Live Exploration of Dynamic Rings

G. Di Luna[*], S. Dobrev [†], P. Flocchini [‡], N. Santoro [§]

Abstract

Almost all the vast literature on graph exploration assumes that the graph is *static*: its topology does not change during the exploration, except for occasional faults. To date, very little is known on exploration of *dynamic* graphs, where the topology is continously changing. The few studies have been limited to the *centralized* (or post-mortem) case, assuming complete a priori knowledge of the changes and the times of their occurrence, and have only considered fully synchronous systems.

In this paper, we start the study of the *decentralized* (or live) exploration of dynamic graphs, i.e. when the agents operate in the graph unaware of the location and timing of the changes. We consider dynamic rings under the standard *1-interval-connected* restriction, and investigate the feasibility of their exploration, in both the fully synchronous and semi-synchronous cases. When exploration is possible we examine at what cost, focusing on the minimum number of agents capable of exploring the ring. We establish several results highlighting the impact that *anonymity* and *structural knowledge* have on the feasibility and complexity of the problem.

1 Introduction

1.1 Framework

Graph exploration is an extensively investigated problem, starting from the pioneering work of Shannon [41]. The problem requires a team of mobile computational entities, called *agents* or *robots*, capable of moving from node to neighbouring node, to visit the nodes of the graph, with the requirement that each node is eventually visited by at least one agent. The exploration is said to be *with termination* if the agents are required to stop within finite time upon completing the task, *perpetual* otherwise. In the vast literature on the subject (e.g., see [1, 16, 17, 20, 25, 29, 40]), a huge spectrum of different assumptions have been made and examined e.g. with regards to: the computational power of the agent(s) (e.g., FSM, Turing, etc.); the structure of the graph and its properties; the level of topological knowledge available to the agents; whether or not the network is anonymous (i.e., the nodes lack distinct identifiers); whether the nodes can be marked (e.g., by leaving a pebble); the level of synchronization and the means of communication (in case of multiple agents); the presence of faulty nodes and/or faulty links; whether the agents are required to stop within finite time upon completing the task (exploration with termination) or not (perpetual exploration); etc.

Regardless of their differences, all these investigations share the common assumption that the graph is *static*: its topological structure does not change during the exploration. This is true also for those investigations considering faulty nodes and/or faulty links (e.g., see [6, 15, 21, 22]).

Dynamic Graphs Recently, within distributed computing, researchers started to investigate situations where the graph is highly *dynamic*, that is where the topology changes continuously. This research is motivated by the rapid development of networked systems where changes in the topology are extensive, and occur continuously. Such are, for example, the infrastructure-less networks created by wireless mobile

[*]University of Rome "Sapienza", diluna@dis.uniroma1.it
[†]Slovak Academy of Sciences, stefan.dobrev@savba.sk
[‡]University of Ottawa, flocchin@site.uottawa.ca
[§]Carleton University, santoro@scs.carleton.ca

entities (e.g. vehicles, satellites, robots, or pedestrian smartphones): an edge exists between two entities at a given time if they are within communication range at that time; hence, the topology changes continuously as the movements of the entities destroy old connections and create new ones. In these highly dynamic networks, changes are not anomalies (e.g., faults) but rather integral part of the nature of the system. These infrastructure-less highly dynamic networks, variously called delay-tolerant, disruptive-tolerant, challenged, epidemic, opportunistic, have been long and extensively investigated by the engineering community and, more recently, by distributed computing researchers.

These highly dynamic networks are modelled in a natural way in terms of *time-varying graphs* (sometimes called temporal graphs) where the *presence* of an edge as well as its *latency* (i.e. the time necessary for an agent or a message to traverse it) are functions of time; this model has been formally defined in [13], identifying main classes of systems studied in the literature and their computational relationship. If time is discrete (e.g., changes occur in rounds), the evolution of these systems can be equivalently described as a sequence of static graphs, called *evolving graph*, a model suggested in [31] and formalised in [24].

The study of distributed computations in highly dynamic networks using these models has focused mainly on problems of information diffusion and reachability (e.g., broadcast, routing, election, consensus, etc) with or without faults; e.g., see [2, 3, 5, 7, 8, 10–12, 14, 19, 30, 35, 36]). Clearly, all these studies make strong assumptions in order to restrict the universe of the possible topological changes and their temporal occurrence. One such restriction is by assuming that topological changes are *periodic* (Class 8 of [13]); this assumption describes in a natural way systems generated by the movement of entities with periodic routes (e.g., see [9, 26, 27, 33, 37]). A popular restiction is by assuming that the network is *always connected* (Class 9 of [13]): at each time instant, there is a connected spanning subgraph, and the latency is sufficient for an agent movement or a message transmission; further assuming that such connected spanning subgraph persists for $T \geq 1$ time units defines the extensively studied sub-class of *T-interval-connected* systems (e.g., see [18, 35, 36, 39]).

Exploration of Dynamic Graphs Returning to the exploration problem, very little is known in the case of dynamic graphs. On the probabilistic side, there is the early seminal work on random walks [4]. On the deterministic side, the only studies are on: the complexity of computing a foremost exploration schedule under the *1-interval-connected* assumption [38], this topic has been extended in [23] where several results on the exploration complexity for specific topologies are given; the computation of an exploration schedule for *rings* under the *T-interval-connected* assumption [34]; and the computation of an exploration schedule for *cactuses* under the *1-interval-connected* assumption [32]. All these studies are however *centralized* (or off-line, or post-mortem); that is, they assume that the exploring agents have complete A PRIORI knowledge of the topological changes and the times of their occurence.

Little is known on the *distributed* (or on-line, or live) case, i.e. when the location and timing of the changes are unknown to the agents. Exploting a reduction technique employed in some the proofs of [23, 34], it is possible to infer that in the synchronous setting two agents starting in the same initial position, and unaware of the adversarial schedule, can explore a ring in $O(n)$ rounds. Apart from this basic result, nothing else is known.

In this paper, we start such an investigation and study the deterministic decentralized exploration of a dynamic ring under the *1-interval-connected* assumption, focusing on the impact that *synchrony*, *anonymity* and *topological knowledge* have on its computability and complexity.

1.2 Contributions

We examine the problem of exploring *1-interval-connected* dynamic rings. That is, we consider a ring network in which, at each time step, at most one link is missing, the choice being performed by an adversary. We study under what conditions all nodes can be visited by a team of agents unaware of the choices of the adversary. When exploration is possible, we examine the cost, in terms of number of agents and time.

We first consider the *fully-synchronous* systems (Section 3), traditionally assumed in the literature; i.e., all agents are active at each time step, We then introduce the notion of *semi-synchronous* systems (Section 4), where only a subset of agents might be active at each time step (the choice of the subset is made by an adversary). The semi-synchronous model is common in the context of mobile agents in continuous spaces

N. Agents	*Assumptions*	*Exploration and Termination*
2	No bounds on n, Anonymous Ring	Exploration with Termination **impossible** (Th. 1)
2	Known upper bound N, Anonymous Ring, Chirality	Exploration with Termination in time $3N$
2	Known upper bound N, Anonymous Ring, No Chirality	Exploration with Termination in time $5N$
2	No bounds on n, Landmark, Chirality	Exploration with Termination in time $O(n)$ (Sec. 3.2.2)
2	No bounds on n, Landmark, No Chirality	Exploration with Termination in time $O(n \log(n))$

Figure 1: Results for $\mathcal{FSYNC}$ model

Model	*N. Agents*	*Assumptions*	*Exploration and Termination*
(NS)	Any	Non-Anonymous Ring, Known n	Exploration **impossible** (Th. 8)
(PT)	2	Known n	Exploration **impossible** (Th. 9)
		Known Upper Bound N, Chirality	Exploration with Termination of one agent in moves $O(N^2)$; Lower Bound of $\Omega(nN)$ moves (Th. 11)
	3	Known Upper Bound N, No Chirality	Exploration with Termination of one agent in moves $O(N^2)$
(ET)	Any	Known upper bound N, Non-Anonymous Ring, Chirality	Exploration with Termination **impossible** (Th. 14)
	3	Known ring size, No Chirality	Exploration with Termination of one agent (Sec. 4.2)

Figure 2: Results for $\mathcal{SSYNC}$ models

(e.g., [28]) but has never been studied before for agents moving in graphs. Our main focus is on the impact that the level of synchrony as well as other factors such as knowledge of the size of the ring, chirality ((i.e., common sense of orientation), anonymity, and communication, have on the solvability of the problem.

– We first concentrate on *fully synchronous* systems ($\mathcal{FSYNC}$) examining solvability of the exploration problem with *two* agents (after showing that it is unsolvable with one).

For *anonymous* rings, we establish a computational separation between exploration with termination and perpetual exploration. More precisely, we prove that *perpetual exploration is possible* with two agents, even without knowledge of the ring size, chirality and communication. By contrast, we show that *exploration with termination is impossible* without knowledge of (an upperbound on) the ring size; this holds even if the agents have unique IDs and can communicate when they meet at the same node. We conclude by showing that knowledge of an upperbound on the ring size is actually sufficient for two agents to explore with termination.

For *non-anonymous* rings, we show that the presence of a single observably different node (*landmark*) allows two agents to solve the exploration problem with global termination without the need of any additional information.

– We then examine *semi-synchronous* systems ($\mathcal{SSYNC}$), distinguishing among different *transportation models* (described in details later in the paper) depending on what happens to an agent a waiting to traverse a missing link e, if the agent is inactive when e appears.

If a is not allowed to move (*No Simultaneity Model* - NS), exploration is impossible with any number of agents, even with exact knowledge of the ring size, nodes with distinct IDs, agents with communication ability and common chirality. If a is not allowed to move, but is guaranteed to be eventually active at a round when the edge is present (*Eventual Transport Model* - ET)), exploration with global termination is impossible, with any number of agents, even if an upperbound on the ring size is known, nodes have distinct IDs, agents communicate and agree on chirality. On the other hand, with exact knowledge of the ring size, we prove that exploration is possible with three agents even without chirality or communication, and with a stronger termination condition than perpetual exploration: at least one of the agent terminates within finite time. Finally, if a is passively transported on e agents it appear (*Passive Transport Model* - (PT)), we show that, without chirality, two anonymous robots with finite memory are not sufficient to explore the ring; the result holds even if there is a distinguished landmark node and the exact network size is known. On the other hand, with chirality, two agents with a known upperbound on the ring size can perform the exploration.

All the sufficiency proofs are constructive. A summary of the results is shown in Figures 1 and 2.

2 Model and Basic Limitations

2.1 Model and Terminology

Let $\mathcal{R} = (v_0, \ldots v_{n-1})$ be a discrete time-varying graph that is never disconnected and whose footprint is a ring; in other words, $\mathcal{R}$ is a synchronous ring where, at any time step $t \in N$, one of its edges might not be present. Such a dynamic network is known in the literature as a *1-interval connected* ring.

The ring is anonymous, that is the nodes have no distinguishable identifiers. Each node v_i is connected to its two neighbours v_{i-1} and v_{i+1} via distinctly labeled ports q_{i-} and q_{i+}, respectively (all operations on the indices are modulo n); the labeling of the ports may not be globally consistent and thus might not provide an orientation.

Operating in $\mathcal{R}$ is a set $A = \{a_0, \ldots, a_{m-1}\}$ of agents, each provided with memory and computational capabilities. The agents are anonymous and all execute the same protocol. Any number of agents can reside at a node at the same time. Initially located at arbitrary nodes, they do not have any explicit communication mechanism, nor can leave marks on the nodes. The agents are *mobile*, that is they can move from node to neighboring node. To move, an agent has to position itself on the port from which it wants to leave and access to a port is done in mutual exclusion: at every time step, on each port there is at most one agent.

Each agent a_j has a consistent private orientation of the ring; that is, it has a function λ_j which designates each port either *left* or *right* and $\lambda_j(q_{i-}) = \lambda_j(q_{k-})$, for all $0 \leq i, k < n$. The orientation of the agents might not be the same. If all agents agree on the orientation, we say that there is *chirality*.

The system operates in *synchronous* time steps, called *rounds*. Initially, all agents are *inactive*. Each time step $t \in N$ starts with a non-empty subset $A(t) \subseteq A$ of the agents becoming *active*. Upon activation, agent $a_j \in A(t)$ at node v_i performs a sequence of operations: Look, Compute, and (possibly) Move.

- Look: The agent determines its own position within the node (i.e., whether or not is on a port, and if so on which one), and the position of the other agents (if any) at that node. We call this information a *snapshot*. Let $myPos \in \{left, right, nil\}$ indicate the position of a_j.
- Compute: Based on the snapshot and the content of its local memory, the agent executes its protocol (the same for all agents) to determine whether or not to move and, if so, in what direction; the result will be $direction \in \{left, righ, nil\}$, where $left$ and $right$ are with respect to its own local orientation. If $myPos \in \{left, right\}$ and $direction \neq myPos$, the agent leaves the port. Then, if $direction = nil$, the agent becomes inactive. If $direction \neq nil$, a_j attempts to access the appropriate port (if not already there); if it gains access, it positions itself on the port, otherwise it sets private variable $moved = false$ and becomes inactive.
- Move: Let the agent be positioned on port q_{i-} (resp., q_{i+}) after computing. If the link between v_i and v_{i-1} (resp., v_{i+1}) is present in this round, then agent a_j will move to v_{i-1} (resp., v_{i+1}), reach it, set private variable $moved = true$, and become inactive. If the link between v_i and v_{i-1} (resp., v_{i+1}) is not present, then agent a_j will remain in the port, set $moved = false$, and become inactive. In either case, access to port q_{i-} (resp., q_{i+}) continues to be denied to any other requesting agent during this round.

By definition, the delays are such that all active agents have become inactive by the end of round t; the system then starts the new round $t + 1$.

Notice that, since access to a port is in mutual exclusion, in the same round at most one agent will move in each direction on the same edge. Also note that two agents moving in opposite directions on the same edge in the same round might not be able to detect each other.

A major computational factor is the nature of the activation schedule of the agents. If $A(t) = A$ for all $t \in N$, that is all agents are activated at every time step, the system is said to be *fully synchronous*

($\mathcal{FSYNC}$). Otherwise the system is said to be *semi-synchronous* ($\mathcal{SSYNC}$); the agents that are not activated in a round are said to be *sleeping* in that round; every agent is activated infinitely often.

Observe that in $\mathcal{SSYNC}$ it is possible for an agent to be sleeping on a port. This is indeed the case when an agent a gains access to a port q when the link is not there (thus, it remains on q), and a is not activated in the next round. What may happen to an agent sleeping on a port gives raise to different models, described in the following in a decreasing order of computational power (for the agents):

- Passive Transport (PT): If an agent is sleeping on a port at round t and the corresponding edge is present in that round, the agent is moved to the other endpoint of the edge in round t.
- Eventual Transport (ET): A sleeping agent cannot move. If an agent is sleeping on a port at round t and it does not exists a $t' > t$ such that the corresponding edge is missing for any round $t' > t$, then the agent will eventually become active at a round $t'' > t$ when the corresponding edge is present (simultaneity condition).
- No Simultaneity (NS): A sleeping agent cannot move. There is no guarantee of simultaneity for an agent sleeping on a port.

2.2 Basic Limitations

We begin our study by showing simple impossibility results. It is interesting to notice that, without some knowledge of the size of the ring, or without the asymmetry introduced by a landmark node, exploration with termination is impossible even in the fully synchronous model, even if the agents have distinct IDs and are equipped with face to face communication.

Observation 1. *The adversary can prevent an agent from leaving the initial node v_0, by always removing the edge over which the agent wants to leave v_0.*

From this Observation, we immediately get:

Corollary 1. *A single agent is not able explore the ring.*

Hence at least two agents are needed. In case the two agents start from different locations, obviously they can only meet by arriving simultaneously at the same node or by one reaching the other while it is still. The adversary can prevent both occasions from occurring by removing the appropriate edge. That is,

Observation 2. *The adversary can prevent two agents starting at different locations from meeting each other, even if they have unlimited memory, unlimited communication capabilities, and know each other's ID.*

Since the ring is anonymous, if its size is unknown, the only way to detect the termination of exploration is to meet at least once. Hence, Observation 2 yields:

Theorem 1. *There does not exist an explicitly terminating deterministic exploration algorithm of anonymous rings of unknown size by two agents with unique IDs.*

3 Ring Exploration in $\mathcal{FSYNC}$

We consider exploration when the system is fully synchronous, presenting and analyzing protocols that solve the problem under different assumptions on knowledge of the ring size, anonimity of the nodes, and presence of chirality. All these solutions do not require the agents to be able to communicate explicitly.

Our algorithms use as a building block procedure Explore ($dir \mid p_1 : s_1;\ p_2 : s_2;\ \ldots;\ p_k : s_k$), where *dir* is either *left* or *right*, p_i is a predicate, and s_i is a state. In Procedure Explore, the agent performs Look, then evaluates the predicates $p_1, \ldots, p_k$ in order; As soon as a predicate is satisfied, say p_i, the procedure exits and the agent does a transition to the specified state, say s_i. If no predicate is satisfied, the agent tries to Move in the specified direction *dir* and the procedure is executed again in the next round.

Furthermore, the following variables are maintained by the algorithms:

- $Ttime$, $Tsteps$: the total number of rounds and the successful moves, respectively, since the beginning of the execution of the algorithm.
- $Etime$, $Esteps$: the total number of rounds and the successful moves, respectively, since procedure Explore has been called.
- $Btime$: the number of consecutive rounds the agent has been currently waiting in a queue.

In particular, the following predicates are used:

- *meeting*: both agents are in the node, having performed successful move.
- *catches*: the agent is in the node after a successful move, the other agent is observed on a port (in your moving direction).
- *catched*: the agent is on the port after a failed move, the other agent is observed in the node.

Observe that, in a synchronous system, when predicate *catches* holds for an agent then *catched* holds for the other agent. In the following, we says that the agents *catch each other* if those predicates hold.

3.1 Known Upper Bound on Ring Size

In this section we study the simple case of exploring the ring when the agents know an upper-bound $N \geq n$ on the ring size. We first show how to solve the problem when the agents agree on the ring chirality; we then show how the two agents can explore the ring even if no such agreement exists, albeit with a higher time complexity.

3.1.1 With Chirality

If the agents agree on chirality (i.e., on *left*/*right* orientation), then they can explore the ring and terminate, even if they are anonymous. The algorithm is fairly simple: Upon wake-up, an agent explores to the left until it crosses N edges, or $3N$ time steps elapsed since the start, or it catches up with the other agent. In the first two cases, the agent terminates. In the latter case, it continues the exploration changing direction, and terminates at time $3N$.

```
States: {Init (initial state), Bounce (change direction), Terminate}.
In state Init:
    EXPLORE(left | $Ttime \geq 3N \vee Tsteps \geq N$: Terminate; catches: Bounce)
In state Bounce:
    EXPLORE(right | $Ttime \geq 3N$: Terminate)
```

Figure 3: Algorithm KnownNWithChirality

Theorem 2. *Algorithm* KnownNWithChirality *allows two anonymous agents with chirality to explore a 1-interval connected ring and to terminate in time $3N$, where N is a known upper-bound on the ring size.*

Proof. Termination follows by construction since in all calls to Explore a $Ttime \geq 3N$ threshold is specified for termination. It remains to show that at the moment of termination the ring has been explored.

Assume the contrary. If an agent terminates due to the $Tsteps$ threshold being reached, this agent has explored the whole ring. Observe that (by chirality and construction) in each round either at least one agent makes progress, or both agents are waiting on the same edge at the opposite endpoints. Note that the latter case means that the ring has been explored. Hence, at time $3N$, the total number of moves by the two agents is at least $3N$; since only one of them can reverse direction, at least one of them has traveled N edges in one direction, exploring the ring completely. □

3.1.2 Without Chirality

Also without chirality, the problem is solvable (with a slightly higher complexity). Note that *left* and *right* now refer to the local orientation of an individual agent.

```
States: {Init, Bounce, , Forward, Terminate}.
In state Init:
    Explore(left | $Ttime \geq 5N$: Terminate; $Btime = N$: Bounce; catches: Bounce; catched: Forward)
In state Bounce:
    Explore(right | $Ttime \geq 5N$: Terminate)
In state Forward:
    Explore(left | $Ttime \geq 5N$: Terminate)
```

Figure 4: Algorithm KnownNNoChirality

Theorem 3. *Algorithm* KnownNNoChirality *allows two anonymous agents without chirality to explore a 1-interval connected ring and to terminate in time $5N-7$, where N is a known upper-bound on the ring size.*

Proof. As in Theorem 2, it is sufficient to show that the ring has been explored in the case that both agents terminate when $Ttime = 5N-7$.

First observe that each agent changes direction at most once. This means, that, if the number of moves by both agents is at least $4N-7$, at least one of them has made $N-1$ moves in one direction and has fully explored the ring. Next, note that the only rounds when no agent makes a move is when they are both waiting on the same edge at the opposite endpoints. However, as long as the ring has not been explored, this can happen only once (when they first approach each other from opposite directions), and for at most N consecutive rounds. Should they be both waiting on the same for a second time, the ring has been already been explored. Hence, exploration is guaranteed after $5N-7$ rounds. $\square$

3.2 No Bounds On Ring Size

We now consider exploring the ring when no upper-bound on its size is available to the agents. Under this condition, by Theorem 1, it is *impossible* for two agents to explore an anonymous ring with termination, even if the agents have unique IDs. Hence, for exploration to occur, either termination must not be required or the ring must not be anonymous. In the following we consider precisely those two cases. We first show how perpetual exploration can be performed without any other condition even if the agents are anonymous. We then consider a ring in which there is a special node, called landmark, different from the others and visible to the agents; we prove that exploration can be performed with termination, even if the agents are anonymous, in time $O(n)$ if there is chirality, $O(n \log n)$ otherwise.

3.2.1 Perpetual Exploration

We present a protocol, PerpetualExploration, that allows two anonymous agents to perform perpetual exploration without knowing any bound on the ring size. The basic idea of the algorithm is for each agent to guess the size of the ring with an initial estimate and move in one direction for a time equal to twice the estimate; the agent will then double the size estimate, change direction, and repeat this process with the new guess.

```
States: {Init, Bounce, Reverse, Forward, Terminate}.
In state Init:
    $N \leftarrow 2$, $dir \leftarrow left$
    Explore($dir$; $Etime \geq 2N$: Reverse, $catches$: Bounce, $catched$:Forward)
In state Reverse:
    $N \leftarrow 2 * N$, $dir \leftarrow opposite(dir)$
    Explore($dir$; $Etime \geq 2N$: Reverse, $catches$: Bounce, $catched$:Forward)
In state Bounce:
    Explore($opposite(dir)$)
In state Forward:
    Explore($dir$)
```

Figure 5: Algorithm PerpetualExploration

Theorem 4. *Algorithm* PerpetualExploration *allows two anonymous agents without chirality to explore a 1-interval connected ring in $O(n)$ time (but never explicitly terminates).*

Proof. If the agents catch each other, then they start moving in opposite directions and, in the subsequent $n-1$ moves, (unknown to them) they will explore the whole ring. Consider now the case when the agents never catch each other. Since N is always doubled after $2N$ time steps, at time $t_n \leq 4n$, $N \geq n$. If, at that time, the agents are moving in the same direction, since they do not catch each other and in each time step at least one of them makes progress, in the next $2N$ time steps they will explore the ring. If, at that time, the agents are moving in opposite directions, they will either explore the ring, or get blocked on the same edge. In the latter case, they reverse direction at time $t_n + 2N$ and the ring is explored by time $t_n + 3N \in O(n)$. □

3.2.2 Landmark and Chirality

Consider a ring with a special node v^*, called landmark, identifiable by the agents. When performing a Look operation at some node v, a flag $IsLandmark$ is set to *true* if and only if $v = v^*$. In this subsection we assume chirality, but no other additional knowledge. We show that two anonymous agents can explore the ring and terminate.

The basic idea is to explore the ring using the landmark to compute the size and allow termination. In order to coordinate termination, the agents implicitly "communicate" when they catch each other (by waiting at the node if not sure whether to terminate, and by leaving it if they already know that the ring is explored). When the agents catch each other for the first time, they break symmetry and assume different roles. We assign to them logical names: F for the agent being caught, and B for the one that caught F. These names do not change afterwards, even though it is possible for F to catch B later on.

Procedure LExplore is very similar to Explore with the following additions:

- Each agent keeps track of whether it is crossing the landmark and in which direction; furthermore, it tracks its distance from the landmark (since encountering it for the first time). In this way, it can detect whether it made a full loop around the ring. When it does so for the first time, variable n is set to the ring size (n is initialized to infinity, all the tests using it while it has this initial value will fail).
- An additional variable $Ntime$ is maintained, tracking the total number of rounds since the agent learned n.

The complete pseudocode is shown in Figure 6. Both agent start going left. If they never meet, they terminate (see Lemma 1). If they catch each other, the naming is done. After naming, agent F keeps going left. Agent B moves right until either it completes a loop of the ring or it is blocked for a number of rounds

```
States: {Init, Bounce, Return, Forward, Terminate, BComm, FComm}.
In state Init:
    LExplore($left$ | $Ntime > 2n$: Terminate; $catches$: Bounce; $catched$: Forward)
In state Bounce:
    LExplore($right$ | $meeting$: Terminate; $Etime > 2Esteps \vee Ntime > 0$: Return)
In state Return:
    $bounceSteps \leftarrow Esteps$
    LExplore($left$ | $Ntime > 3n \vee catched$: Terminate; $catches$: BComm)
In state Forward:
    LExplore($left$ | $Ntime > 5n \vee meeting \vee catches$: Terminate; $catched$: FComm)
In state BComm:
    $returnSteps \leftarrow Esteps$
    if $returnSteps \leq 2 * bounceSteps$ then                    ▷ both must have waited on the same edge
        Move ($right$)                                            ▷ signal the need to terminate
        Terminate in the next round
    else
        Stay for one round in the node
        if agent F is in the node then                            ▷ agent F waited to learn whether to terminate
            change state to Bounce and process it (in the same round)
        else                ▷ agent F left, or tried to leave and is on the port – signalling to terminate
            Terminate
In state FComm:
    if you know that the ring is explored ($n$ is known) then
        Move ($left$)                                             ▷ signal to B that F knows $n$
        Terminate in the next round
    else
        Move from the port to the node                            ▷ i.e. staying at the same node
        if agent B is in the node then                            ▷ this happens next round
            Change state to Forward and process it (in the same round)
        else                                                      ▷ B has left or is on the port
            Terminate
```

Figure 6: Algorithm LandmarkWithChirality

equal to twice the number of edges it has traversed so farm, i.e. predicate $Etime > 2Esteps$. When one of these conditions is satisfied, agent B goes left, and it tries to catch up with F.

If they catch up and F has done less than $Esteps$ steps to the left from its old position, then B and F have waited on the same edge, and hence the ring has been explored; B can detect this, and it "communicates" the end of exploration to F.

If they do not catch up, for a certain number of rounds, then they will both know that the ring is explored and they can terminate independently (see Lemma 2).

If they catch up but F has done at least $Esteps$ steps to the left from its old position (B can detect this), and they both keep executing the algorithm. Note, that F has made progress towards completing a ring loop, a condition that can be detected because of the landmark. Should such event occur then both F and B will eventually terminate.

Lemma 1. *In Algorithm* LandmarkWithChirality*, if the agents do not catch each other and stay in the* Init *state, then they will explore the ring and terminate by round* $7n - 2$*.*

Proof. As the agents are moving in the same direction, but they start from different nodes, in each round at least one of them makes progress. Since they do not catch each other, the difference between the number of successful moves by the agents is at most $n - 1$. Therefore, if by round $5n - 2$ no agent has terminated, then both agents have crossed at least $2n - 1$ edges and hence both know n. By construction, in a further $2n$ steps, the agents will terminate. If an agent has terminated at round $r < 5n - 2$, this means that at time $r - 2n$ this agent knew n, i.e. it has entered the landmark for the second time. As the agents did not catch each other, the other agent must have already entered the landmark. Since in the subsequent $2n$ steps the agents do not catch each other and together made progress at least $2n$ times, by round r the other agent will enter the landmark for the second time, and by round $r + 2n$ it will terminate as well. □

Lemma 2. *In Algorithm* LANDMARKWITHCHIRALITY*, if an agent terminates, then the ring has been explored and the other agent will terminate as well.*

Proof. We show the lemma by case analysis on how the agents terminate. According to the algorithm, both agents terminate at the same time in only the four cases considered below; when this happens the ring has been explored:

- Agent F, in state Forward, catches agent B in state Return at node w: Consider the node v where B catched F and changed state to Bounce the last time. The counter-clockwise segment from v to w has been explored by F, that never changed its direction. Consider now the node z where B changed state to Return the last time; it is not difficult to see that z must be in the counter-clockwise segment from v to w; this in turn implies that B has explored the clockwise segment from v to w. Thus the entire ring has been explored.
- The agents moving in opposite direction meet at a node: The entire ring has clearly been explored.
- Agent F knowing n and signals B to terminate: By construction.
- Agent B signals F to terminate: According to the algorithm when this occurs either B knows n, or $returnSteps < 2 * bounceSteps$. If B knows n the lemma trivially holds. Consider the second condition. When agent B changes its direction, it has been blocked (not necessarily on the same edge) at least $bounceSteps$ times. Satisfying the test $returnSteps < 2 * bounceSteps$ means F has either made progress of less then $bounceSteps$, or it has made one or more whole loops and then less than $bounceSteps$. In the first case F has been blocked during one of the rounds when B has been blocked; this can only happen if they had been blocked on the same edge, i.e. the ring has been explored. In the latter case, the ring has obviously been explored.

Consider now the cases when one agent terminates first (the case of agents never meeting is handled by Lemma 1):

- Agent B terminates due to timeout $Ntime \geq 3n$: As the agents are moving in the same direction, the number of successful moves differs by at most $n - 1$. Since in each time step at least one of them advances, in less than $3n$ time steps from the moment when B learned n, F will also learn n and eventually terminate.
- Agent F terminates due to timeout $Ntime \geq 5n$: Since the agents did not cross each other (satisfying the *meeting* predicate), at most $2n - 1$ time units from the moment when F learned n, agent B switches to state Return. Now the analogous argument as in the previous case applies, and agent B will learn n with at most $3n$ additional steps, and eventually terminate.

□

Theorem 5. *Algorithm* LANDMARKWITHCHIRALITY *allows two anonymous agents with chirality to explore a 1-interval connected ring with a landmark and to terminate in $O(n)$ time.*

Proof. If the agents do not catch each other, the proof follows from Lemma 1. Consider now the case that the agents catch each other at least once. Also by Lemma 1, we know that the meeting will happen no later than in round $7n - 3$. The crucial observation is that, either the time between two consecutive meetings is linear in the progress made by agent F, or the agents terminate following the catch.

Let $pTime_i$ denote the time between i-th and $i + 1$-th catch and let $forwardSteps_i$ be the progress made in that time by agent F. We have:

$$returnSteps_i = bounceSteps_i + forwardsSteps_i$$

Furthermore,

$$pTime_i \leq 2 * bounceSteps_i + returnSteps_i + forwardSteps_i$$

Substituting *returnSteps* into the latter yields

$$pTime_i \leq 3 * bounceSteps_i + 2 * forwardSteps_i$$

If the agents do not terminate after this catch, it must be $forwardSteps_i > bounceSteps_i$, hence $pTime_i \leq 5 * forwardSteps_i$. This means that by time $5n$ at the latest since the first catch, agent F will know n and will terminate in $5n$ further rounds (if it does not terminate earlier due to some other terminating condition). The correctness now follows from Lemma 2. □

3.2.3 Landmark without Chirality

We first consider and solve the problem when both agents start from the landmark; we then adapt the algorithm to work when agents start in arbitrary positions.

Starting from the Landmark The pseudocode of Algorithm StartFromLandmarkNoChirality is in Figure 7. The main difficulty lies in the case when the agents start in opposite directions and never break the symmetry. Our approach to solve this case is to add an initial phase in which the agents use the event of waiting on a missing edge to break symmetry, obtain different IDs (of size $O(\log n)$) and then use these IDs to ensure that, if the agents do not catch each other (or outright explore the ring), then they eventually move in the same direction for a sufficiently long time so that Algorithm LandmarkWithChirality succeeds.

Let us remark that if the agents somehow catch each other, they establish chirality, and then they can use Algorithm LandmarkWithChirality which leads to exploration and termination. Therefore, if at any point the agents catch each other, they enter states Forward and Bounce and proceed with Algorithm LandmarkWithChirality.

Computing the ID: Each agent tries to compute its ID according to the procedure described below. If an agent does not succeed in computing its ID then it has explored the ring and it is aware of that.

If an agent does not know the ring size, the first two times it enters in a waiting queue, it immediately changes direction. Let r_1 and r_2 be the rounds when this occurs. Let r_3 be te time when it entered the landmark for the first time between times r_1 and r_2 (0 if it did not enter it at that time interval).

The computed ID of this agent consists of the interleaved bits of the numerical indices of rounds r_1, r_2 and r_3.

Note that two IDs are equal if and only if their r_i's are equal (the same is not necessarily true if we constructed the IDs by concatenation).

Moreover, notice that if a round r_1 or r_2 does not exist, because the agent encountered a missing edge less than two times, then that agent has looped around the landmark; in this case it enters in the Happy State (c.f. pseudocode). So it knows the ring size and it can compute an upper bound on termination time of the other agent.

Using the IDs to decide the direction: The following procedure is used when an agent has computed its ID. Agents agree on a predetermined subdivision of rounds in phases. Round r belongs to phase j, $r \in phase(j)$, iff $r \in (\sum_{i=0}^{j-1} 2^i, \sum_{i=0}^{j-1} 2^i + 2^j]$. Given the ID, an agent computes a string of bits $S(ID) = 10 \circ (b(\text{ID})) \circ 0$, where $\circ$ is the string concatenation and $b(\text{ID})$ is the minimal binary representation of ID. Given a string S we use $(S)_i$ to denote the $i-th$ character of S. Let us define as $\overline{j}$ the minimum value for which $2^{\overline{j}} \geq len(S(ID))$ and $\overline{S(ID)} = (0)^{2^{\overline{j}}-len(S(ID))} \circ S(ID)$ hold, where $len(S)$ is the lenght of the string S. For each phase $j \geq \overline{j}$ we associate the binary string $d(ID, j) = Dup(\overline{S(ID)}, 2^{(j-\overline{j})})$, where $Dup(S, k)$ is the string obtained from S by repeating each character k times, e.g. $Dup(1010, 2) = 11001100$. For each round $r \in phase(j)$, with $j > \overline{j}$, the direction of the agent is equal to $left$ if $(d(ID, j))_{r-(\sum_{i=0}^{j-1} 2^i+1)} = 0$, otherwise it is $right$.

In our algorithm this procedure is implemented using three functions:

- *set*(ID): This function takes as parameter the ID of the agent, and it initializes the aforementioned procedure.

```
States: {InitL, Happy, FirstBlockL, AtLandmarkL, Ready, Reverse, Bounce, Return, Forward, Terminate, BComm, FComm}.
In state InitL:
    $dir \leftarrow left$, $r_1 \leftarrow 0$, $r_2 \leftarrow 0$, $r_3 \leftarrow 0$
    LExplore($dir$ | $n$ is known: Happy; $Btime \geq 0$: FirstBlockL; $catches$: Bounce, $catched$: Forward)
In state Happy:
    LExplore($dir$ | $Ttime \geq 32((3\lceil \log(n) \rceil + 3)5 \cdot n) + 1$: Terminate; $catches$: Bounce; $catched$: Forward)
In state FirstBlockL:
    $dir \leftarrow right$, $r_1 \leftarrow Ttime$
    LExplore($dir$ | $n$ is known: Happy, $isLandmark$: AtLandmarkL; $Btime \geq 0$: Ready; $catches$: Bounce $catched$: Forward)
In state AtLandmarkL:
    $r_3 \leftarrow Etime$
    if both agents are at the landmark then
        Wait one round
        if both agents are at the landmark then
            Terminate
    LExplore($dir$ | $n$ is known: Happy, $Btime \geq 0$: Ready; $catches$: Bounce; $catched$: Forward)
In state Ready:
    $r_2 \leftarrow Ttime - max(r_1, r_3)$
    Compute your ID by interleaving bits of $r_1$, $r_2$ and $r_3$.
    $set$(ID)
    Change to state Reverse and process it
In state Reverse:
    $dir \leftarrow direction(Ttime)$
    if $n$ is known then
        LExplore($dir$ | $Ttime \geq 32((3\lceil \log(n) \rceil + 3)5 \cdot n)$: Terminate; $catches$: Bounce; $catched$: Forward)
    else
        LExplore($dir$ | $switch(Ttime)$: Reverse; $catches$: Bounce; $catched$: Forward)
In state Bounce, Return, Forward, BComm, FComm:
    The same as in Algorithm LandmarkWithChirality.
```

Figure 7: Algorithm StartFromLandmarkNoChirality

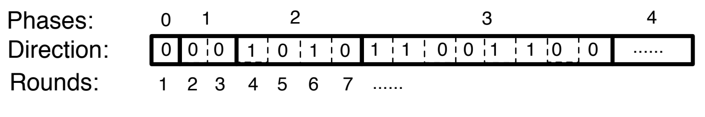


Figure 8: Directions for an agent with $ID = 1$, a round with value 0/1 corresponds to left/right direction

- *direction*$(Ttime)$: This function takes as parameter the current round and it returns the direction according to the aforementioned procedure.
- *switch*$(Ttime)$: This function takes as parameter the current round and it returns true if *direction*$(Ttime) \neq$ *direction*$(Ttime - 1)$.

Lemma 3. *Let us consider two agents with different IDs:*$\{ID, ID'\}$*, with* $len(ID) \geq len(ID')$*. For any constant* $c > 0$*, by round* $r < 32((len(ID) + 3)c \cdot n) + 1$ *there has been a sequence of* $c \cdot n$ *rounds in which the agents had the same direction. Moreover, by round* r*, each agent has moved in both directions for a sequence of rounds of length at least* $c \cdot n$*.*

Proof. By definition $\overline{j} = \lceil \log(len(ID) + 3) \rceil$. Phase $\overline{j}$ starts at round $r < 2^{\log(len(ID)+3)+2} < 4(len(ID) + 3)$.

Now let us consider $d(ID, \overline{j})$ and $d(ID', \overline{j})$. We now show that there exist two indices x, y such that $(d(ID, \overline{j}))_x \neq (d(ID', \overline{j}))_x$ and $(d(ID, \overline{j}))_y = (d(ID', \overline{j}))_y$. It is easy to verify that $y = len(d(ID, \overline{j}))$, by construction since $d(.)$ always terminate with 0. For the index x we consider two cases:

- $d_s(ID', \overline{j}) = \overline{S(ID')}$: If $len(S(ID)) > len(S(ID'))$ then x is the index of the first non zero bit of $d_s(ID, \overline{j})$. Otherwise, if $len(S(ID)) = len(S(ID'))$ in this case by assumption $ID \neq ID'$ therefore index x exists.
- $d_s(ID', \overline{j}) \neq \overline{S(ID')}$: by construction, $d_s(ID', \overline{j})$ is composed by sequences of equal bits of length at least 2. Also by construction, the first 3 bits of $S(ID)$ are equal to 101. This means that the substring 101 cannot be contained in $d_s(ID', \overline{j})$, and it is contained in $d_s(ID, \overline{j})$; this implies that index x exists.

If the agents agree on the direction, they will have the same direction in the round corresponding to index y, otherwise they will have the same direction in the round corresponding to index x.

In phase $j > \overline{j}$, by construction, we have a sequence of rounds where agents have the same direction of length at least $2^{j-\overline{j}}$. We have $2^{j-\overline{j}} > c \cdot n$ when $j > \log(len(ID) + 3) + \log(c \cdot n) + 3$. We reach the end round of this phase by $r = \sum_{i=0}^{\log(len(ID)+3)+\log(c \cdot n)+4} 2^i \leq 32((len(ID) + 3)c \cdot n)$. The last statement of the lemma derives directly by the presence of 1 and 0 in each possible $S(ID)$.

□

Theorem 6. *Algorithm* StartFromLandmarkNoChirality *allows two anonymous agents starting from the landmark without chirality to explore a 1-interval connected ring with a landmark and to terminate in* $O(n \log(n))$ *time.*

Proof. First note that if the agents catch each other, by the proof of Theorem 5, they will explore the ring and terminate in $O(n)$ time since the moment they catch. Hence, in the remainder of the proof we deal with the case when the agents never catch each other.

Second, if the agents meet at the landmark and terminate from state AtLandmark, they must have bounced from the same edge and the ring has been explored; this is because they started from the landmark and returned at the same time while both were blocked exactly once.

Third, observe that, by time $2n + 1$, either an agent knows n (and terminates in $O(n \log(n))$ time from Happy state, or it knows its own ID. Note that IDs are bound from above by n^3, since each r_i is at most n, which implies $len(ID) \leq 3\lceil \log(n) \rceil$.

Consider now the case that at time $2n + 1$ an agent (say a) does not know its ID (and hence since time $2n+1$ knows n), while the other (b) knows its ID but does not know n. Agent b therefore repeatedly switches its direction in state Reverse, while agent a moves in the same direction. Note that by Lemma 3, by time $32((3\lceil \log(n) \rceil + 3)5 \cdot n) + 1$, agent b has moved to the *left* and *right* direction for a sequence of rounds of length at least $5n$, in one of the two both a and b move in the same direction. As at least one agent makes progress in each of those time steps, while (by assumption) they don't catch each other, b must have moved for at least $2n$ time units. This means that b learns n and eventually terminates as well.

The final case to consider is when both agents know their IDs, but do not know n. Note that if the agents have the same values of r_1 and r_2, they must have covered the whole ring and at least one of them will have

```
States: {Init, AtLandmark, FirstBlock, FirstBlockL, AtLandmarkL, Ready, Reverse, Bounce, Return, Forward, Terminate, BComm, FComm}.
In state Init:
    $dir \leftarrow left$, $r_1 \leftarrow 0$, $r_2 \leftarrow 0$, $r_3 \leftarrow 0$
    LExplore($dir \mid n$ is known: Happy; $Btime \geq 0$: FirstBlock; $catches$: Bounce; $catched$: Forward)
In state FirstBlock:
    $dir \leftarrow right$, $r_1 \leftarrow Ttime$
    LExplore($dir \mid n$ is known: Happy; $isLandmark$: AtLandmark; $Btime \geq 0$: Ready; $catches$: Bounce; $catched$: Forward)
In state AtLandmark:
    if both agents are at the landmark then
        $r_1 \leftarrow 0$, $r_2 \leftarrow 0$, $r_3 \leftarrow 0$, $Ttime \leftarrow 0$
        LExplore($dir \mid n$ is known: Happy, $Btime \geq 0$: FirstBlock; $catches$: Bounce; $catched$: Forward)
    else
        $r_3 \leftarrow Etime$
        LExplore($dir \mid n$ is known: Happy; $Btime \geq 0$: Ready; $catches$: Bounce; $catched$: Forward)
In state $S \notin$ {Init, FirstBlock, AtLandmark}:
    The same as in Algorithm StartAtLandmarkNoChirality.
```

Figure 9: Algorithm LandmarkNoChirality

$r_3 \neq 0$. This means that the agents necessarily have different IDs, since if they had the same values of r_1 and $r_3 \neq 0$, they would have terminated in AtLandmark state.

Since the IDs are different, by Lemma 3, by round $32((3\lceil\log(n)\rceil + 3)5 \cdot n) + 1$ there has been a time segment of length $5n$ in which both agents were moving in the same direction. Thus, either they catch each other, or both learnt n and terminated thereafter. □

Arbitrary initial positions Algorithm StartAtLandmarkNoChirality almost works also in the case of agents starting in arbitrary position. The only failure would be due to the fact that, when the agents meet in the landmark while establishing r_1 and r_2, it does not necessarily mean that they have already explored the ring. The modification to introduce is not to terminate in this case, but to reset and start a new instance in state InitL, executing algorithm StartAtLandmarkNoChirality, as now the agents are indeed starting at the landmark. If the agents do not meet at the landmark, then their values of r_3 are different and the algorithm works using the same arguments. The complete pseudocode is in Figure 9. Since this adds at most $O(n)$ to the overall time, we obtain the following theorem.

Theorem 7. *Algorithm* LandmarkNoChirality *allows two anonymous agents starting in any initial position and without chirality to explore a 1-interval connected ring with a landmark and to terminate in* $O(n\log(n))$ *time.*

4 Ring exploration in $\mathcal{SSYNC}$

In this section we investigate the exploration problem when the system is *semi-synchronous*. The complexity measure we consider in this case is the total number of edges traversed by the agents. Let us begin by showing an intuitive result for the weak NS model:

Theorem 8. *In the* NS *model, exploring the ring is impossible with any number of agents, regardless of their computational capabilities and the orientation of the ring.*

Proof. (sketch) Let us consider a ring where all agents start in node v_0. Initially, the adversary removes the "right" edge leading to v_{-1} and pauses all the agents that, if active, would have moved there. So no agent leaves v_0 in the first round. In the next round, the adversary pauses the agents that (if active) would move left and removes the right edge. By continuing this alternating process, the adversary prevents the agents to explore other nodes besides v_0. ☐

Motivated by this impossibility result, we now examine the other $\mathcal{SSYNC}$ models.

4.1 Exploration in the PT Model

We begin our investigation of the PT Model by showing that without chirality two agents cannot explore the ring, even with precise knowledge of the network size and with the presence of a landmark.

To understand this impossibility, consider the behaviour of an agent a, executing a solution algorithm for a fixed ID and network size, in the following scenario:
Scenario: (i) let u be the starting node of the agent and let u' be the neighbour of u towards which the agent initially decided to depart; (ii) whenever a tries to cross an edge different from (u, u'), that edge is blocked; neither u nor u' are the landmark, nor does the other agent enter these nodes.

In this scenario, there are two possible behaviours:

- Eventually Fixed Direction: Eventually, the agent decides to wait indefinitely on an edge until it becomes available

- Perpetual Switching: The agent forever keeps switching direction on blocked edges (with possibly different timeouts).

Lemma 4. *In a ring of at least five nodes, if the algorithms for both agents have Perpetual Switching behaviour, then the agents cannot explore the ring.*

Proof. W.l.o.g. assume that agent a starts at u, going towards u' and agent b starts at v going towards v'. The adversary can select u and v in such manner that u, u', v, v' and the landmark are all different.

The adversary first makes a active and b passive and lets a travel until it tries to leave $\{u, u'\}$. At this moment, it will block the edge that a is trying to cross until a decides to change direction and enters edge (u, u') (possibly from u'). Then, the adversary makes a passive, and activates b. Agent b has been passively transported to v' and eventually tries to exit $\{v, v'\}$. The adversary blocks the edge via which b tries to exit until b changes direction and enters (v, v'). At this moment the adversary makes b passive and activates a (which has passively crossed edge (u, u') and the initial configuration is repeated. As this can be infinitely repeated, the agents never manage to exit nodes u, u', v, v'. ☐

Theorem 9. *In the* PT *model without chirality two anonymous agents with finite memory are not sufficient to explore a ring. The result holds even if there is a distinguished landmark node and the exact network size is known to agents.*

Proof. W.l.o.g. assume that agent a starts at u, going towards u' and b starts at v going towards v'. The adversary can select u and v in such a way that u, u', v, v' and the landmark are all different. Since the agents are anonymous, their algorithms are either both **Perpetual Switching**, or both **Eventually Fixed Direction**. By Lemma 4 in the first case the agents cannot explore the ring; hence it is sufficient to consider the case of both agents being **Eventually Fixed Direction**. Let $(u*, u'')$, where $u* \in \{u, u'\}$ is the edge on which a would eventually start indefinite waiting if the edges exiting $\{u, u'\}$ were always blocked, and let $(v*, v'')$ be such an edge for b. Since the agents have no chirality and they never enter u'' and v'', the adversary can choose u, v and an initial orientation of the agents in such a way that $u* = v''$ and $v* = u''$. The adversary starts by making agent b passive and letting agent a be active, but always blocking the edges leaving $\{u, u'\}$ until it enters (u, u') for the last time before starting indefinite wait on $(u*, u'')$. Now, it makes a passive and activates b but always blocking the edges leaving $\{v, v'\}$ until b starts the indefinite wait on $(v*, v'')$, and now it activates a (b is still active) and blocks edge $(u*, v*)$ forever. Since both a and b are waiting indefinitely, the algorithm will never explore the ring. ☐

As a consequence of Theorem 9, the following algorithms will either use chirality, or will employ three agents. In the algorithms we present, at *least one* agent always explicitly terminates, the other agents will either terminate or will stop moving.

```
$tSL \leftarrow 0$                                                    ▷ totalStepsLeft
In state Init:
    EXPLORE($left$ | $Esteps \geq N$: Terminate, $catches$: Bounce)
In state Bounce:
    if $totalStepsLeft = 0$ then
        $tSL \leftarrow Esteps$
    else
        $leftSteps \leftarrow Esteps$
        if $rightSteps \geq leftSteps$ then
            Terminate
        else
            $tSL \leftarrow tSL + (leftSteps - rightSteps)$
            if $tSL \geq N$ then
                Terminate
    EXPLORE($right$ | $Esteps \geq N$: Terminate, $Btime > 0$: Reverse)
In state Reverse:
    $rightSteps \leftarrow Esteps$
    EXPLORE($left$ | $Esteps \geq N$: Terminate, $catches$: Bounce)
```

Figure 10: Algorithm PTBoundWithChirality

4.1.1 Chirality and Knowledge of an Upper Bound

The algorithm for two agents with chirality and knowledge of an upper bound is given in Figure 10.

Theorem 10. *Two agents executing Algorithm* PTBoundWithChirality *in the* PT *model with a known upper bound N on the size of the ring and with chirality will explore the ring using at most $O(N^2)$ edge traversals. Furthermore, one agent explicitly terminates, while the other either terminates or it waits perpetually on a port.*

Proof. First we prove exploration. Note that variable *tSL* maintains (after the first bounce) the total distance traveled left from the initial position. Hence, if either $Esteps$ or *tSL* exceeds N, the ring has been explored. The only non-trivial case is the termination due to $rightSteps \geq leftSteps$] condition.

Note that if agent a stayed at the same place while b bounced off it, reversed direction on a blocked edge and returned to a, b must have bounced on the same edge on which a had been blocked (otherwise the PT condition would have ensured passive transport of a). However, in such a case the ring has been explored. If $rightsteps < leftSteps$ then, since a and b were moving in opposite direction after the last bounce of b, this could have happened only if a and b crossed each other, i.e. they have explored the ring after the last bounce of b.

We now have to show that at least one agent terminates. If the adversary keeps an edge perpetually removed, eventually the algorithm terminates due to condition $rightsteps = leftSteps$. Moreover, if an agent is not blocked in its traversal it will eventually do N steps leading to termination. The only possibility that we need to analyze is if a is blocked on edge e_0, b bounces first on edge e_0, then on edge e_x and, when b catches on a, it holds that $rightSteps < leftSteps$. When this happens, both a and b have done at least one step further to the left. Therefore, reiterating this case we will eventually have $tSL > N$ for one of the two. If an agent terminates, the other one cannot bounce to the right. Hence, it will either terminate due to exceeding N left moves, or will be perpetually blocked on a port and the last part of the theorem holds.

Complexity Discussion: Observe that during one Bounce-Reverse phase an agent can do $O(N)$ steps. There could be at most N of these Bounce-Reverse phases: in each of them the agent has to do an additional step left otherwise the termination condition is satisfied. Since the termination check bounds the total number of left steps by N, this yields $O(N^2)$ complexity of Algorithm PTBoundWithChirality. □

Theorem 11. *Let us consider the* PT *model with chirality in which two agents know an upper bound N on the ring size. In any terminating algorithm an agent does at least $\Omega(N \cdot n)$ movements.*

Proof. Let X denote a continuous set of nodes of size $\lceil n/2 \rceil$ containing the initial positions of the agents.

The adversary will block the edges leaving X until the agents try to leave X via the edges on both ends of X. At that moment, the adversary considers what would the agents do if they found the edge they are trying to cross blocked indefinitely. If both of them wait indefinitely, the adversary will chose a ring of size $|X|$ so that both agents wait on the same edge and never terminate (although they have explored the ring).

If one of them (w.l.o.g. a) reverses and crosses to the opposite side of X after a finite number of blocks, then b will be made passive and a will be active until it crosses to b. Note that this costs $O(n)$ moves and, when a catches up with b, b has been passively moved. At this moment, X is shifted one position in the direction of the b's move, both agents are activated and the process repeats.

Finally, if one agent (w.l.o.g. a) terminates while the total explored size is less then $N - 1$ (assuming running on ring of size N), then b will be made passive until a terminates, and then every b's move will be blocked, preventing further exploration – the algorithm fails on rings of size N.

Since each cycle costs $O(n)$ and expands the "virtual" (in ring of size N) explored size by 1 (except the first cycle which explores $\lceil n/2 \rceil$ nodes), the total cost is $O(Nn)$. □

4.1.2 Landmark with Chirality

The algorithm is essentially a variation of the previous, where an agent terminates also when it loops around the landmark. The proof follows the same lines of Th. 10.

4.1.3 No Chirality and Knowledge of an Upper Bound

In this case, we employ three agents, two of which will necessarily agree on the direction. The algorithm is described in Figure 11. An agent always bounces when catching another agent, performing a zig-zag tour. In doing so, it counts the number of steps it took in each direction, and if these lengths stop increasing (or even decrease), it terminates. This termination check is done not only when bouncing, but also when encountering an agent in a node (which might be a passively transported agent).

```
$d \leftarrow 0$
In state Bounce:
    CheckD($Esteps$)
    Explore($right$ | $Esteps \geq N$: Terminate, $meeting$: MeetingB, $catches$: Reverse)
In state Reverse:
    if $d = 0$ then
        $d \leftarrow Esteps$                              ▷ Fist time I change state from Bounce to Reverse
    else
        CheckD($Esteps$)
    Explore($left$ | $Esteps \geq N$: Terminate, $meeting$: MeetingR, $catches$: Bounce)
In state MeetingR:
    CheckD($Esteps$)
    Explore($left$ | $Esteps \geq N$: Terminate, $catches$: Bounce)
In state MeetingB:
    CheckD($Esteps$)
    Explore($right$ | $Esteps \geq N$: Terminate, $catches$: Bounce)
In state Init,Bounce:
    As in Algorithm PTBoundWithChirality

function CheckD(x)
    if $d > 0$ then
        if $Esteps \leq d$ then
            Terminate
        else
            $d \leftarrow Esteps$
```

Figure 11: Algorithm PTBoundNoChirality

Lemma 5. *In Algorithm* PTBoundNoChirality*, if an agent terminates then the ring has been explored.*

Proof. The only non trivial part is the terminating condition [**if** $Esteps \leq d$] in function CheckD. Let a, b, c be the tree agents. W.l.o.g we consider the first round after which two agents are going left (a and b), and c is going right. This has to happen otherwise the agents just keep looping around the ring until one terminates. Consider the first round r_0 in which an agent changed direction on another agent. W.l.o.g. assume b bounced on a in node x_0. Let r be the first round such that c is between a and b (including the position of a and b). Since a and b were moving in different directions and could not bounce on each other, there was no bounce between r_0 and r. Observe that the area between a and b containing x_0 has been explored (let us ignore in the remainder the area explored by c up to this moment), and from this moment on, there is always the leftmost agent expanding this area to the left, the rightmost expanding it to the right and the middle agent traveling between them. Note that the agents can overtake each other (due to passive transport) and change who is the leftmost/rightmost/middle, however there is always at most one middle one.

Note that as long as the endpoints of the explored area do not cross each other (in which case the ring has been fully explored), $d > Esteps$: The check can only be performed on the boundary of the explored area as in the middle there is only one agent. Condition $d \geq Esteps$ follows from the fact that d is the length of the second last crossing and $Esteps$ is the length of the last crossing of this agent. Condition $d = Esteps$ would mean that the explored area did not meanwhile grow, which, from the PT condition, implies that both agents on which this agent bounced were blocked on the same edge (otherwise there would have been passive transport and d would have grown), i.e. the ring is explored. □

Theorem 12. *Three anonymous agents performing Algorithm* PTBoundNoChirality *in the* PT *model with a known upper bound on the ring size and no chirality, explore the ring with $O(N^2)$ edge traversals. One agent explicitly terminates, the other either terminates or waits perpetually on a port.*

Proof. The correctness of termination derives from Lemma 5. It remains to prove that eventually at least one agent terminates. Having three agents, at least two will agree on the same direction. We will consider this direction as global *left*. It is easy to see that if an edge is perpetually removed, then eventually the agents terminate: two agents will be positioned at the end point of the missing edge and the third agent terminates detecting $Esteps = d$. If an agents is not forced to change direction and the edges are not perpetually removed, then it will terminate since $Esteps > N$. Therefore, the adversary has to force the agents to bounce on each other. But let us notice that, as soon as an agent changes state from Bounce to Reverse, it sets a distance d; if this distance does not increase at each state change, the agent terminates. This implies that eventually we will have $d > N$ and termination for $Esteps > N$.

Complexity Discussion: If an agent does not set d, then it performs at most $O(N)$ steps. If an agent sets d, its value is at most $O(N)$; there are at most $O(N)$ increases of d, therefore an agent will do at most $O(N^2)$ movements. Since the number of agents is constant, the total sum of movements over all agents is at most $O(N^2)$. □

4.2 Exploration in the ET Model

Let us first introduce a simple result on perpetual exploration:

Theorem 13. *In the* ET *model with chirality, two robots are sufficient to explore a ring.*

Proof. A trivial perpetual motion adaptation of the Algorithm explained in Section 3.1.1 solves the exploration in ET. □

Given the previous results, a natural question is whether there is an algorithm with at least one agent terminating, as we have shown for the PT model. Unfortunately the following theorem shows that, without exact knowledge of the network size, it is impossible to design such an algorithm.

Theorem 14. *Let us consider the* ET *model with chirality where only an upper bound on the ring size is known. Given any number of agents with finite memory, there does not exist any exploration algorithm where an agent terminates in a bounded number of rounds, signalling the exploration of the ring. This holds true even if the ring has a landmark node and/or the agents have distinct IDs.*

Proof. Let us consider two different rings R_1 and R_2 of size n and n' respectively, where $n < n' < N$. On ring R_2 the adversary places all agents between e_0 and e_n and blocks edges e_0 and e_n at distance n from each other: In rounds where the agents try to enter at most one of e_0 and e_n, the adversary will block that edge. In the *busy* rounds where there are agent entering both e_0 and e_n, it will alternate between making agents waiting on e_0 passive and block edge e_n and making agents waiting on e_n passive and blocking e_0. On ring R_1 the adversary perpetually removes edge e_0 and keeps all agents active except in the busy rounds (agents trying to enter e_0 from both directions) where it alternates sides of e_0 on which it is making the agents passive.

Observe that in such a scenario, the agents cannot distinguish between R_1 and R_2 and either none terminates, or at least one agent terminates (in which case the adversary will put then in R_2).

Note that the above arguments apply also if the agents have distinct IDs. If there is a landmark, it can be placed anywhere in R_1, and in the corresponding place between e_0 and e_n in R_2, R_1 and R_2 will still look the same to the agents. □

4.2.1 Exploration Algorithm for ET

Algorithm ETBoundNoChirality is a direct adaptation of Algorithm PTBoundNoChirality, the only differences are that N is set to $n - 1$ (since from Theorem 14 we know the size needs to be known precisely), and the inequality check in CheckD becomes strict: (**if** $Esteps < d$). As in the PT model, three agents are employed, with no chirality assumption.

Theorem 15. *Three anonymous agents performing Algorithm* ETBoundNoChirality *in the* ET *model with known ring size and no chirality explore the ring, with one agent explicitly terminating and the other agents either terminating or waiting perpetually on a port.*

Proof. Let us first observe that if an agent terminates, then it terminates correctly. The proof follows the same steps as the one for Lemma 5. The only difference is that in ET the CheckD requires (**if** $Esteps < d$), thus the part of the proof that uses the PT assumption to handle the case $Esteps = d$ is not needed anymore.

What remains to be shown is that eventually at least one agent terminates. We show this by contradiction. Let us notice that if an edge is perpetually removed, an agent eventually terminates: two agents will be positioned at the two ports of the missing edge and the other one will do exactly $Esteps = n - 1$ steps going from one endpoint node to the others. If an edge is not perpetually removed then, by construction (the agents bounce only on other agents, not on timeout on blocked edges) and by the ET condition, an agent that is waiting on the port of that edge will eventually traverse it. Since the agents terminate if they traverse $n - 1$ steps in the same direction, the only possibility left to consider is for the adversary to force the agents to perpetually catch each other, changing directions without increasing the d value. In such a case there is a round r^* after which each agent x reaches a certain stable value d_x and it always changes position at the same points l_x and r_x; furthermore, there exists a round $r \geq r^*$ in which two agents go left and one agent goes right. In the remainder we assume the execution is in round at least r^*.

Let a, b, c be the agents and let Lxy indicate the event: agent x catches agent y at l_x (and changes direction from left to right); the event Rxy is analogous. Let $Dxy : D'x'y'$ denote the statement that the event $D'x'y'$ is the catching event immediately following Dxy.

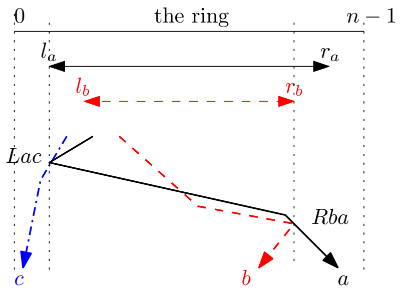
Note that the Dxy event can only be followed by an $\overline{D}xz$ or $\overline{D}zx$ events (here and in the remainder, we implicitly assume $x \neq y \neq z$), where $D \in \{L, R\}$ and $\overline{D}$ is the direction opposite to D. This is so because, after Dxy, agents x and z were moving in the same direction $\overline{D}$, y remained moving in direction D, and only agents that move in the same direction can catch each other. Hence, the possible sequences of direction changes of a perpetual schedule can be represented by a binary tree rooted at the initial Dxy event (w.l.o.g. Lab or Lac), with leaves corresponding to agent termination. We call this tree *Catch Tree*. The assumption that the algorithm does not terminate implies that there is an infinite path in this tree. In the remainder of this proof we will show that there is no such path in the *Catch Tree*.

First, observe that the agents never meet on a node – that would imply at least one of them performing the termination check somewhere inside its range, immediately terminating. This means that unlike the case of Algorithm PTBoundNoChirality we do not have to worry about the agents overtaking each other. (They can still cross each other without noticing when traveling in opposite direction and crossing the same edge at the same time.)

Consider now a sequence of events $Dxy : \overline{D}xz : Dxy$. This corresponds to x bouncing off y and z while those remained stationary. From the ET condition and the fact that y and z are waiting on different edges (otherwise x would detect $d = n - 1$ and terminate) we know that eventually either y or z makes progress, hence this sequence cannot repeat indefinitely and eventually a different event must occur. This can be represented in the *Catch Tree* by removing the bottom Dxy (and its whole subtree) and adding an arrow from its parent to the top Dxy (implying that the same configuration repeats).

Let A, B, C be the ranges (sets of nodes an x agent visits between l_x and r_x) of agents a, b, c, respectively, and $\overline{A}, \overline{B}, \overline{C}$ be their complements.

Lemma 6. *If the algorithm does not terminate, we have* $\forall X, Y \in \{A, B, C\}, X \neq Y : \overline{X} \cap \overline{Y} = \emptyset$.



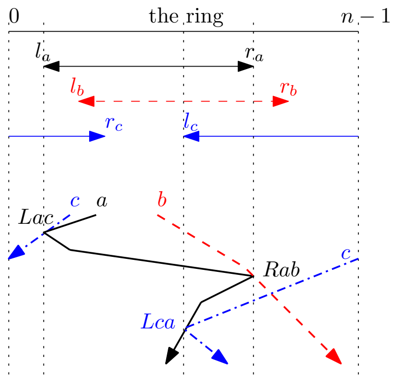


Figure 12: Left: The agent ranges and events for the case $Lac : Rba$. Right: the case $Lac : Rab$.

Proof. The proof is by contradiction. Assume w.l.o.g, A, B such that $\overline{A} \cap \overline{B} \neq \emptyset$. From this and symmetry, we may assume $l_a \notin B$. This means Lac is the only way for the agent a to change direction while going left. Consider the possible next event (consult Figure 12, left). It cannot be Rba: for that r_b must be in A, which combined with $l_a \notin B$ implies $\overline{A} \subset \overline{B}$. Since from the moment of Lac agent a was moving right, starting outside B and to the left of it, in order to bounce off b it would need to overtake it, which is impossible.

Hence, Lac must be followed by Rab, which in turn might be followed by Lac or Lca. Lac is the cyclic case which, as has been discussed above, cannot repeat indefinitely and can therefore be ignored. The only possible case is Lca, which can only happen if $\overline{C} \subset A$.

Observe the movement of the agents (see Figure 12, right: From Rab on, b is moving right from r_a, while from Lca on, c is moving right from l_c. This means that Rbc is impossible, because c would have had to overtake b. However, Rcb is also impossible as b cannot get into position at r_c without changing direction or crossing $\overline{B}$. □

From Lemma 6 it remains to consider the case of pairwise disjoint range complements, as shown in Figure 13.

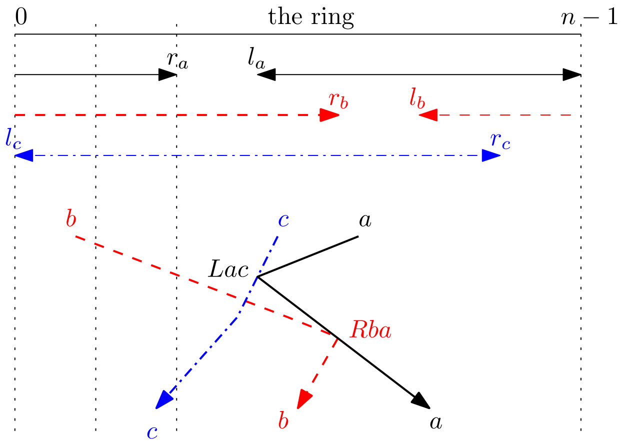


Figure 13: The case of $Lca : Rba$ for disjoint range complements.

Lemma 7. *If the algorithm never terminates, then Lac cannot be immediately followed by Rba.*

Proof. After event Lac, we have that c is moving left from l_a. After event Rba, agent b is moving left from r_b. The next event is either Lbc or Lcb. However, Lbc is impossible, as for c to reach l_b without changing direction it would have to cross $\overline{C}$. Event *Lcb* is also impossible, because *b* would have to overtake *c* to get into position to be bounces off. □

The following corollary is immediately obtained from Lemma 7 by rotation and symmetry:

Corollary 2. *If the algorithm never terminates, then the following sequences are forbidden Lac : Rba, Lba : Rcb, Lcb : Rac, Rbc : Lab, Rca : Lbc, Rab : Lca.*

At some point in the perpetual schedule we should have either *Lab* or *Lac*. As can be seen in Figure 14, any branch of a Catch Tree starting from these configuration leads either to a cycle or to a forbidden sequence of events, leading to a contradiction with the assumption that the algorithm never terminates.

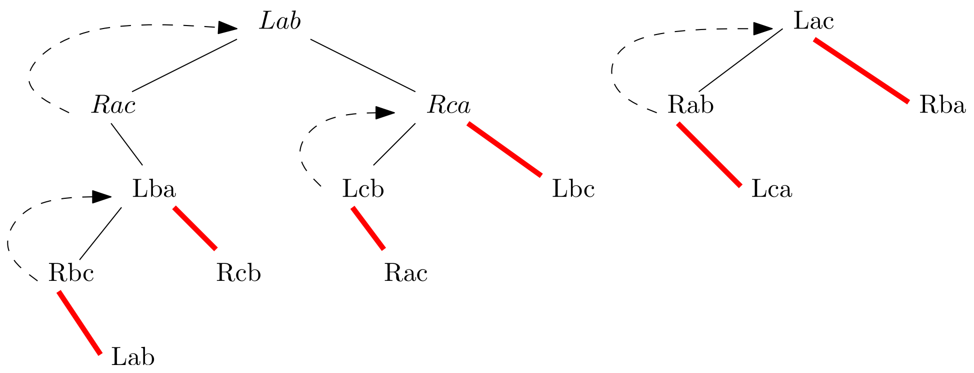


Figure 14: Catch Trees rooted at *Lab* and *Lac*. The red edges denote forbidden sequences, while the dashed ones depict the loops.

Complexity Discussion: Agents may do a finite but unbounded number of moves before termination: when two agents are blocked going on opposite directions on two different edges, the third agent goes back and forth between them. In the ET condition there is no bound on the number of rounds in which this configuration can be kept. □

5 Conclusion

In this paper we started the investigation of the distributed exploration problem for *1-interval-connected* dynamic graphs by focusing on rings in fully synchronous and semi-synchrnous environments. We studied the impact that structural information and knowledge can have on the solvability of the problem. In particular, we considered such factors as: knowledge of the exact size of the ring or of an upper bound, agreement on orientation, anonymity.

References

[1] S. Albers and M. Henzinger. Exploring unknown environments. *In* Proc. of the 29th ACM Symp. on Theory of Computing (STOC), pages 416–425, 1997.

[2] L. Arantes, F. Greve, P. Sens, and V. Simon. Eventual leader election in evolving mobile networks. *In* Proc. of the 17th Int. Conference on Principles of Distributed Systems (OPODIS), pages 23–37, 2013.

[3] J. Augustine, G. Pandurangan, and P. Robinson. Fast byzantine agreement in dynamic networks. *In* Proc. of the 32th Symposium on Principles of Distributed Computing (PODC), pages 74–83, 2013.

[4] C. Avin, M. Koucky, and Z. Lotker. How to explore a fast-changing world. *In* Proc. of the 35th International Colloquium on Automata, Languages and Programming (ICALP), pages 121–132, 2008.

[5] B. Awerbuch and S. Even. Efficient and reliable broadcast is achievable in an eventually connected network. *In* Proc. of the 3th Symposium on Principles of Distributed Computing (PODC), pages 278–281, 1984.

[6] B. Balamohan, S. Dobrev, P. Flocchini, and N. Santoro. Exploring an unknown dangerous graph with a constant number of tokens. Theoretical Computer Science, (to Appear), 2016.

[7] H. Baumann, P. Crescenzi, and P. Fraigniaud. Parsimonious flooding in dynamic graphs. *In* Proc. of the 28th Symposium on Principles of Distributed Computing (PODC), pages 260–269, 2009.

[8] M. Biely, P. Robinson, and U. Schmid. Agreement in directed dynamic networks. *In* Proc. of the 19th International Colloquium on Structural Information and Communication Complexity (SIROCCO), pages 73–84, 2012.

[9] B. Brejova, S. Dobrev, R. Kralovic, and T. Vinar. Efficient routing in carrier-based mobile networks. Theoretical Computer Science, 509:113–121, 2013.

[10] B. Bui-Xuan, A. Ferreira, and A. Jarry. Computing shortest, fastest, and foremost journeys in dynamic networks. International Journal of Foundations of Computer Science, 14(2):267–285, 2003.

[11] A. Casteigts, P. Flocchini, B. Mans, and N. Santoro. Measuring temporal lags in delay-tolerant networks. IEEE Transactions on Computers, 63(2):397–410, 2014.

[12] A. Casteigts, P. Flocchini, B. Mans, and N. Santoro. Shortest, fastest, and foremost broadcast in dynamic networks. International Journal of Foundations of Computer Science, (to appear) 2015.

[13] Arnaud Casteigts, Paola Flocchini, Walter Quattrociocchi, and Nicola Santoro. Time-varying graphs and dynamic networks. Int. Journal of Parallel, Emergent and Distributed Systems, 27(5):387–408, 2012.

[14] A.E.F. Clementi, A. Monti, F. Pasquale, and R. Silvestri. Information spreading in stationary markovian evolving graphs. IEEE Transactions on Parallel and Distributed Systems, 22(9):1425–1432, 2011.

[15] C. Cooper, R. Klasing, and T. Radzi. Searching for black-hole faults in a network using multiple agents. *In* Proc. of the 10th International Conference on Principles of Distributed Systems (OPODIS), pages 320–332, 2006.

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

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

[18] G. Di Luna and R. Baldoni. Brief announcement: Investigating the cost of anonymity on dynamic networks. *In* Proc of the the 34th Symposium on Principles of Distributed Computing, pages 339–341, 2015.

[19] G. Di Luna, R. Baldoni, S. Bonomi, and I. Chatzigiannakis. Conscious and unconscious counting on anonymous dynamic networks. *In* Proc. of the 15th International Conference on Distributed Computing and Networking (ICDCN), pages 257–271, 2014.

[20] Y. Dieudonn and A. Pelc. Deterministic network exploration by anonymous silent agents with local traffic reports. ACM Transactions on Algorithms, 11(2):Article No. 10, 2014.

[21] S. Dobrev, P. Flocchini, R. Kralovic, and N. Santoro. Exploring a dangerous unknown graph using tokens. Theoretical Computer Science, 472:28–45, 2013.

[22] S. Dobrev, P. Flocchini, G. Prencipe, and N. Santoro. Searching for a black hole in arbitrary networks: Optimal mobile agents protocolss. Distributed Computing, 19(1):1–19, 2006.

[23] T. Erlebach, M. Hoffmann, and F. Kammer. On temporal graph exploration. *In* Proc. of 42th International Colloquium on Automata, Languages, and Programming (ICALP), pages 444–455, 2015.

[24] A. Ferreira. Building a reference combinatorial model for MANETs. *IEEE Network*, 18(5):24–29, 2004.

[25] P. Flocchini, M. Kellett, P. Mason, and N. Santoro. Map construction and exploration by mobile agents scattered in a dangerous network. *In* Proc. of the 24th IEEE International Parallel and Distributed Processing Symposium (IPDPS), pages 1–10, 2009.

[26] P. Flocchini, M. Kellett, P. Mason, and N. Santoro. Searching for black holes in subways. Theory of Computing Systems, 50(1):158–184, 2012.

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

[28] Paola Flocchini, Giuseppe Prencipe, and Nicola Santoro. *Distributed Computing by Oblivious Mobile Robots*. *Synthesis Lectures on Distributed Computing Theory*, 3(2):1–185, 2012.

[29] P. Fraigniaud, D. Ilcinkas, G. Peer, A. Pelc, and D. Peleg. Graph exploration by a finite automaton. *In* Proc. of the 29th Symposium on Mathematical Foundations of Computer Science (MFCS), pages 451–462, 2004.

[30] E. Godard and D. Mazauric. Computing the dynamic diameter of non-deterministic dynamic networks is hard. *In* Proc. of the 10th Symposium on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics (ALGSENSORS), pages 88–102, 2014.

[31] F. Harary and G. Gupta. Dynamic graph models. Mathematical and Computer Modelling, 25(7):79–88, 1997.

[32] D. Ilcinkas, R. Klasing, and A.M. Wade. Exploration of constantly connected dynamic graphs based on cactuses. *In* Proc. 21st International Colloquium Structural Information and Communication Complexity (SIROCCO), pages 250–262, 2014.

[33] D. Ilcinkas and A.M. Wade. On the power of waiting when exploring public transportation systems. *In* Proc. 15th International Conference on Principles of Distributed Systems (OPODIS), pages 451–464, 2011.

[34] D. Ilcinkas and A.M. Wade. Exploration of the t-interval-connected dynamic graphs: the case of the ring. *In* Proc. 20th International Colloquium on Structural Information and Communication Complexity (SIROCCO), pages 13–23, 2013.

[35] F. Kuhn, N. Lynch, and R. Oshman. Distributed computation in dynamic networks. *In* Proc. of the 42th Symposium on Theory of Computing (STOC), pages 513–522, 2010.

[36] F. Kuhn, Y. Moses, and R. Oshman. Coordinated consensus in dynamic networks. *In* Proceedings 30th Symposium on Principles of Distributed Computing (PODC), pages 1–10, 2011.

[37] C. Liu and J. Wu. Scalable routing in cyclic mobile networks. IEEE Transactions on Parallel and Distributed Systems, 20(9):1325–1338, 2009.

[38] Othon Michail and Paul G. Spirakis. Traveling salesman problems in temporal graphs. *In* Proc. of the 39th International Symposium on Mathematical Foundations of Computer Science (MFCS), pages 553–564, 2014.

[39] R. O'Dell and R. Wattenhofer. Information dissemination in highly dynamic graphs. *In* Proc. of the Joint Workshop on Foundations of Mobile Computing (DIALM-POMC), pages 104–110, 2005.

[40] P. Panaite and A. Pelc. Exploring unknown undirected graphs. Journal of Algorithms, 33:281–295, 1999.

[41] Claude Shannon. Presentation of a maze-solving machine. *In* Proc. of the 8th Conf. of the Josiah Macy Jr. Foundation (Cybernetics), pages 173–180, 1951.