = BtoA$, and A and B were both trying to move on an edge removed by the adversary, then C determines that the ring has been explored. Note that it is not necessary that A and B were awake when C visited, but merely that they were attempting to move.

The reasoning behind the above changes is that Stage 1 and Stage 4 require robots to rely on counting the number of rounds. The simple trick of counting successful moves solves the Stage 1 problem, but there is no immediate fix to the problems present in Stage 4. However, as only partial termination is considered, Stage 4 of the algorithm is not required. For Stage 3, we require the above change in order to protect against the adversary simply putting A and B to sleep while C moves back and forth between them (and accidentally terminating prematurely). The condition ensures that C has to see them both wanting to move (but not necessarily awake) and prevented by the adversary before deciding to terminate. Since the adversary can only keep a robot asleep for a finite number of rounds and only remove at most one edge from the graph, eventually, one of the robots will make progress on the ring until either the condition is met or A and B meet.

There is the following subtlety to take into account. Suppose that two robots cross the same edge in the same round (or end up co-located at the same node) and at least one of them is asleep. We need both of them to detect that such an edge crossing (or meeting) occurred and furthermore, be able to swap data with one another. This data should include information about whether one of the robots tried to move along an edge while awake but was subsequently put to sleep before the move could be completed.

With the above modifications, we get an algorithm with the following properties.

Theorem 7.1. *There exists an algorithm that correctly explores a* 1*-interval connected dynamic (anonymous) ring of size* n *with partial termination in* $O(fn)$ *steps, where* f *is the largest interval of time between two consecutive activations of any robot, using* 3 *robots with unique IDs and edge crossing detection that have neither common chirality, nor knowledge of an upper bound on* n*, nor access to a landmark.*

Note that we measure number of steps moved and not running time. Furthermore, note that the number of steps is $O(fn)$ and not $O(n)$. This is due to the fact that the adversary can put A and B to sleep for an arbitrarily long time in Stage 3, but not an infinitely long time.

8. Conclusion

In this paper, we have looked into the problem of exploration of a dynamic ring in the presence of 1-interval connectivity. We have first shown that exploration with explicit termination subject to some constraints with just two robots equipped with unique IDs even with access to edge crossing detection and randomness is impossible. Subsequently, we have presented a deterministic algorithm where three uniquely identifiable robots with edge crossing detection capability explore any 1-interval connected dynamic ring in optimal time. We have also shown how to remove the requirement of this capability and allow the robots to be anonymous while still achieving explicit termination with high success probability through the use of randomness. We have finally extended our results to the semi-synchronous setting.

There is an interesting line of future research. Our algorithms intimately used advance knowledge of the number of robots present in the system. If that knowledge is unknown and ≥ 3 robots are present, is there an algorithm to solve exploration with explicit termination?

References

[1] S. Mandal, A. R. Molla, W. 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), Springer, 2020, pp. 92–107.

[2] X. Deng, C. H. Papadimitriou, Exploring an unknown graph, Journal of Graph Theory 32 (3) (1999) 265–297.

[3] C. A. Duncan, S. G. Kobourov, V. A. Kumar, Optimal constrained graph exploration, ACM Transactions on Algorithms 2 (3) (2006) 380–402.

[4] J. Augustine, W. K. Moses Jr., Dispersion of mobile robots: A study of memory-time trade-offs, CoRR abs/1707.05629.

[5] A. D. Kshemkalyani, A. R. Molla, G. Sharma, Dispersion of mobile robots in the global communication model, in: International Conference on Distributed Computing and Networking (ICDCN), 2020, pp. 1–10.

[6] M. Cieliebak, P. Flocchini, G. Prencipe, N. Santoro, Solving the robots gathering problem, in: International Colloquium on Automata, Languages, and Programming (ICALP), Springer, 2003, pp. 1181–1196.

[7] G. Prencipe, Impossibility of gathering by a set of autonomous mobile robots, Theoretical Computer Science 384 (2-3) (2007) 222–231.

[8] L. Barriere, P. Flocchini, E. Mesa-Barrameda, N. Santoro, Uniform scattering of autonomous mobile robots in a grid, International Journal of Foundations of Computer Science 22 (03) (2011) 679–697.

[9] T. Izumi, M. G. Potop-Butucaru, S. Tixeuil, Connectivity-preserving scattering of mobile robots with limited visibility, in: International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS), Springer, 2010, pp. 319–331.

[10] M. Dynia, J. ŁopuszaŃski, C. Schindelhauer, Why robots need maps, in: International Colloquium on Structural Information and Communication Complexity (ICALP), Springer, 2007, pp. 41–50.

[11] P. Brass, F. Cabrera-Mora, A. Gasparri, J. Xiao, Multirobot tree and graph exploration, IEEE Transactions on Robotics 27 (4) (2011) 707–717.

[12] P. Brass, I. Vigan, N. Xu, Improved analysis of a multirobot graph exploration strategy, in: 13th International Conference on Control Automation Robotics & Vision (ICARCV), IEEE, 2014, pp. 1906–1910.

[13] D. Dereniowski, Y. Disser, A. Kosowski, D. Pająk, P. Uznański, Fast collaborative graph exploration, Information and Computation 243 (2015) 37–49.

[14] S. Das, P. Flocchini, S. Kutten, A. Nayak, N. Santoro, Map construction of unknown graphs by multiple agents, Theoretical Computer Science 385 (1-3) (2007) 34–48.

[15] B. Awerbuch, M. Betke, R. L. Rivest, M. Singh, Piecemeal graph exploration by a mobile robot, Information and Computation 152 (2) (1999) 155–172.

[16] R. Baldoni, F. Bonnet, A. Milani, M. Raynal, Anonymous graph exploration without collision by mobile robots, Information Processing Letters 109 (2) (2008) 98–103.

[17] S. Albers, M. R. Henzinger, Exploring unknown environments, SIAM Journal on Computing 29 (4) (2000) 1164–1188.

[18] X. Deng, C. H. Papadimitriou, Exploring an unknown graph, Journal of Graph Theory 32 (3) (1999) 265–297.

[19] Y. Dieudonné, A. Pelc, Deterministic network exploration by anonymous silent agents with local traffic reports, ACM Transactions on Algorithms 11 (2) (2014) 10:1–10:29.

[20] P. Fraigniaud, D. Ilcinkas, G. Peer, A. Pelc, D. Peleg, Graph exploration by a finite automaton, Theoretical Computer Science 345 (2-3) (2005) 331–344.

[21] P. Panaite, A. Pelc, Exploring unknown undirected graphs, Journal of Algorithms 33 (2) (1999) 281–295.

[22] G. A. Di Luna, S. Dobrev, P. Flocchini, N. Santoro, Distributed exploration of dynamic rings, Distributed Computing 33 (1) (2020) 41–67.

[23] S. Das, Graph explorations with mobile agents, in: Distributed Computing by Mobile Entities, Springer, 2019, pp. 403–422.

[24] G. A. Di Luna, Mobile agents on dynamic graphs, in: Distributed Computing by Mobile Entities, Current Research in Moving and Computing, Springer, 2019, pp. 549–584.

[25] F. Kuhn, N. Lynch, R. Oshman, Distributed computation in dynamic networks, in: ACM symposium on Theory of computing (STOC), ACM, 2010, pp. 513–522.

[26] R. O'Dell, R. Wattenhofer, Information dissemination in highly dynamic graphs, in: Joint Workshop on Foundations of Mobile Computing (DIALM-POMC), ACM, 2005, pp. 104–110.

[27] C. Avin, M. Kouckỳ, Z. Lotker, How to explore a fast-changing world (cover time of a simple random walk on evolving graphs), in: International Colloquium on Automata, Languages, and Programming (ICALP), Springer, 2008, pp. 121–132.

[28] D. Ilcinkas, A. M. Wade, Exploration of the t-interval-connected dynamic graphs: The case of the ring, Theory of Computing Systems 62 (5) (2018) 1144–1160.

[29] D. Ilcinkas, R. Klasing, A. M. Wade, Exploration of constantly connected dynamic graphs based on cactuses, in: International Colloquium on Structural Information and Communication Complexity (SIROCCO), Springer, 2014, pp. 250–262.

[30] T. Erlebach, M. Hoffmann, F. Kammer, On temporal graph exploration, Journal of Computer and System Sciences 119 (2021) 1–18.

[31] O. Michail, P. G. Spirakis, Traveling salesman problems in temporal graphs, Theoretical Computer Science 634 (2016) 1–23.

[32] P. Flocchini, B. Mans, N. Santoro, On the exploration of time-varying networks, Theoretical Computer Science 469 (2013) 53–68.

[33] D. Ilcinkas, A. M. Wade, On the power of waiting when exploring public transportation systems, in: International Conference on Principles of Distributed Systems (OPODIS), Springer, 2011, pp. 451–464.

[34] T. Gotoh, P. Flocchini, T. Masuzawa, N. Santoro, Tight bounds on distributed exploration of temporal graphs, in: International Conference on Principles of Distributed Systems (OPODIS), 2019.

[35] G. A. Di Luna, P. Flocchini, L. Pagli, G. Prencipe, N. Santoro, G. Viglietta, Gathering in dynamic rings, Theoretical Computer Science 811 (2020) 79–98.

[36] A. Agarwalla, J. Augustine, W. K. Moses Jr., M. Sankar K., A. K. Sridhar, Deterministic dispersion of mobile robots in dynamic rings, in: International Conference on Distributed Computing and Networking (ICDCN), ACM, 2018, pp. 19:1–19:4.

[37] A. D. Kshemkalyani, A. R. Molla, G. Sharma, Efficient dispersion of mobile robots on dynamic graphs, in: IEEE International Conference on Distributed Computing Systems (ICDCS), IEEE, 2020, pp. 732–742.

[38] F. Kuhn, R. Oshman, Dynamic networks: models and algorithms, SIGACT News 42 (1) (2011) 82–96.

[39] M. Mitzenmacher, E. Upfal, Probability and Computing: Randomized Algorithms and Probabilistic Analysis, Cambridge University Press, New York, NY, USA, 2005.