Live Exploration of Dynamic Rings

G.A. Di Luna
University of Ottawa
Ottawa, Canada
gdiluna@uottawa.ca

S. Dobrev
Slovak Academy of Sciences
Bratislava, Slovakia
stefan.dobrev@savba.sk

P. Flocchini
University of Ottawa
Ottawa, Canada
paola.flocchini@uottawa.ca

N. Santoro
Carleton University
Ottawa, Canada
santoro@scs.carleton.ca

***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.

***Keywords*-Distributed algorithms; Mobile agents;**

I. Introduction

A. Framework

Consider a networked system, modeled as a graph, supporting mobile computational entities, called agents. The *exploration* problem requires a team of system agents to move in the graph so that, within finite time, each node is 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. This problem has been extensively investigated under a huge spectrum of different assumptions (e.g., see [8]–[10], [17]) but always assuming that the graph is *static*: its topological structure does not change during the exploration, except for occasional faults.

Recently, distributed computing researchers have started to investigate environments where the graph is highly *dynamic*, that is where the topology changes continuosly (see [5], [24] for recent surveys). This research is motivated by the rapid development of networked systems (e.g., ad-hoc wireless mobile networks) where changes are not anomalies (e.g., faults) but rather integral part of the nature of the system.

The studies of distributed computations in highly dynamic networks (e.g., see [1], [3], [4], [7], [11], [13] [15], [18]) clearly make strong assumptions in order to restrict the universe of the possible topological changes and their temporal occurence. A common assumption made in the deterministic investigations is that the network is *1-interval-connected*: at each time step, the network topology may change but there is always a connected spanning subgraph, and the latency is sufficient for an agent movement or a message transmission before the next time step (e.g., [20], [25]). Some studies further assume that the connected spanning subgraph persists for $T > 1$ time steps (e.g., see [14], [19], [22], [23]), defining the sub-class of *T-interval-connected* systems.

Very little is known on the exploration of dynamic networks. On the probabilistic side, there is the early seminal work on random walks [2]. On the deterministic side, the only studies are on: the complexity of computing a foremost exploration schedule in *1-interval-connected* systems [14] or on general temporal graphs [26]; the computation of an exploration schedule for *T-interval-connected rings* [22]; and the computation of an exploration schedule for *1-interval-connected cactuses* [20].

All these studies however assume that the exploring agents have complete a-priori knowledge of all the topological changes and the times of their occurence; that is, they are *centralized post-mortem* investigations. Furthermore, they assume that the system is *fully synchronous*; that is all the entities are always active in every time step.

The only decentralized investigation is on the exploration of periodic time-varying graphs by a single agent [15]. Otherwise, for the *distributed live* case, i.e. when the location and timing of the changes are unknown to the agents, no results are known, even in the fully synchronous setting. In this paper, we start such an investigation extending it also to the more complex semi-synchronous environment.

B. 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 III), traditionally assumed in the literature; i.e., all agents are active at each round, We then introduce the notion of *semi-synchronous* systems (Section IV), 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 $O(n)$
2	No bounds on n, Landmark, No Chirality	Exploration with Termination in time $O(n \log(n))$

Fig. 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 Terminantion **impossible** (Th. 14)
	3	Known ring size, No Chirality	Exploration with Termination of one agent

Fig. 2. Results for $\mathcal{SSYNC}$ models

(e.g., [16]) 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. Due to lack of space some proofs are omitted and can be found in [12].

II. Model and Basic Limitations

A. Model and Terminology

Let $\mathcal{R} = (v_0, \dots v_{n-1})$ 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 identifier. Each node v_i is connected to its two neighbours v_{i-1} and v_{i+1} via distinctly labeled ports q_{i-1} and q_{i+1}, 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 $A = \{a_0, \dots, a_{m-1}\}$ 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 $\lambda_j(q_{i-1}) = \lambda_j(q_{k-1})$, 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 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 $direction \in \{left, right, nil\}$, 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-1} (resp., q_{i+1}) 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-1} (resp., q_{i+1}) 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, it 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.

B. Basic Impossibilities

We begin our study by showing simple impossibility results. It is interesting to notice that, without some knowledge of the ring topology, or without the asymmetry introduced by a landmark node, exploration with termination is impossible even in the fully synchronous model, and 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.*

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

Since the ring is anonymous, if its size in 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 endowed with only face-to-face communication.*

III. Ring Exploration in $\mathcal{FSYNC}$

We consider exploration when the system is fully synchronous, presenting and analyzing solutions under different assumptions on knowledge of the ring size, anonimity of the nodes, and presence of chirality. In all solutions we assume the agents cannot communicate explicitly.

Our algorithms use procedure EXPLORE ($dir \mid p_1 : s_1; p_2 : s_2; \ldots; p_k : s_k$) as a building block, where dir is either *left* or *right*, p_i is a predicate and s_i is a state. Procedure EXPLORE essentially describes an exploration in the specified direction: The agent performs Look, then evaluates the predicates from left to right. When a predicate is satisfied, the procedure exits and the agent transitions to the specified state. Otherwise it tries to Move in the specified direction and the process is repeated in the next round.

Furthermore, the following variables are maintained:

- $Ttime$, $Tsteps$: the total number of rounds and the successful moves, respectively, since the beginning of the execution.
- $Etime$, $Esteps$: 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 waiting in a queue.

In addition to predicates referring to these variables, the following predicates are used:

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

A. 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.

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 moving 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 $3N$.

In state Init:
  EXPLORE(*left* | $Ttime \geq 3N \vee Tsteps \geq N$: Terminate; *catches*: Bounce)
In state Bounce:
  EXPLORE(*right* | $Ttime \geq 3N$: Terminate)

Fig. 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 from the opposite directions. 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. ■

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.

In state Init:
  EXPLORE(*left* | $Ttime \geq 5N$: Terminate; $Btime = N$: Reverse; *catches*: Bounce; *catched*: Forward)
In state Reverse or Bounce:
  EXPLORE(*right* | $Ttime \geq 5N$: Terminate)
In state Forward:
  EXPLORE(*left* | $Ttime \geq 5N$: Terminate)

Fig. 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. Second, note that the only steps when no agent makes a move is when they are both waiting on the same edge in the opposite direction. However, as long as the ring has not been explored, this can happen only once (when they first approach each other from opposite directions; the second approaching means the ring has been explored), and for at most N consecutive rounds. Hence, exploration is guaranteed after $5N - 7$ rounds. ■

B. 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 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.

B.1 Perpetual Exploration

We present a protocol that allows two anonymous agents to perform perpetual exploration without knowing any bound on the ring size. The basic idea of Algorithm PERPETUALEXPLORATION 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.

In state Init:
  $N \leftarrow 2$, $dir \leftarrow$ *left*
  EXPLORE(dir; $Etime \geq 2N$: Reverse, *catches*: Backward, *catched*:Forward)
In state Reverse:
  $N \leftarrow 2 * N$, $dir \leftarrow opposite(dir)$
  EXPLORE(dir; $Etime \geq 2N$: Reverse, *catches*: Backward, *catched*:Forward)
In state Backward:
  EXPLORE($opposite(dir)$)
In state Forward:
  EXPLORE(dir)

Fig. 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 $t_0 \leq 4n$ it exceeds n. If 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 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 $t_0 + 2N$ and explore the ring by time $t_0 + 3N \in O(n)$. ■

B.2 Landmark and Chirality

We consider a ring with a special landmark 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 $v = v^*$. 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 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:

- The agent keeps track whether it is crossing the landmark and in which direction; furthermore, it tracks its distance from the landmark (since the moment it has encountered the landmark 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.

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

Fig. 6. Algorithm LANDMARKWITHCHIRALITY

The complete pseudocode is in Figure 6. Both agent start going left, if they catch each other, the naming is done. If they never meet, they terminate (see Lemma 1). After naming, agent F keeps going left. Agent B goes right until it is blocked for a number of rounds that is equal to two times the number of edges it has traversed ($Etime > 2Esteps$) or it does a loop on the ring. When one of these conditions is satisfied agent B goes left, and it tries to catch F. If F has done less than $Esteps$ steps to the left from its old position, then B and F have waited on the same edge. B knows this, and it can "communicate" the end of exploration to F. Intuitively, the condition ($Etime > 2Esteps$) on B forces F to do the same steps of B to the left direction, and this leads to a linear termination time (see Th. 5 and Lemma 2). If the agents cannot "communicate" because they do not meet for a certain number of rounds, then they will both know that the ring is explored and they can terminate independently (see Lemma 2).

Lemma 1. *In Algorithm* PERPETUALEXPLORATION*, 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 are in 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 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* PERPETUALEXPLORATION*, if an agent terminates, then the ring has been explored and the other agent will terminate as well.*

Theorem 5. *Algorithm* PERPETUALEXPLORATION *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. By the same lemma 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 the 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 further $5n$ rounds (if it does not terminate earlier due to some other terminating condition). The correctness now follows from Lemma 2. ∎

B.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.

B.3(a) Starting from the Landmark: The pseudocode of the Algorithm 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 $O(n^3)$) and then use these IDs to ensure that if the agents do not catch each other (or outright explore the ring), then eventually there is sufficiently long time in which they are moving in the same direction so that Algorithm LANDMARKWITHCHIRALITY succeeds.

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.

Fig. 7. Algorithm STARTFROMLANDMARKNOCHIRALITY

Let us remark that if the agents somehow catch each other, they establish chirality, thus using Algorithm LANDMARKWITHCHIRALITY will lead 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, then it immediately changes direction the first two times it enters in a waiting queue.

The computed ID of an agent consists of interleaved bits of three rounds r_1, r_2 and r_3. Where r_1 is the first round at which an agent was waiting in the queue of a missing edge, r_2 is the second round where the agent was waiting in a queue and r_3 is the 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). 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 do not exist, because the agent encountered a missing edge less than two times, then we have that the 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, in Happy state an agent will not change direction.

`Using the IDs to decide the direction.` The following procedure is used when an agent has computed

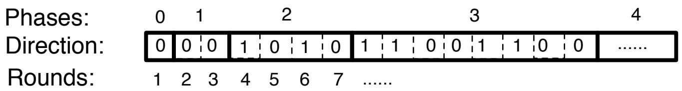

Fig. 8. Directions for an agent with $ID = 1$, a round with value $0/1$ corresponds to left/right direction

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 bit $S(ID) = 10 \circ (b(\text{ID})) \circ 0$, where $\circ$ is the string concatenation and b(ID) is the minimal binary representation of ID. Let us define as $\bar{j}$ the minimum value for which $2^{\bar{j}} \geq len(S(ID))$ and $\overline{S(ID)} = (0)^{2^{\bar{j}} - len(S(ID))} \circ S(ID)$ hold. For each phase $j \geq \bar{j}$ we associate the binary string $d_s(ID, j) = Dup(\overline{S(ID)}, 2^{(j-\bar{j})})$, where $Dup(s, k)$ is the string obtained by s repeating each character k times, e.g. $Dup(1010, 2) = 11001100$. For each round $r \in phase(j)$, with $j > \bar{j}$, the direction of the agent is equal to $left$ if $(d_s(ID, j))_{r - (\sum_{i=0}^{j-1} 2^i + 1)} == 0$, otherwise it is $right$.

In our algorithm we abstract this procedure by using three functions:

- *set*(ID): This function takes as parameter the ID of the agent, and it initializes the aforementioned procedure.
- *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* $len(ID) \geq len(ID')$*. Given* $c \cdot n$*, by round* $r < 32((len(ID) + 3)c \cdot n) + 1$ *there have 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* $c \cdot n$*.*

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 in 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 $r_3 \neq 0$. 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. ■

B.3(b) Starting from 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.*

IV. 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.*

In state Init:
$dir \leftarrow$ *left*, $r_1 \leftarrow 0$, $r_2 \leftarrow 0$, $r_3 \leftarrow 0$
LEXPLORE(dir | n is known: Happy; $Btime \geq 0$: FirstBlock; *catches*: Bounce; *catched*: Forward)
In state FirstBlock:
$dir \leftarrow$ *right*, $r_1 \leftarrow Ttime$
LEXPLORE(dir | 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 | n is known: Happy, $Btime \geq 0$: FirstBlock; *catches*: Bounce; *catched*: Forward)
else
$r_3 \leftarrow Etime$
LEXPLORE(dir | 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.

Fig. 9. Algorithm LANDMARKNOCHIRALITY

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.

A. 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.*

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 any solution must either use chirality or employ three agents. In the algorithms we present, *at east 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)

Fig. 10. Algorithm PTBOUNDWITHCHIRALITY

A.1 Chirality and Knowledge of an Upper Bound

An algorithm for two agents 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.*

A.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 as the one of Theorem 10.

A.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).

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

```
d ← 0
In state Bounce:
    CheckD(Esteps)
    Explore(right | Esteps ≥ N: Terminate, meeting: MeetingB, catches: Reverse)
In state Reverse:
    if d = 0 then
        d ← Esteps        ▷ Fist time I change state from Bounce to Reverse
    else
        CheckD(Esteps)
    Explore(left | Esteps ≥ N: Terminate, meeting: MeetingR, catches: Bounce)
In state MeetingR:
    CheckD(Esteps)
    Explore(left | Esteps ≥ N: Terminate, catches: Bounce)
In state MeetingB:
    CheckD(Esteps)
    Explore(right | Esteps ≥ N: Terminate, catches: Bounce)
In state Init,Bounce:
    As in Algorithm PTBoundWithChirality
function CheckD(x)
    if d > 0 then
        if Esteps ≤ d then
            Terminate
        else
            d ← Esteps
```

Fig. 11. Algorithm PTBoundNoChirality

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 now), and from now 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 $O(N^2)$. ■

B. **Exploration in the** ET *Model*

A trivial perpetual motion adaptation of the Algorithm explained in Section III-A1 solves the exploration also in ET.

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

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.*

We know from Theorem 14 that the size of the ring must be known. With this knowledge, it is easy to adapt Algorithm PTBOUNDNOCHIRALITY to the ET model, setting N to $n-1$, and making the inequality check in CheckD strict: (**if** $Esteps < d$). As in the PT model, three agents are employed, with no chirality assumption.

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

a) Acknowledgment: This research has been supported in part by the Natural Sciences and Engineering Research Council (Canada) under the Discovery Grant program, and by Prof. Flocchini's University Research Chair

REFERENCES

[1] J. Augustine, G. Pandurangan, and P. Robinson. Fast byzantine agreement in dynamic networks. In *Proc. of the 32nd Symp. on Principles of Distributed Computing (PODC)*, 74-83, 2013.

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

[3] A. Casteigts, P. Flocchini, B. Mans, and N. Santoro. Measuring temporal lags in delay-tolerant networks. *IEEE Trans. Comp.* 63:397–410, 2014.

[4] A. Casteigts, P. Flocchini, B. Mans, and N. Santoro. Shortest, fastest, and foremost broadcast in dynamic networks. *Int. Journal of Foundations of Computer Science*, 26(4): 499–522, 2015.

[5] A. Casteigts, P. Flocchini, W. Quattrociocchi, and N. Santoro. Time-varying graphs and dynamic networks. *Int. Journal of Parallel, Emergent and Distributed Systems*, 27(5):387–408, 2012.

[6] 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.

[7] A.E.F. Clementi, R. Silvestri, and L. Trevisan. Information spreading in dynamic graphs. *Distributed Computing* 28(1): 55-73, 2015.

[8] 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.

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

[10] K. Diks, P. Fraigniaud, E. Kranakis, and A. Pelc. Tree exploration with little memory. *Journal of Algorithms*, 51: 38-63. 2004.

[11] G.A. Di Luna, R. Baldoni, S. Bonomi, and I. Chatzigiannakis. Conscious and unconscious counting on anonymous dynamic networks. In *Proc. 15th Int. Conf. on Dist. Comp. and Networking (ICDCN)*, 257–271, 2014.

[12] G.A. Di Luna, S. Dobrev, P. Flocchini, N. Santoro. Live exploration of dynamic rings *arXiv*:1512.05306, 2015.

[13] C. Dutta, G. Pandurangan, R. Rajaraman, Z. Sun, and E. Viola. On the complexity of information spreading in dynamic networks. In *Proc. of ACM-SIAM Symp. on Discrete Algorithms (SODA)* 717–736, 2013.

[14] T. Erlebach, M. Hoffmann, and F. Kammer. On temporal graph exploration In *Proc. of 42nd Int. Coll. on Automata, Languages, and Programming (ICALP)* 444-455, 2015.

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

[16] P. Flocchini, G. Prencipe, and N. Santoro. *Distributed Computing by Oblivious Mobile Robots*. Morgan & Claypool Publishers, 2012.

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

[18] C. Gomez-Calzado, A. Lafuente, M. Larrea, and M. Raynal. Fault-tolerant leader election in mobile dynamic distributed systems. In *Proc. 19th Pacific Rim Int. Symp. on Depend. Comput. (PRDC)*, 78–87, 2013.

[19] B. Haeupler and F. Kuhn. Lower bounds on information dissemination in dynamic networks. In *Proc. of 26th Int. Symp. on Distributed Computing (DISC)*,166-180, 2012.

[20] D. Ilcinkas, R. Klasing, and A.M. Wade. Exploration of constantly connected dynamic graphs based on cactuses. In *Proc. 21st Int. Coll. Structural Inform. and Comm. Complexity (SIROCCO)*, 250–262, 2014.

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

[22] D. Ilcinkas and A.M. Wade. Exploration of the T-Interval-connected dynamic graphs: the case of the ring. In *Proc. 20th Int. Coll. on Structural Inform. and Comm. Complexity (SIROCCO)*, 13-23, 2013.

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

[24] O. Michail. An Introduction to Temporal Graphs: An Algorithmic Perspective. In *Algorithms, Probability, Networks, and Games*, Springer, 308-343, 2015.

[25] O. Michail, I. Chatzigiannakis, and P.G.. Spirakis. Naming and counting in anonymous unknown dynamic networks In *Proc. 15th Int. Symp. Stab., Safety, Security (SSS)*, Springer, 281-295, 2013.

[26] O. Michail, and P.G.. Spirakis. Traveling Salesman Problems in Temporal Graphs. In *Proc. 39th Int. Symp. on Mathematical Foundations of Computer Science (MFCS)*, Springer, 553–564, 2014.