Journal of Computer and System Sciences 122 (2021) 1–18

Contents lists available at ScienceDirect

Journal of Computer and System Sciences

www.elsevier.com/locate/jcss

Exploration of dynamic networks: Tight bounds on the number of agents[☆]

Tsuyoshi Gotoh[a,*], Paola Flocchini[b], Toshimitsu Masuzawa[a], Nicola Santoro[c]

[a] *Osaka University, Japan*
[b] *University of Ottawa, Canada*
[c] *Carleton University, Canada*

ARTICLE INFO

Article history:
Received 31 May 2020
Received in revised form 4 January 2021
Accepted 14 April 2021
Available online 11 May 2021

Keywords:
Distributed algorithm
Mobile agents
Exploration of dynamic networks
Arbitrary footprint

ABSTRACT

We consider, for the first time, the exploration of dynamic graphs of arbitrary unknown topology. We study the number of agents necessary and sufficient to explore such graphs under the fully synchronous (FSYNC) and the semi-synchronous (SSYNC) activation schedulers. We prove that, under the minimal assumption on the dynamics, *temporal connectivity*, the number of agents sufficient to perform exploration depends on a parameter we call *evanescence* of the graph, and this number is tight. We then consider the stronger well-known assumption of *1-interval connectivity* when the number of edges missing at each time is bounded. We provide tight bounds also in this setting, proving the existence of a difference between FSYNC and SSYNC, as well as between anonymous and non-anonymous agents.

1. Introduction

1.1. Graph exploration

The *graph exploration* problem (EXPLORATION), first introduced by Shannon [1], is a fundamental problem in theoretical computer science, in particular in the field of distributed computing by mobile entities. It requires each node of the graph to be visited by one or more mobile computational entities, called *agents*, a finite number of times (exploration *with termination*) or infinitely often (*perpetual* exploration). In addition to its theoretical importance, EXPLORATION is relevant from a practical viewpoint in networked systems supporting mobile entities (e.g., software agents, vehicles, or robots): by visiting all nodes, agents can check whether there are some nodes with problems in the network, propagate some data across the network, or collect (or search) specific information from the whole network.

This problem has been extensively studied over a variety of assumptions and settings depending on whether the nodes have distinct labellings or are anonymous, on whether the agents have Ids or are anonymous, the type of mechanism available to the agents for interaction or communication (i.e., whiteboards, tokens, face-to-face, vision), on the degree of synchronization (i.e., asynchronous, semi-synchronous, fully-synchronous), on the level of knowledge the agents have about the graph, on their memory, etc. (e.g., see [2–10], and [11] for a recent survey). In spite of all the differences, the existing

[☆] A preliminary extended abstract of this paper appeared in the proceedings of *23rd International Conference on Principles of Distributed Systems (OPODIS 2019)*.

[*] Corresponding author.
E-mail address: t-gotoh@ist.osaka-u.ac.jp (T. Gotoh).

https://doi.org/10.1016/j.jcss.2021.04.003
0022-0000/© 2021 Published by Elsevier Inc.

literature has until very recently made a common assumption: the graph is *static*, i.e., the link structure does not change during the exploration. Static graphs are a common representation of traditional networks, where the changes are typically due to failures; such graphs however fail to describe the new generation of infrastructure-less highly dynamic networks.

1.2. Dynamic networks

In the recent (and now pervasive) generation of *highly dynamic networks*, the topological changes are not sporadic or anomalous; rather they are extensive, continuous, inherent in the nature of the network. These 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. Various models have been proposed to describe some of their aspects, under a variety of names. A unifying model that describes these networks in a simple and natural way is the one of *time-varying graph* (TVG), formally defined in [12], where main classes of systems studied in the literature and their computational relationship were identified.

When time is assumed to be *discrete* (i.e., the system is *synchronous*), the dynamics of the network can be equivalently described as a sequence of static graphs, $\langle G_0, G_1, G_2, \ldots\rangle$, called *evolving graph* or *temporal graph*, where G_i describes the topology of the network at time $t = i$; this representation was originally suggested in [13] and first formalized in [14]. Each G_i is called a *snapshot*, while the aggregate graph $G = \cup_i\{G_i\}$ is called the *footprint* of the temporal graph.

Computations in temporal graphs have been investigated in distributed computing quite extensively. If the dynamics of the changes is arbitrary and unrestricted, clearly any non-trivial computation is unfeasible and any non-trivial problem is unsolvable. Hence, all the studies are carried out under some assumptions restricting the arbitrariness of the dynamics.

The minimal (i.e., less restrictive) assumption is *temporal connectivity*: starting at any time, from any node there exists a temporal path, called journey, to any other node (e.g., [15–17]). Let us stress that, if temporal connectivity does not hold, any non-trivial task and computation is impossible, and any non-trivial problem is unsolvable.

Stronger assumptions include *periodicity*: the network is temporally connected and there is (a known) $p > 1$ such that, for all $i \geq 0$, $G_i = G_{i+p}$ (e.g., [18–22]); *1-interval connectivity*: every G_i is connected (e.g., [23–25]); and *T-interval connectivity*: for every i, the graphs $G_i, G_{i+1}, \ldots, G_{i+(T-1)}$ contain the same spanning-tree (e.g., [22,24]). A classification of the most common assumptions was done in [12].

The studies on computations by mobile agents in temporal graphs are rather recent and quite limited (for a recent survey see [26]). Investigations have focused on gathering [27,28], scattering [29], and graph exploration (discussed in the next section).

1.3. Temporal graph exploration

Many results on Exploration of temporal graphs are *centralized* (or off-line); that is, they assume that the exploring agents have complete a priori knowledge of the topological changes and the times of their occurrence. They include: the study of the complexity of computing a foremost exploration schedule under the 1-interval-connectivity assumption [30], generalized and extended in [31] and then in [32,33]; the computation of an exploration schedule for *rings* under the stronger T-interval-connectivity assumption [22]; the computation of an exploration schedule for *cactuses* under the 1-interval-connectivity assumption [34].

Fewer studies use a *decentralized* (i.e., *distributed*) approach. On the probabilistic side, there is an early seminal work on random walks [35]. On the deterministic side, exploration has been studied under particular constraints on the network connectivity and on its underlying topology. Exploration with termination by a single agent of periodic temporal networks, including *carrier networks*, has been studied in [19–22]. Perpetual exploration by three agents on temporally connected *rings* has been studied in [36,15]. Exploration with termination of 1-interval connected *rings* by two and three agents has been studied in [37], where, in addition to the traditional *fully-synchronous* (Fsync) scheduler (where all the agents are active at every round), they considered also the *semi-synchronous* (Ssync) scheduler where only a subset of the agents is active at each round. Exploration with termination by $O(n)$ agents of $n \times m$ dynamic *tori* ($n \leq m$), where each column and row is a 1-interval connected ring, has been investigated in [38]. Exploration with termination by one agent with partial information about dynamic changes has been studied in [39] for 1-interval connected rings.

Summarizing, all the existing results on distributed exploration of time-varying graphs have been obtained for temporal graphs with very specific topologies (rings, tori, or collections of cycles in the case of carrier networks). In this paper we start the investigation of the exploration of temporal graphs with *arbitrary* and *unknown* topologies.

1.4. Main contributions

In this paper, we consider perpetual exploration by mobile agents of time varying graphs whose topology is arbitrary and unknown to the agents. We focus on solvability of the exploration of such dynamic graphs, and specifically on the number of agents that are necessary and sufficient for exploration under the Fsync and Ssync activation schedulers.

Clearly, if the graph is not *temporally connected*, exploration is trivially impossible to achieve. We thus start our investigation with the class $\mathcal{H}$ of temporally connected temporal graphs. We first prove that the number of agents sufficient to perform exploration is related to the number of its transient edges, a parameter $\eta(\mathcal{G})$ we call *evanescence* of the graph.

Table 1
Tight bounds on the number of agents.

		Anonymous	With leader
Temporally connected: $\mathcal{H}$	FSYNC, SSYNC	$2\eta+1$	$2\eta+1$
ℓ-bounded 1-interval: $\mathcal{W}(\ell)$	SSYNC	$2\ell+1$	2ℓ
	FSYNC	2ℓ	$2\ell-1$

More precisely, we prove that any $\mathcal{G} \in \mathcal{H}$ can be explored by a team of $k \geq 2\eta(\mathcal{G})+1$ anonymous agents. We show that this bound is tight by proving that there are $\mathcal{G} \in \mathcal{H}$ that cannot be explored by $2\eta(\mathcal{G})$ agents.

The impossibility holds under very strong conditions: FSYNC scheduler, agents and nodes with distinct IDs, knowledge on n and k, unbounded-size whiteboards. On the other hand, the proposed exploration algorithm, based on the rotor router technique, works under very weak conditions: SSYNC scheduler (under the weakest transport condition), anonymous agents, no knowledge of topological parameters, and $O(\log \delta_v)$ bits whiteboard at node v (where δ_v denotes the degree of v in the footprint of the temporal graph).

We then turn our attention to the stronger assumption on the dynamics of the graph, *1-interval connectivity*: the graph is always connected. Let $\mathcal{W}(\ell) \subset \mathcal{H}$ be the class of these always-connected temporal graphs where the number of missing edges at each time is at most ℓ.

We start by considering the case of *anonymous* agents. We first prove a tight bound of $2\ell+1$ agents under the SSYNC scheduler. The proposed algorithm performs exploration even if the network size and the number of agents are not known, and with the weakest transport condition; the impossibility with fewer agents holds even if the network size and the number of agents are known and with whiteboards of unbounded-size.

We then prove the existence of a difference between FSYNC and SSYNC when the network size and the number of agents are known. In fact, in this case, we show a tight bound of 2ℓ for FSYNC. Moreover, we show that with $2\ell+1$ agents *exploration with termination* is possible in FSYNC.

Finally, we consider the case of non-anonymous agents, assuming the presence of a *leader* agent. While the lower bound on the number of agents needed for the exploration of $\mathcal{H}$ holds regardless of the existence of a leader, we prove that non-anonymity has an impact on the exploration of $\mathcal{W}(\ell)$. In fact, by exploiting the presence of a leader, the bound on the number of agents decreases by one both in FSYNC and in SSYNC. Moreover, we show that, with a leader, 2ℓ agents can *explore with termination* in FSYNC.

These results are summarized in Table 1.

Our results indicate, among other things, that the much weaker condition of semi-synchrony (with respect to full-synchrony) is enough to undermine the advantages provided by the much stronger connectivity assumption of $\mathcal{W}$ (with respect to $\mathcal{H}$). Indeed, when considering the class $\mathcal{H}(\ell)$ of temporally connected graphs with at most ℓ transient edges and the class $\mathcal{W}(\ell) \subset \mathcal{H}(\ell)$ of ℓ-bounded 1-interval connected graph, we have that the bound on the number of agents for $\mathcal{H}(\ell)$ is the same as the one for $\mathcal{W}(\ell)$ for SSYNC, while the two differ in the case of FSYNC.

2. The model

2.1. Temporal graph

The highly dynamic network is modeled as a time-varying graph (TVG), $\mathcal{G} = (V, E, \mathbb{T}, \rho)$, where V is a set of nodes, E is a set of edges, $\mathbb{T}$ is the temporal domain, and $\rho : E \times \mathbb{T} \to \{0, 1\}$, called *presence function*, indicates whether a given edge is available at a given time. The graph $G = (V, E)$ is called *underlying* graph (or *footprint*) of $\mathcal{G}$, with $|V| = n$ and $|E| = m$. Let $E(v)$ denote the set of edges incident on node v in the footprint, let $\delta_v = |E(v)|$ be the degree of node v in the footprint, and let $\Delta = Max_v\{\delta_v\}$ be the maximum degree of G. The nodes in V are anonymous (i.e., they have no IDs). Each edge incident to node v is locally labeled (i.e., has a port-number); the labeling function is a bijection $\lambda_v : E(v) \to \{0, \ldots, \delta_v - 1\}$ that associate a different label to each edge incident to v; no other assumption is made about the labels.

A *journey* is a temporal walk in $\mathcal{G}$ and it is defined as a sequence of couples $\mathcal{J} = \{(e_1, t_1), (e_2, t_2) \ldots, (e_k, t_k)\}$, such that $\{e_1, e_2, ..., e_k\}$ is a walk in G and $\forall i, 1 \leq i < k$, $\rho(e_i, t_i) = 1$ and $t_{i+1} > t_i$. Let $J(u, v, t)$ denote the set of journeys from u to v starting at time $t' \geq t$.

In this paper we assume *discrete* time; that is, $\mathbb{T} = \mathbb{Z}^+$. In this case, the TVG $\mathcal{G}$ is usually called *temporal graph* (or *evolving graph*), and can be viewed as a sequence of static graphs: $\mathcal{S}_\mathcal{G} = G_0, G_1, \ldots, G_t, \ldots$, where $G_t = (V, E_t)$ is the graph induced by the edges present at time t (called *snapshot* of $\mathcal{G}$ at time t). We denote by $\bar{E}_t = E \setminus E_t$ $(\subseteq E)$ the set of edges that do not appear in the snapshot at time t.

An edge $e \in E$ is said to be *recurrent* if $\forall t \in \mathbb{Z}^+, \exists t' > t : \rho(e^*, t') = 1$; in other words, a recurrent edge appears infinitely often. An edge $e \in E$ that is not recurrent is said to be *transient*; in other words, a transient edge appears only in a finite number of snapshots. Let E^* and E^- denote the set of recurrent and of transient edges, respectively; the number $\sigma(\mathcal{G}) = |E^*|$ of recurrent edges is called the *solidity* of $\mathcal{G}$; while the number $\eta(\mathcal{G}) = |E^-| = |E| - \sigma(\mathcal{G}))$ of transient edges is called the *evanescence* of $\mathcal{G}$. Let $G_r = (V, E^*)$, i.e., G_r is a subgraph of G induced by recurrent edges.

2.2. Connectivity

Temporal graphs can be classified in terms of the effect that the dynamic topological changes have on their connectivity.

Definition 1 *(Temporally connected).* A temporal graph $\mathcal{G}$ is *temporally connected* (or *connected over time*) if $\forall t \in \mathbb{Z}^+$, $\forall u, v \in V$, $J(u, v, t) \neq \emptyset$.

Note that temporal connectivity is the minimal condition to be able to perform any global task regardless of the initial position of the agents; in particular, any problem requiring every node to be involved (e.g., exploration) is trivially unsolvable if $\mathcal{G}$ is not temporally connected. Let $\mathcal{H}$ denote the class of temporally connected TVGs.

A variety of stronger assumptions have been studied in the literature. In this paper we are interested also in the well-known class of temporal graphs where connectivity is actually guaranteed at every time, and in particular when the number of missing edges at any given time is bounded.

Definition 2 *(ℓ-Bounded 1-interval connected).* A temporal graph $\mathcal{G}$ is *1-interval connected* (or *always connected*) if $\forall G_i \in \mathcal{S}_{\mathcal{G}}$, G_i is connected. Moreover, $\mathcal{G}$ is *ℓ-bounded 1-interval connected* if it is always connected and $|\bar{E}_t| \leq \ell$.

Let $\mathcal{W}(\ell) \subset \mathcal{H}$ denote the class of ℓ-bounded 1-interval connected temporal graphs.

2.3. Agents

A set $A = \{a_0, a_1, \ldots, a_{k-1}\}$ of k *agents* operates in $\mathcal{G}$, initially occupying arbitrary positions. When the agents are all undistinguishable, we say that they are *anonymous*; if one of them is different from all the others, we say that they have a *leader* (and are not-anonymous). Each agent $a \in A$ is a computational entity endowed with private memory (called notebook), and capable of moving from a node to a neighboring node (provided that edge exists at the time).

When at a node v, an agent has access to the node's ports and rotor-router mechanism. More precisely, in correspondence of each edge $e \in E(v)$, there is in v a *port* p_i where $i = \lambda_v(e)$, used by agents (at most one at a time) intending to leave v through e. Additionally, v provides a rotor-router mechanism, which indicates one of the ports; this indication can be read and modified by the agents; access to this mechanism is in fair mutual exclusion. Note that this mechanism can be implemented at a node using more traditional tools for inter-agent communication (e.g., token, whiteboard) offering access in fair mutual exclusion; for example, by a single pebble that can be placed in correspondence of one of the ports, and moved to another port when necessary; alternatively, by a whiteboard of $O(\log \delta_v)$ bits. In the following, for simplicity, our presentation and discussions will be in terms of a whiteboard implementation. More precisely, we assume each node v has some local storage space, called *whiteboard*, of size $O(\log \delta_v)$ bits that can be accessed by the agents located at node v. Access to the whiteboard is assumed to be done in mutual exclusion.

The agents operate in synchronous rounds, and each round is composed of three phases: Look, Compute, and Move, during which they execute the following actions [40]:

Look: Agent a_i observes the content of its notebook and of the whiteboard of the node where it currently resides; it checks the node and its ports to determine if there are other agents at this node and where (e.g., which ports).
Compute: On the basis of the information obtained in the Look phase, a_i decides whether to move or not, and it can write information on the whiteboard. If it decides to move, it places itself in correspondence of the selected port (if it is not occupied by another agent).
Move: If a_i is at a port, it tries to move; if the corresponding edge exists, a_i reaches the other side, otherwise it stays on the port. If a_i does not occupy a port, it does not move.

Each Look and Compute phase is executed as an atomic action. Atomic actions of agents in the same node are executed with mutual exclusion access to the whiteboard. After Look and Compute of all the agents finish, they simultaneously start the Move phase. For example, if there are two agents a and b on a node v, a first executes its Look and Compute, then, b executes its Look and Compute, and finally a and b (and all the other agents) execute Move.

We distinguish between the *fully-synchronous* activation scheduler (Fsync), when all the agents are activated in every round, and the *semi-synchronous* one (Ssync), when an arbitrary subset of the agents is activated at each round. In Ssync, the scheduler is an adversary which knows the algorithm of the agents, has infinite computing capacity, and tries to prevent agents from completing their task; however, it must activate every agent infinitely often. An agent which is not activated at round t is said to be *sleeping* at that round. The length of the sleeping time is finite but unbounded.

Under the semi-synchronous scheduler, it is necessary to specify the behavior of the agents that fall asleep on a port when the corresponding edge is missing. We consider the weakest condition, *eventual transport*, according to which the agent sleeping at a port will eventually be activated at a time when the edge corresponding to the port is present [37]; this prevents the adversary from using semi-synchronicity to block an agent forever on a recurrent edge.

2.4. Configuration and execution

A *configuration* C_t is defined by: the contents of the whiteboards, the local memory of the agents, and the locations of the agents at the start of round t.

An *execution* $\mathcal{E}(\mathcal{A}) = C_0C_1\ldots$ of an algorithm $\mathcal{A}$ is an infinite sequence of configurations such that C_0 is an initial configuration (i.e., a configuration at round 0) and C_{t+1} is obtained from C_t by executing one round of algorithm $\mathcal{A}$. This execution is subject to two types of adversarial actions: those by the activation scheduler deciding which agents are activated in that round, and those of the topological scheduler deciding which edges are missing in that round. When no ambiguity arises, we use $\mathcal{E}$ instead of $\mathcal{E}(\mathcal{A})$.

2.5. Augmented configuration and execution

We use an *augmented configuration* and an *augmented execution* in Sections 4.2 and 5.2. To define an augmented configuration, we introduce variable $\mathsf{visited}_v$ for all $v \in V$ which is written and read only by an external observer. The initial value of $\mathsf{visited}_v$ is 0. When v is visited, $\mathsf{visited}_v$ is set to 1 by the external observer.

Then, an augmented configuration C_t^{aug} is defined by: configuration C_t and the value of $\mathsf{visited}_v$ of every node v at round t. We say that an augmented configuration is *terminal* when $\mathsf{visited}_v = 1$ for any node v.

An augmented execution $\mathcal{E}^{\mathrm{aug}}(\mathcal{A}) = C_0^{\mathrm{aug}}C_1^{\mathrm{aug}}\ldots C_r^{\mathrm{aug}}$ is a sequence of augmented configurations such that C_0^{aug} is an initial augmented configuration; C_{t+1}^{aug} is obtained from C_t^{aug} by executing one round of algorithm $\mathcal{A}$; C_r^{aug} is a unique terminal configuration in $\mathcal{E}^{\mathrm{aug}}$. An augmented execution is also subject to the two types of adversarial actions. Note that the agents may keep executing $\mathcal{A}$ after round r, but augmented configurations after round r are ignored in $\mathcal{E}^{\mathrm{aug}}$. When no ambiguity arises, we use $\mathcal{E}^{\mathrm{aug}}$ instead of $\mathcal{E}^{\mathrm{aug}}(\mathcal{A})$ and an "execution" instead of an "augmented execution".

2.6. Exploration

We say that a node v is *visited at* round t if v contains an agent at the beginning of round t. We say that a node v is *explored by* round t if v is visited at round t' for some t' $(0 \le t' \le t)$. We say that the network is explored by round t if every node is explored by round t.

A *perpetual* exploration algorithm is one where, in every execution, every node is visited at an infinite number of rounds. An exploration algorithm *with termination* is one where, in every execution, all the agents terminate after all nodes have been visited at least once. In this paper, we are mainly concerned with *perpetual exploration*.

3. Exploration of temporally connected TVGs

In this section, we consider the minimal class of explorable temporal graphs: *temporally connected TVGs*, and we show that the feasibility of the exploration of $\mathcal{G}$ is related to its evanescence η, providing a tight bound of $2\eta(\mathcal{G}) + 1$ agents.

3.1. Impossibility

Let $\mathcal{H}(\ell) = \{\mathcal{G} \in \mathcal{H} : \eta(\mathcal{G}) \le \ell\}$ be the class of temporally connected TVGs with evanescence at most ℓ. In this section we show that it is impossible to perform perpetual exploration of all $\mathcal{G} \in \mathcal{H}(\ell)$ with 2ℓ agents. The result is quite strong as it applies also to TVGs that are connected at every time step, with uniquely labeled nodes and agents, under a fully-synchronous scheduler, and in presence of topological knowledge.

Theorem 3. *There exist temporally connected time-varying graphs $\mathcal{G} \in \mathcal{H}(\ell)$ that cannot be explored by $k = 2\ell$ agents. The result holds even if nodes and/or agents have distinct IDs, the network is always connected, the agents know n, m and k, and the scheduler is fully-synchronous.*

Proof. We show the theorem by constructing a graph $\mathcal{G} \in \mathcal{H}(\ell)$ that cannot be explored by 2ℓ agents by any algorithm. The main point of this proof is that an agent can eventually have only one of these two behaviors when wishing to traverse an edge that is missing: (i) the agent stays permanently on the chosen port, waiting for the appearance of the continuously missing edge; (ii) the agent eventually chooses a different edge. The agents of the former type are called (with respect to the number of changes of a selected edge) *finite agents* and those of the latter are *infinite agents*.

The components for constructing the graph are as follows. For $0 \le i \le 2\ell - 1$ $(= k - 1)$, let S_i be a star with center node c_i and 3 leaf nodes $\{b_{(i,0)}, b_{(i,1)}, b_{(i,2)}\}$. We construct the graph using S_i for $0 \le i \le 2\ell - 1$ and an additional node u.

Each component is connected as follows. For $0 \le i \le 2\ell - 1$ and $j \in \{0, 1\}$, each $b_{(i,j)}$ is connected with u by edge $(b_{(i,j)}, u)$; and for $0 \le i \le \ell - 1$, each $b_{(2i,2)}$ connected with $b_{(2i+1,2)}$ by $(b_{(2i,2)}, b_{(2i+1,2)})$. A graph for $\ell = 2$ $(k = 4)$ is depicted in Fig. 1.

For the constructed graph, we first show that, given any exploration algorithm using 2ℓ agents, the adversary can construct an execution for the algorithm such that in the execution $\mathcal{G}$ cannot be explored while the adversary may violate the

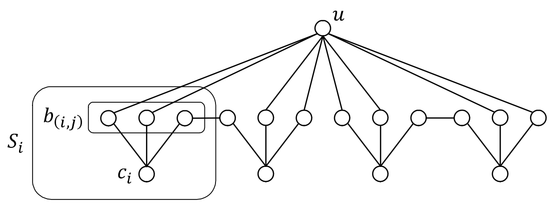

Fig. 1. Example of a graph for $\ell = 2$ and $k = 2\ell = 4$ that cannot be explored by 2ℓ agents. There are four stars S_i for $0 \le i \le 3$ in the figure. Each star S_i has one center node c_i and three leaf nodes $\{b_{(i,0)}, b_{(i,1)}, b_{(i,2)}\}$.

restriction of $\mathcal{H}(\ell)$, i.e., $\eta(\mathcal{G})$ may be more than ℓ. Then, we give a way to convert the execution into another execution such that $\eta(\mathcal{G})$ is at most ℓ in the new execution and the agents cannot distinguish these two executions and thus cannot explore $\mathcal{G}$ also in the new execution.

We start by showing that, given any exploration algorithm, say $\mathcal{A}$, using 2ℓ agents, the adversary can construct an execution $\mathcal{E}_1$ of $\mathcal{A}$ in which the agents cannot explore $\mathcal{G}$. The adversary puts agent a_i on c_i for $0 \le i \le 2\ell - 1$ in the initial configuration of $\mathcal{E}_1$. During execution $\mathcal{E}_1$ of $\mathcal{A}$, the adversary deletes the edge leading to u or the other star whenever a_i is on $b_{(i,j)}$. Clearly, this prevents any agent executing $\mathcal{A}$ from visiting u and thus $\mathcal{G}$ is not explored permanently while the adversary violates the restriction for the number of transient edges (it is at most 2ℓ in $\mathcal{E}_1$).

We now show how the adversary converts $\mathcal{E}_1$ into another execution, say $\mathcal{E}_2$, so that the agents cannot distinguish $\mathcal{E}_1$ and $\mathcal{E}_2$ and $\eta(\mathcal{G})$ is at most ℓ in $\mathcal{E}_2$. The adversary first separates the agents into two groups: *finite agents* and *infinite agents* depending on their behavior when faced with a missing edge during $\mathcal{E}_1$. Let f $(0 \le f \le k)$ be the number of *finite agents*. In the following, *finite agents* are denoted by $a_0^{\mathsf{fin}}, \dots, a_{f-1}^{\mathsf{fin}}$. In the initial configuration of $\mathcal{E}_2$, each agent (a_i) is put on the same node (c_i) as in $\mathcal{E}_1$.

Then, the adversary constructs a new assignment of the port labels and the node ID (if any) of nodes so that every agent cannot distinguish $\mathcal{E}_1$ and $\mathcal{E}_2$ as follows. For *infinite agents*, the adversary does nothing. For *finite agents*, let $a_i^{\mathsf{fin}} = a_{i'}$ and $b_{(i',x_i)}$ be the node where a_i^{fin} finally waits for a missing edge permanently in $\mathcal{E}_1$. For $0 \le i \le f - 1$, the adversary does the following: if $x_i = 2$, the adversary does nothing; and otherwise, the adversary swaps the assignment of the port labels and the node ID of $b_{(i',2)}$ and $b_{(i',x_i)}$ and accordingly permutes the port labeling of $c_{i'}$.

Execution $\mathcal{E}_2$ with the initial configuration, the node ID, and the assignment of port labels is constructed similarly to $\mathcal{E}_1$: the adversary deletes the edge leading to u or the other star when a_i exists on $b_{(i,j)}$. Obviously, every agent cannot distinguish $\mathcal{E}_1$ and $\mathcal{E}_2$: for all the agents, the node IDs and the port labeling observed in $\mathcal{E}_2$ is the same as $\mathcal{E}_1$. Thus, $\mathcal{G}$ cannot be explored since u is not visited by any agent also in $\mathcal{E}_2$.

Since the edges waited permanently by an agent are only $(b_{(2i,2)}, b_{(2i+1,2)})$ for $0 \le i \le \ell - 1$, $\eta(\mathcal{G})$ is at most ℓ in $\mathcal{E}_2$. □

3.2. *Semi synchronous exploration by* $2\eta(\mathcal{G}) + 1$ *agents*

In this section, we show that every temporally connected time-varying graph $\mathcal{G} \in \mathcal{H}$ can be explored by $2\eta(\mathcal{G}) + 1$ anonymous agents that do not know the topology. In fact, we propose an exploration algorithm for $2\eta(\mathcal{G}) + 1$ anonymous agents in an anonymous network, which works under the semi-synchronous scheduler with eventual transport.

The strategy is simple and it is based on the classical *rotor router* mechanism, which was introduced as a deterministic alternative to random walk and was studied in a variety of contexts, including static graph exploration (e.g., [41–45]).

In rotor router, each node v has a variable written on its whiteboard, $\mathsf{pointer}_v$, indicating one of its incident ports. When an agent a visits node v, a checks each port in ascending order from the port pointed by $\mathsf{pointer}_v$. If a finds some unoccupied port p, a moves to that port and sets $\mathsf{pointer}_v$ to $p + 1$. If a finishes to check all the ports and they all are occupied, a does nothing.

Algorithm 1 Computation at node v.

```
1: if not on a port then
2:   $i \leftarrow 0$
3:   $p \leftarrow \mathsf{pointer}_v$
4:   while $i < \delta_v \wedge$ port $p$ is occupied do
5:     $p \leftarrow (p + 1) \bmod \delta_v$
6:     $i \leftarrow i + 1$
7:   if $i < \delta_v$ then
8:     $\mathsf{pointer}_v \leftarrow (p + 1) \bmod \delta_v$
9:     move to port $p$
```

We first show that, in any round, there exists at least one agent succeeding to move within finite time (Lemma 4). We then show that, $2\ell + 1$ agents achieve perpetual exploration using Algorithm 1 (Theorem 5).

Lemma 4. *For any round t, if $2\eta(\mathcal{G})+1$ agents execute Algorithm 1 in a temporally connected temporal graph $\mathcal{G}$, at least one of them eventually moves after t.*

Proof. By contradiction, assume that there exists a round t such that every agent never succeeds to move after t. We consider two cases: (i) there exists a node v containing more than $\delta_v - 1$ agents, and (ii) there does not exist such a node.

In the first case, every agent on v is activated within finite time after t because of the fairness of the scheduler, which means that every port of v is eventually occupied by an agent. Since at least one of the edges incident to v is a recurrent edge, say e, the agent sleeping on the corresponding port of e eventually succeeds to move because of the eventual transport rule. This is a contradiction.

Also in the second case, every agent on v is activated within finite time after round t because of the fairness of the scheduler. Since there is no node containing more agents than its degree, every agent eventually stays on a port. When this happens, at least one of the agents is sleeping at the port of a recurrent edge since the number of agents is $2\eta(\mathcal{G})+1$ and there exist at most $2\eta(\mathcal{G})$ ports corresponding to transient edges. This means that, by the eventual transport rule, the agent sleeping at the port of a recurrent edge eventually succeeds to move after t; a contradiction. □

Then, the following theorem holds.

Theorem 5. *Any $\mathcal{G} \in \mathcal{H}$ can be explored by $2\eta(\mathcal{G})+1$ anonymous agents under the semi-synchronous scheduler.*

Proof. Consider Algorithm 1. By the definition of transient edges, there exists a time step t_e for any transient edge e such that $\rho(e,t)=0$ for all $t > t_e$. Let t_E be $\max_{e\in E^-} t_e$, i.e., a time when all the transient edges have ceased to exist and all the edges that appear from this moment are recurrent. In the following, we consider the execution after time t_E. Let $x(t)$ be the sum of the number of agent moves from a node to another node over all the agents from the beginning of the execution up to time t.

We now show that, from an arbitrary initial configuration, $2\eta(\mathcal{G})+1$ agents following Algorithm 1 visit all the nodes infinitely often.

First, note that there exists a node, say v, that is visited infinitely often (for $t \to \infty$) because $x(t)$ goes to infinity (for $t \to \infty$) by Lemma 4.

We now show that every neighbor of v connected by a recurrent edge is also visited at an infinite number of rounds. We prove it by contradiction. Suppose that a neighbor u of v connected by a recurrent edge is visited at only a finite number of times and let t' be the last round when u is visited at. Since v is visited at an infinite number of rounds and the agents execute Algorithm 1 perpetually, some agent a visiting v eventually chooses (v,u) as the edge from which a moves out of v after time t'. Recall that (v,u) is a recurrent edge and the agents are activated by the eventual transport rule. It follows that a eventually visits u after round t'; a contradiction.

Since G_r is temporally connected, we can apply inductively the claim (e.g., the neighbors of a neighbor of v are also visited infinitely often) to all the nodes, proving the theorem. □

From Theorems 3 and 5, the following Theorem holds.

Theorem 6. *Exploration of all temporal graphs in $\mathcal{H}(\ell)$ by k agents is possible iff*

$$k \geq 2\ell + 1$$

Note that, if a graph is temporally connected, then its solidity $\sigma(\mathcal{G}) \geq n-1$; as a consequence, we have:

Theorem 7. *Every temporally connected temporal graph with n nodes and whose footprint has m edges can be explored by $2(m-n)+3$ agents.*

4. Exploration of 1-interval connected TVGs by anonymous agents

In this Section, we turn our attention to the class $\mathcal{W}(\ell)$ of 1-interval connected temporal graphs where the number of missing edges is bounded in each round by a constant ℓ. In other words, at any time t the TVG is connected, and no more than ℓ edges are missing. We establish tight bounds for the exploration of this class of temporal graphs by *anonymous* agents, in SSYNC and in FSYNC.

4.1. Semi-synchronous model

We first consider ℓ-bounded, 1-interval connected TVGs operating under a semi-synchronous scheduler and we show that there exist TVGs that cannot be explored by 2ℓ anonymous agents.

Theorem 8. *There exist 1-interval connected time-varying graphs $\mathcal{G} \in \mathcal{W}(\ell)$ that cannot be explored by $k = 2\ell$ anonymous agents. The result holds even if the agents know n, m and k and whiteboards are of unbounded size.*

Proof. We use the same graph $\mathcal{G}$ constructed for the proof of Theorem 3. The construction is omitted in this proof.

We first show that, given any exploration algorithm, say $\mathcal{A}$, using 2ℓ agents, the adversary can construct an execution $\mathcal{E}_1$ of $\mathcal{A}$, possibly violating the eventual transport rule, in which the agents cannot explore $\mathcal{G}$. We then show that it is always possible to convert this execution into another execution $\mathcal{E}_2$ that does not violate the eventual transport rule, and where the agents cannot explore $\mathcal{G}$.

In execution $\mathcal{E}_1$, the adversary puts agent a_i on c_i for $0 \le i \le k-1 = 2\ell - 1$ in the initial configuration of $\mathcal{E}_1$. During $\mathcal{E}_1$, exactly one agent is activated at each round: a_i is activated at round t when $t \equiv i \pmod{k}$. When the adversary activates a_i and a_i exists on $b_{(i,j)}$, the adversary deletes the edge leading to u or the other star whereas all the other edges are present. Note that the agents and the nodes are anonymous and thus either they are all *finite* (i.e., every agent permanently waits for appearance of its selected edge if the edge is permanently missing) or they are all *infinite* (i.e., every agent eventually changes its selected edge if the edge remains missing) in $\mathcal{E}_1$.

If the agents are *infinite*, the eventual transport rule is not violated even in $\mathcal{E}_1$ and thus the adversary can prevent the agents from completing the exploration in $\mathcal{E}_1$.

If the agents are *finite*, the adversary converts $\mathcal{E}_1$ into another execution, say $\mathcal{E}_2$, as follows. The adversary first puts a_i $(0 \le i \le k-1)$ on c_i in the initial configuration of $\mathcal{E}_2$. Then, the adversary changes the assignment of the port labels and the node ID (if any) of each node in S_i in the same way explained in the proof of Theorem 3 (also omitted in this proof). In $\mathcal{E}_2$, the adversary activates each agent in the same order as in $\mathcal{E}_1$ and deletes an edge leading to u or the other star whenever a_i is on $b_{(i,j)}$. After some round t from which every agent a_i does not change its selected edge, i.e., $b_{(i,2)}$, and waits at a port of $b_{(i,2)}$ forever for $0 \le i \le 2l$, the adversary deletes $(b_{(2j,2)}, b_{(2j+1,2)})$ for $0 \le j \le \ell - 1$ at every round. Obviously, every agent cannot distinguish $\mathcal{E}_2$ from $\mathcal{E}_1$ and $\mathcal{G}$ cannot be explored since u is not visited by any agent in $\mathcal{E}_2$. It is also clear that the eventual transport rule is not violated in $\mathcal{E}_2$. □

Clearly, $\mathcal{W}(\ell) \subset H(\ell)$, thus any $\mathcal{G} \in \mathcal{W}(\ell)$ can be explored by Algorithm 1; that is:

Theorem 9. *Any $\mathcal{G} \in \mathcal{W}(\ell)$ can be explored by $2\ell + 1$ anonymous agents under the semi-synchronous scheduler with eventual transport.*

From Theorems 8 and 9 it follows that:

Theorem 10. *Under a semi-synchronous scheduler, exploration of all ℓ-bounded 1-interval connected TVG by k anonymous agents is possible iff $k \ge 2\ell + 1$.*

4.2. Fully-synchronous model

In this section, we show that, if the network size and the number of agents are known, there exists a difference between FSYNC and SSYNC in the exploration of ℓ-bounded 1-interval TVGs. In fact, we show that, $\mathcal{G} \in \mathcal{W}(\ell)$ can be explored if $k \ge 2\ell$, while there exist graphs that cannot be explored with $2\ell - 1$ agents.

4.2.1. Impossibility

We now consider ℓ-bounded, 1-interval connected TVGs operating under a fully-synchronous scheduler and we show that there exist TVGs that cannot be explored by $2\ell - 1$ agents, even if the agents know n, m, and k.

Theorem 11. *There exist ℓ-bounded 1-interval time-varying graphs $\mathcal{G} \in \mathcal{W}(\ell)$ that cannot be explored by $k = 2\ell - 1$ anonymous agents in FSYNC. The result holds even if the agents know n, m, and k, and whiteboards are of unbounded size.*

Proof. Let $K_{2\ell} = (V_{2\ell}, E_{2\ell})$ be the complete graph with 2ℓ nodes where $V_{2\ell} = \{v_0, v_1, \ldots, v_{2\ell-1}\}$. It is well known that the edges of $K_{2\ell}$ can be colored with $2\ell - 1$ colors, that is, $E_{2\ell}$ can be partitioned into $2\ell - 1$ disjoint independent edge sets (or complete matchings): $E_{2\ell}^{(0)}, E_{2\ell}^{(1)}, \ldots, E_{2\ell}^{(2\ell-2)}$. For example, the following separation leads to disjoint independent edge sets: each $E_{2\ell}^{(i)}$ has ℓ edges, $(v_i, v_{2\ell-1})$, (v_{i-1}, v_{i+1}), (v_{i-2}, v_{i+2}), $\ldots$, $(v_{i-\ell+1}, v_{i+\ell-1})$, see Fig. 2 (for simplicity, mod 2ℓ is omitted).

The execution where $v_{2\ell-1}$ remains unvisited is constructed as follows. For $0 \le i \le 2\ell - 1$, the adversary places each agent a_i on v_i and for $0 \le j \le 2\ell - 2$ assigns a label j to the port of v_i corresponding to e, if $e \in E_{2\ell}^{(j)}$. Note that, since agents and nodes are anonymous, all the agents select the port with the same label to move at each round. Thus, the adversary can prevent any agent from moving by deleting all the edges of $E_{2\ell}^{(i)}$ when the agent selects port i; as a consequence, none of the agents can move out of their current nodes. This means that $v_{2\ell-1}$ remains unvisited forever.

In this execution, the number of missing edges is always ℓ and the network is obviously kept connected. Thus, the theorem holds. □

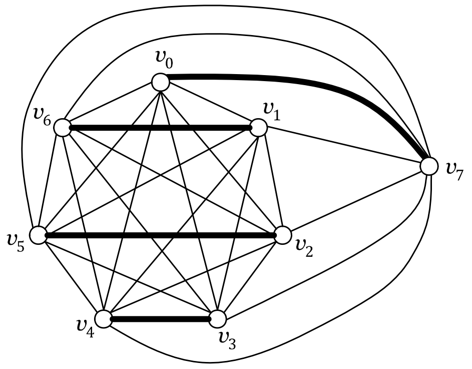

Fig. 2. Example of a graph for $\ell = 4$ and $k = 2\ell - 1 = 7$ that cannot be explored by $2\ell - 1$ agents and its coloring. The bold lines are the edges of $E_8^{(0)}$.

4.2.2. Bound on exploration time

Let $\mathcal{G} \in \mathcal{W}(\ell)$. Since $\mathcal{W}(\ell) \subset H(\ell)$, $2\ell + 1$ agents can clearly completes the exploration by Algorithm 1 in graph $\mathcal{G}$. Interestingly, when executed on $\mathcal{G} \in \mathcal{W}(\ell)$, it can be shown that the time complexity of exploration can be bounded under the fully-synchronous scheduler. More specifically, we show that within $\Delta^n(\Delta+1)^k(n-1)^k$ rounds, all nodes of the graph have been visited at least once by a team of $k = 2\ell + 1$ agents.

We prove the theorem by a sequence of lemmas. First of all, we can easily show that $2\ell+1$ agents executing Algorithm 1 cannot be all prevented from moving at any given round.

Lemma 12. *If $2\ell + 1$ agents activated fully-synchronously execute Algorithm 1 in ℓ-bounded 1-interval TVGs, at least one of them succeeds to move at every round.*

Proof. There exist two cases as in the proof of Lemma 4: at round t, (i) there exists a node v containing more than $\delta_v - 1$ agents, and (ii) there does not exist such a node.

In the first case, since there are more than $\delta_v - 1$ agents at v, every port is occupied by one agent at t since every agent is activated. In addition to that, v has at least one adjacent edge present at t by the connectivity of the TVG. This implies that at least one agent succeeds to move at round t.

In the second case, each agent occupies one port by assumption and by fully-synchronous activation, which means that $2\ell + 1$ ports are occupied. Moreover, at most ℓ edges are missing at each round, which means that at most 2ℓ ports are blocked at each round. It follows that at least one agent can move at round t also in this case. $\square$

For $\mathcal{E}^{\mathrm{aug}}$ of Algorithm 1, the following lemma holds.

Lemma 13. *In an augmented execution of Algorithm 1 by $2\ell + 1$ agents, any two augmented configurations are different.*

Proof. First note that Lemma 12 precludes the same two consecutive augmented configurations C_t^{aug} and C_{t+1}^{aug} in an augmented execution $\mathcal{E}^{\mathrm{aug}}$ of Algorithm 1 where no agents move between C_t^{aug} and C_{t+1}^{aug}. Suppose that there exist two augmented configurations C_t^{aug} and $C_{t'}^{\mathrm{aug}}$ for $t < t'$ in $\mathcal{E}^{\mathrm{aug}}$. Let $\mathcal{E}_{t,t'}^{\mathrm{aug}} = C_t^{\mathrm{aug}} C_{t+1}^{\mathrm{aug}} \cdots C_{t'-1}^{\mathrm{aug}}$ be a subsequence of $\mathcal{E}^{\mathrm{aug}}$. In this case, the adversary can create an infinite augmented execution from $\mathcal{E}^{\mathrm{aug}}$ by repeating $\mathcal{E}_{t,t'}^{\mathrm{aug}}$, which means that the adversary can create an (augmented) execution where $2\ell + 1$ agents cannot complete the exploration forever. This contradicts Theorem 5. Thus, the lemma holds. $\square$

We are now ready to show an upper bound on the exploration time of Algorithm 1, which is obtained by calculating the maximum length among all the augmented executions.

Lemma 14. *The length of any possible augmented execution by $k = 2\ell + 1$ agents is bounded by $\Delta^n(\Delta+1)^k(n-1)^k$.*

Proof. Let α be the maximum length among all the possible augmented executions. By Lemma 13, α is bounded by the number of possible augmented configurations in an execution.

The number of possible configurations on a fixed node set $V' \subseteq V$ is bounded by $\Delta^{|V'|}(|V'|(\Delta+1))^k$, which corresponds to all the combinations of the possible values of $\mathsf{pointer}_v$ (i.e., $\Delta^{|V'|}$) and all of the agents' locations (i.e., $(|V'|(\Delta+1))^k$). Notice that only $\mathsf{pointer}_v$ of each node v is used as a variable in Algorithm 1. Since the number of nodes visited by an agent is not decreasing during the exploration, the exploration time is smaller than or equal to the sum of $\Delta^{|V'|}(|V'|(\Delta+1))^k$ for $1 \le |V'| \le n-1$, i.e., $\alpha \le \sum_{|V'|=1}^{n-1} \Delta^{|V'|}(|V'|(\Delta+1))^k \le \Delta^n(\Delta+1)^k(n-1)^k$ rounds. $\square$

It then follows that:

Theorem 15. *In* FSYNC, *Algorithm 1 executed by $k = 2\ell + 1$ anonymous agents explores any ℓ-bounded 1-interval connected TVG within $\Delta^n(\Delta + 1)^k(n - 1)^k$ rounds.*

Note that, as a consequence, we obtain a *terminating exploration* algorithm for ℓ-bounded 1-interval connected TVGs.

Theorem 16. *In* FSYNC, *with knowledge of n and k,* exploration with termination *of an arbitrary ℓ-bounded 1-interval connected temporal graph $\mathcal{W}(\ell)$ can be achieved in n^{n+2k} rounds by $k = 2\ell + 1$ agents.*

4.2.3. Exploration by 2ℓ agents

The result of the previous section can be used to obtain a perpetual exploration algorithm of ℓ-bounded 1-interval connected graphs by 2ℓ agents (which know n and k). The solution (Algorithm 2 below) is obtained by applying to Algorithm 1 bounding the waiting time of an agent blocked on a missing edge.

In fact, while an agent keeps waiting for a missing edge forever in Algorithm 1, in Algorithm 2 an agent waits for a missing edge up to kT rounds where T is calculated on the basis of the results of Section 4.2.2.

Apart from the waiting time, the rest of the algorithm is the same as in Algorithm 1: each node has $\mathsf{pointer}_v$ pointing at a port. When agent a visits v, a checks each port in ascending order from the port pointed by $\mathsf{pointer}_v$. If a finds some unoccupied port p, a moves to the port and sets $\mathsf{pointer}_v$ to $p + 1$. If a finishes to check all the ports and they all are occupied, a does nothing.

Variable Waiting of an agent represents the elapsed time since the last round when the agent moved to the current port.

Algorithm 2 Computation at node v.

```
1: if on a port then
2:     Waiting ← Waiting + 1
3:     if Waiting > $kT$ then
4:         exit the current port
5: if not on a port then
6:     Waiting ← 0
7:     $i$ ← 0
8:     $p$ ← $\mathsf{pointer}_v$
9:     while $i < \delta_v \wedge$ port $p$ is occupied do
10:        $p \leftarrow (p + 1) \bmod \delta_v$
11:        $i \leftarrow i + 1$
12:    if $i < \delta_v$ then
13:        $\mathsf{pointer}_v \leftarrow (p + 1) \bmod \delta_v$
14:        move to the port $p$
```

Lemma 17. *Let 2ℓ agents execute Algorithm 2. If an agent waits at u for a missing edge $e = (u, v)$ for kT rounds, during this time either another agent starts to wait for e at v, or every node is visited by an agent at least once.*

Proof. Suppose that an agent a at u starts to wait for a missing edge (u, v) at round t and (u, v) is kept missing for the next kT rounds (including t).

We first show that there exist T successive rounds in $[t, t+kT)$ during which all the agents but a do not satisfy predicate $\mathsf{Waiting} > kT$ even if their selected edge remains missing.

We show the claim by contradiction. We assume that in any interval of T successive rounds in $[t, t + kT)$, there is an agent that satisfies $\mathsf{Waiting} > kT$.

By assumption, at least k agents other than a must satisfy $\mathsf{Waiting} > kT$, since $kT/T = k$. This means that at least one agent (different from a) satisfies the predicate twice since the number of the agents (excluding a) is $k - 1$. However, once an agent satisfies $\mathsf{Waiting} > kT$ at round $t' \in [t, t + kT)$, the agent never satisfies the predicate again in $[t, t + kT)$ since the length of the interval is kT. This is a contradiction. Thus, there exist T successive rounds in $[t, t + kT)$ during which all the agents (except for a) do not satisfy $\mathsf{Waiting} > kT$ even if their chosen edge is kept missing.

Now, we show the lemma, i.e., show that another agent at v starts to wait for $e = (u, v)$ or the exploration is completed. Suppose that no agent at v starts to wait for e in these T rounds. Since e is missing during these T rounds, during that time the network (without e) can be considered as a $(\ell - 1)$-bounded 1-interval connected TVG. By Theorem 15, $2(\ell - 1) + 1 = 2\ell - 1$ agents complete the exploration of the $(\ell - 1)$-bounded TVGs in these T rounds. This means that every node of the network without e is visited at least once by an agent during these T rounds, because none of them starts to wait for e at v during that time by assumption. Thus, the lemma holds. □

Theorem 18. *In* FSYNC, *any ℓ-bounded 1-interval connected temporal graph $\mathcal{G} \in \mathcal{W}(\ell)$ can be explored by $k = 2\ell$ anonymous agents with knowledge of n and k.*

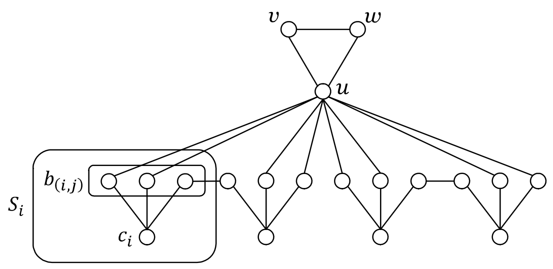

Fig. 3. Example of a graph for $\ell = 3$ and $k = 2\ell - 1 = 5$ that cannot be explored by $2\ell - 1$ agents with a leader.

Proof. Executing Algorithm 2, there clearly exists at least a node v that is visited at an infinite number of rounds (e.g., any node containing an agent blocked forever waiting for an edge that will never appear). We then show that all the neighbors of v are also visited at an infinite number of rounds by agents. We prove it by contradiction. Suppose that a neighbor u of v is visited at only a finite number of rounds and let t' be the last round when u is visited. Since v is visited at an infinite number of rounds and the agents execute Algorithm 2, some agent a visiting v eventually chooses (v, u) as the edge from which a moves after t'. If (v, u) appears by the kT-th round since a chooses it, a visits u as soon as (v, u) appears. Otherwise, another agent visits u by Lemma 17. It follows that u is eventually visited after t', which is a contradiction.

By the connectivity assumption, we can apply inductively the claim (e.g., the neighbors of a neighbor of v are also visited at an infinite number of rounds) to all the nodes, proving the theorem. □

From Theorems 11 and 18, we have:

Theorem 19. *In* FSYNC, *with knowledge of n and k, the exploration of all ℓ-bounded 1-interval connected TVGs is possible iff $k \geq 2\ell$.*

5. Exploration of 1-interval connected graphs with a leader

In this section, we continue to consider the class $\mathcal{W}(\ell)$ of 1-interval connected temporal graphs with bounded missing edges, but we turn our attention to the case when one agent, the *leader*, is distinguishable from the others (the *non-leaders*). Also in this setting, we establish tight bounds for the exploration of this class of temporal graphs in SSYNC and in FSYNC showing that the presence of the leader allows the exploration to be performed using one fewer agent.

5.1. Semi-synchronous model

In this section, we show that, if there exists a leader, the bounds decrease by one in the exploration of ℓ-bounded 1-interval TVGs. In fact, we show that, $\mathcal{G} \in \mathcal{W}(\ell)$ can be explored by 2ℓ agents with one leader, while there exist graphs $\mathcal{G} \in \mathcal{W}(\ell)$ that cannot be explored by $2\ell - 1$ agents with one leader.

5.1.1. Impossibility

We start by showing the impossibility result.

Theorem 20. *There exist 1-interval connected time-varying graphs $\mathcal{G} \in \mathcal{W}(\ell)$ that cannot be explored by $k = 2\ell - 1$ agents with a leader. The result holds even if the agents know n, m and k, and whiteboards are of unbounded size.*

Proof. We construct a graph $\mathcal{G}'$ using a graph similar to the one employed in the proof of Theorem 8, where however we use $2\ell - 2$ copies of stars instead of 2ℓ, and we add two new nodes v and w connected to u (see Fig. 3). The subgraph corresponding to $\mathcal{G}$ (including u) is denoted by $\mathcal{G}'_1$ and the subgraph induced by u, v and w is denoted by $\mathcal{G}'_2$.

Let each non-leader agent a_i be on one of the nodes c_i, and the leader agent $\hat{a}$ be on w.

Consider $\mathcal{G}'_2$ and the following behavior of the adversary: whenever $\hat{a}$ chooses the port corresponding to (v, w) the adversary deletes (v, w), otherwise it deletes (u, w). With these dynamics, $\hat{a}$ never visits u; moreover $\hat{a}$ has no effect on the exploration of $\mathcal{G}'_1$.

Consider now $\mathcal{G}'_1$: we let the adversary delete at most $\ell - 1$ edges at each round. Then, by Theorem 8 and since $\hat{a}$ has no effect on the exploration of $\mathcal{G}'_1$, the $2\ell - 2 = 2(\ell - 1)$ non-leader agents are also prevented from visiting u. Clearly, the number of missing edges at each round is at most ℓ and the graph is always connected. □

5.1.2. Exploration by 2ℓ agents with a leader

We now describe a strategy for 2ℓ agents (one of which is a leader) to explore ℓ-bounded 1-interval connected graphs. The general idea is simple: the leader agent always changes its chosen edge whenever it is blocked by a missing edge, while a non-leader agent always waits for its chosen edge to appear.

However, if we implement this strategy using one pointer for each node, like we did in Sections 3 and 4, two problems can occur: (i) a *broken rotor* and (ii) a *skipped port*.

A broken rotor is a pointer that can be changed by the adversary freely. Since the leader changes a pointer whenever it is blocked, the adversary can make the leader choose pointers in such a way that the leader is repeatedly blocked. To avoid this situation, we use an additional pointer $\mathsf{pointerL}_v$ for each node v that only the leader can change.

A skipped port is a port that remains unused. Suppose that a port p_v at node v is occupied by the leader. Since non-leaders skip an occupied port, they continue to skip p_v as long as the leader occupies it. Without restrictions, even if the leader, finding itself blocked at p_v, changes its port and moves away from p_v, it might return to v and occupy p_v always when the edge is missing and whenever a non-leader would be arriving to v, hence the adversary could continue to prevent the use of the port and thus the exploration of the node on the other side.

To avoid these two potential problems, (a) $\mathsf{pointerL}_v$ is changed so that $\mathsf{pointerL}_v$ points to an occupied port p if and only if the agent occupying p is the leader, (b) a non-leader waits for an occupied port to be unoccupied when the port is pointed by $\mathsf{pointerL}_v$, and (c) the leader, as long as finding an agent not on a port, stays at v.

Algorithm 3 is the exploration algorithm of the leader and Algorithm 4 is the exploration algorithm of the non-leaders. In Algorithms 3 and 4, Setting $\mathsf{pointerL}_v$ to -1 is done to prevent $\mathsf{pointerL}_v$ from pointing to a port occupied by a non-leader. We assume that $\mathsf{pointerL}_v$ is initialized to -1.

Algorithm 3 Computation of the leader at node v.

```
1:  if on a port then
2:      exit the current port
3:  if (not on a port) $\wedge$ (all other agents at $v$ are on a port) then
4:      $i \leftarrow 0$
5:      $p \leftarrow \mathsf{pointerL}_v + 1$
6:      while $i < \delta_v \wedge$ port $p$ is occupied do
7:          $p \leftarrow (p+1) \bmod \delta_v$
8:          $i \leftarrow i + 1$
9:      if $i < \delta_v$ then
10:         $\mathsf{pointerL}_v \leftarrow p$
11:         move to the port $p$
12:     else
13:         $\mathsf{pointerL}_v \leftarrow -1$
```

Algorithm 4 Computation of a non-leader at node v.

```
1:  if not on a port then
2:      $i \leftarrow 0$
3:      $p \leftarrow \mathsf{pointer}_v$
4:      while $i < \delta_v \wedge$ port $p$ is occupied do
5:          if $p = \mathsf{pointerL}_v$ then
6:              $i \leftarrow \delta_v$, $\mathsf{pointer}_v \leftarrow p$
7:              break from this loop
8:          $p \leftarrow (p+1) \bmod \delta_v$
9:          $i \leftarrow i + 1$
10:     if $i < \delta_v$ then
11:         $\mathsf{pointer}_v \leftarrow (p+1) \bmod \delta_v$
12:         if $p = \mathsf{pointerL}_v$ then
13:             $\mathsf{pointerL}_v \leftarrow -1$
14:         move to the port $p$
```

First, we show that $\mathsf{pointerL}_v$ behaves correctly.

Lemma 21. *Variable $\mathsf{pointerL}_v$ points at an occupied port if and only if the agent occupying the port is the leader.*

Proof. ($\Leftarrow$) When the leader moves to a port p, it changes $\mathsf{pointerL}_v$ to p.

($\Rightarrow$) Let us show the contraposition of the claim: if the agent occupying port p is a non-leader, $\mathsf{pointerL}_v$ never points to p. At the beginning, the value of each $\mathsf{pointerL}_v$ is -1 and thus the claim holds. If a non-leader, say a_i, decides to move to a port p and $\mathsf{pointerL}_v$ points to p, then a_i changes $\mathsf{pointerL}_v$ to -1 before moving to p. By induction, $\mathsf{pointerL}_v$ never points to a port occupied by a non-leader. □

Theorem 22. *Any ℓ-bounded 1-interval connected temporal graph $\mathcal{G} \in \mathcal{W}(\ell)$ can be explored by $k = 2\ell$ agents with a leader under the semi-synchronous scheduler with eventual transport.*

Proof. Consider the leader executing Algorithm 3 and the $2\ell - 1$ non-leaders executing Algorithm 4. First, we show that unless $2\ell - 1$ non-leaders are all blocked forever, the $2\ell - 1$ non-leaders visit all the nodes infinitely often. Note that the adversary needs ℓ transient edges to block $2\ell - 1$ non-leaders.

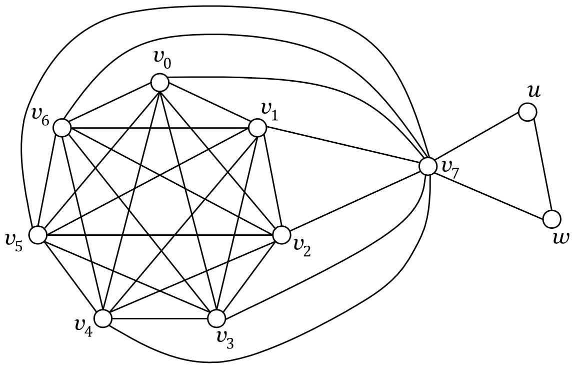

Fig. 4. Example of a graph for $\ell = 5$ and $k = 2\ell - 2 = 8$ that cannot be explored by $2\ell - 2$ agents with one leader. It is constructed with the graph in Fig. 2 and nodes u and w being connected to $v_{2\ell-3}$.

First assume that some non-leader agents can move from a node to another node infinitely often. Let A_m be the non empty set of such non-leaders, let t_e be the round such that after t_e, every agent $b \notin A_m$ is kept blocked forever, and let $x(t)$ be the total number of agent moves from a node to another node over all the agents in A_m from round t_e of the execution up to time t. Since $a \in A_m$ is never blocked by a transient edge, $x(t)$ goes to infinity (for $t \to \infty$). Thus, there exists a node, say v, which is visited at an infinite number of rounds by $a \in A_m$. Then, by an argument similar to the one used in the proof of Theorem 5, we can show that every neighbor of v connected with a recurrent edge is also visited at an infinite number of rounds by $a \in A_m$ and, inductively, that all the nodes are visited at an infinite number of rounds.

Suppose instead that every non-leader agent is blocked at some port forever after some round, and let $t'_e > t_e$ be a round when they are all blocked and all the 2ℓ transient edges have disappeared forever. In this case, we show that the leader completes the exploration. First observe that, since all the non-leaders are blocked at some port, after round t'_e the leader is never required to stop to wait for non-leaders to move to a port.

Moreover, since ℓ missing edges are transient and do not exist anymore after time t'_e, from this time, the network can be regarded as a static network with 2ℓ unusable ports: the $2\ell - 1$ occupied by non-leaders and one unoccupied. The leader, by construction, just skips the ports that are not available. In doing so, it executes the rotor-router algorithm on the static network induced by deleting all the transient edges from the footprint of the network. Hence, by the property of the rotor-router algorithm, the leader correctly performs the exploration. □

From Theorems 20 and 22, we have:

Theorem 23. *In* Ssync, *with a leader, the exploration of all ℓ-bounded 1-interval connected TVGs is possible iff $k \geq 2\ell$.*

5.2. Fully-synchronous model

In this section, we show that, if there exists one leader and the agents are activated in Fsync, the bounds on the number of agents for exploration in ℓ-bounded 1-interval TVGs decreases even further. In fact, we show that, with a leader, $\mathcal{G} \in \mathcal{W}(\ell)$ can be explored by $2\ell - 1$ agents if $\ell \geq 2$ (it is clear that when $\ell = 1$ and $k = 2\ell - 1 = 1$, the exploration is impossible), while there exist graphs $\mathcal{G} \in \mathcal{W}(\ell)$ that cannot be explored by $2\ell - 2$ agents.

5.2.1. Impossibility

We now consider ℓ-bounded, 1-interval connected TVGs operating under a fully-synchronous scheduler and we show that there exist TVGs that cannot be explored by $2\ell - 2$ agents with one leader ($2\ell - 3$ non-leaders and one leader agent), even if the agents know n, m, and k.

Theorem 24. *In* Fsync, *there exist 1-interval connected time-varying graphs $\mathcal{G} \in \mathcal{W}(\ell)$ that cannot be explored by $k = 2\ell - 2$ agents with one leader. The result holds even if the agents know n, m and k, and whiteboards are of unbounded size.*

Proof. We construct a graph $K'_{2\ell-2}$ by adding two nodes u and w to the graph $K_{2\ell-2}$ used in the proof of Theorem 11 and connecting them to $v_{2\ell-3}$ (see Fig. 4). The subgraph corresponding to $K_{2\ell-2}$ (including $v_{2\ell-3}$) is denoted by $K_{(1)}$ and the subgraph induced by $v_{2\ell-3}$, u, and w is denoted by $K_{(2)}$.

Let each non-leader a_i be on each v_i and let the leader agent $\hat{a}$ be on w. Consider $K_{(2)}$ and the following behavior of the adversary: whenever $\hat{a}$ chooses the port corresponding to $(v_{2\ell-3}, w)$ the adversary deletes $(v_{2\ell-3}, w)$, otherwise it deletes (u, w). With these dynamics, $\hat{a}$ never visits $v_{2\ell-3}$; moreover, $\hat{a}$ has no effect on the exploration of $K_{(1)}$.

For $K_{(1)}$, we let the adversary delete at most $\ell - 1$ edges at each round. Then, by Theorem 11 and since $\hat{a}$ has no effect on the exploration of $K_{(1)}$, the $2\ell - 3 = 2(\ell - 1) - 1$ non-leaders are also prevented from visiting $v_{2\ell-3}$. Clearly, the number of missing edges at each round is at most ℓ and the graph is always connected. □

5.2.2. Bound on exploration time

To prove that $2\ell - 1$ agents (one of them the leader) suffice to explore a graph $\mathcal{G} \in \mathcal{W}(\ell)$, we first establish in this subsection an auxiliary result. More precisely, we determine an upper bound on the time sufficient for 2ℓ agents (one of them the leader) to explore $\mathcal{G}$ using Algorithms 3 and 4 described in Section 5.1.2.

We establish the bound through a sequence of lemmas. We start by showing that the leader executing Algorithm 3 and $2\ell - 1$ non-leaders executing Algorithm 4 cannot be all prevented from moving or changing their port at any given round.

Lemma 25. *If 2ℓ agents activated fully-synchronously execute Algorithms 3 (for the leader) and 4 (for the non-leaders) in ℓ-bounded 1-interval TVGs, at least one of them succeeds to change its location at every round (i.e., moving to a port or a neighbor, or changing its port).*

Proof. We have two cases as in the proof of Lemma 4: at round t, (i) there exists a node v containing more than $\delta_v - 1$ agents, or (ii) there does not exist such a node.

In the first case, we can show the claim by the same argument used in the proof of Lemma 4.

We then consider the second case. If some agent is not on a port, this agent moves to a port or a neighbor. If every agent is on a port, the leader tries to change its port by construction. Note that since every node v is occupied by at most $\delta_v - 1$ agents, there is at least one unoccupied port at every node. Thus, the leader succeeds to change its port. □

Using the same argument as the one of the proof of Lemma 13, we have:

Lemma 26. *In an augmented execution of Algorithm 3 executed by the leader and Algorithm 4 executed by the $2\ell - 1$ non-leaders, any two augmented configurations are different.*

Proof. We can show the lemma by the same argument used in the proof of Lemma 13. □

We are now ready to show an upper bound on the exploration time of Algorithms 3 and 4, which is obtained by calculating the maximum length among all the augmented executions.

Lemma 27. *The length of any possible augmented execution of Algorithm 3 executed by the leader and Algorithm 4 executed by the $2\ell - 1$ non-leaders is bounded by $\Delta^n(\Delta+1)^{k+n}(n-1)^k$.*

Proof. Let α be the maximum length among all the possible augmented executions. By Lemma 26, α is bounded by the number of possible augmented configurations in an execution.

The number of possible configurations on a fixed node set $V' \subseteq V$ is bounded by $(\Delta(\Delta+1))^{|V'|}(|V'|(\Delta+1))^k$, which corresponds to all the combinations of the possible values of $\mathsf{pointer}_v$ and $\mathsf{pointerL}_v$ (i.e., $(\Delta(\Delta+1))^{|V'|}$) and all of the agents' locations (i.e., $(|V'|(\Delta+1))^k$). Notice that only $\mathsf{pointer}_v$ and $\mathsf{pointerL}_v$ of each node v are used as variables in Algorithms 3 and 4. Since the number of visited nodes is not decreasing during the exploration, the exploration time is smaller than or equal to the sum of $(\Delta(\Delta+1))^{|V'|}(|V'|(\Delta+1))^k$ for $1 \le |V'| \le n-1$, i.e., $\alpha \le \sum_{|V'|=1}^{n-1}(\Delta(\Delta+1))^{|V'|}(|V'|(\Delta+1))^k \le \Delta^n(\Delta+1)^{k+n}(n-1)^k$ rounds. □

It then follows that:

Theorem 28. *In* Fsync, *the leader executing Algorithm 3 and $2\ell - 1$ non-leaders executing Algorithm 4 explore any ℓ-bounded 1-interval connected TVG within $\Delta^n(\Delta+1)^{k+n}(n-1)^k$ rounds.*

Note that, as a consequence, we obtain a *terminating exploration* algorithm for ℓ-bounded 1-interval connected TVGs.

Theorem 29. *In* Fsync, *with knowledge of n and k, and the existence of a leader,* exploration with termination *of an arbitrary ℓ-bounded 1-interval connected temporal graph $\mathcal{W}(\ell)$ can be achieved in $n^{2(n+k)}$ rounds by $k = 2\ell$ agents.*

5.2.3. Exploration by $2\ell - 1$ agents with a leader

The result of the previous section can be used to obtain a perpetual exploration algorithm of ℓ-bounded 1-interval connected graphs by $2\ell - 1$ agents (which know n and k) one of which is a distinguishable leader. The solution (Algorithm 5 below) is obtained by applying Algorithm 4, appropriately bounding the waiting time of a non-leader blocked on a missing edge. The algorithm for the leader is the same as the one used in the previous section (i.e., Algorithm 3).

In fact, while a non-leader keeps waiting for a missing edge forever in Algorithm 4, in Algorithm 5 a non-leader waits for a missing edge up to $(k-1)T$ rounds where T is calculated on the basis of the results of Section 5.2.2.

Apart from the waiting time, Algorithm 5 is the same as Algorithm 4: each node has $\mathsf{pointer}_v$ pointing at a port and $\mathsf{pointerL}_v$ for the leader. When a non-leader a_i visits v, a_i checks each port in ascending order from the port pointed by $\mathsf{pointer}_v$. If a_i finds the port occupied by the leader, a_i waits till the leader leaves the port. If a_i finds an unoccupied port p, a_i moves to the port and sets $\mathsf{pointer}_v$ to $p+1$ and, if p is equal to $\mathsf{pointerL}_v$, it sets $\mathsf{pointerL}_v$ to -1. If a_i finishes to check all the ports and they all are occupied, a_i does nothing.

Variable Waiting of a non-leader represents the elapsed time since the last round when the non-leader moved to the current port.

Algorithm 5 Computation of a non-leader at node v.

```
1: if on a port then
2:    Waiting ← Waiting + 1
3:    if Waiting > $(k-1)T$ then
4:       exit the current port
5: if not on a port then
6:    Waiting ← 0
7:    $i \leftarrow 0$
8:    $p \leftarrow \mathsf{pointer}_v$
9:    while $i < \delta_v \wedge$ port $p$ is occupied do
10:      if $p = \mathsf{pointerL}_v$ then
11:         $i \leftarrow \delta_v$, $\mathsf{pointer}_v \leftarrow p$
12:         break from this loop
13:      $p \leftarrow (p+1) \bmod \delta_v$
14:      $i \leftarrow i+1$
15:   if $i < \delta_v$ then
16:      $\mathsf{pointer}_v \leftarrow (p+1) \bmod \delta_v$
17:      if $p = \mathsf{pointerL}_v$ then
18:         $\mathsf{pointerL}_v \leftarrow -1$
19:      move to the port $p$
```

Lemma 30. *Let the leader execute Algorithm 3 and the $2\ell-2$ non-leaders execute Algorithm 5. If a non-leader waits at u for a missing edge $e=(u,v)$ for $(k-1)T$ rounds starting from round t, then in $[t, t+(k-1)T)$ rounds there exist T successive rounds during which all the non-leaders do not satisfy predicate Waiting $> (k-1)T$ even if their selected edge remains missing.*

Proof. Suppose that a non-leader a_i at u starts to wait for a missing edge (u,v) at round t and (u,v) is kept missing for the next $(k-1)T$ rounds (including t).

We show the lemma by contradiction. We assume that in any interval of T successive rounds in $[t, t+(k-1)T)$, there is a non-leader that satisfies Waiting $> (k-1)T$.

By assumption, at least $k-1$ non-leaders other than a_i must satisfy Waiting $> (k-1)T$ since $(k-1)T/T = k-1$. This means that at least one non-leader (different from a_i) satisfies the predicate twice since the number of non-leaders (excluding a_i) is $k-2$. However, once a non-leader satisfies Waiting $> (k-1)T$ at round $t' \in [t, t+(k-1)T)$, the non-leader never satisfies the predicate again in $[t, t+(k-1)T)$ since the length of the interval is $(k-1)T$. This is a contradiction. □

Lemma 31. *Let the leader execute Algorithm 3 and the $2\ell-2$ non-leaders execute Algorithm 5. If a non-leader waits at u for a missing edge $e=(u,v)$ for $(k-1)T$ rounds, during this time either another non-leader starts to wait for e at v, or every node is visited by an agent at least once.*

Proof. Suppose that a non-leader a_i at u starts to wait for a missing edge (u,v) at round t and (u,v) is kept missing for the next $(k-1)T$ rounds (including t).

By Lemma 30, in interval $[t, t+(k-1)T)$ there exist T successive rounds, say $\mathcal{I}$, during which all the non-leaders do not satisfy predicate Waiting $> (k-1)T$ even if their selected edge remains missing. Suppose that no non-leader starts to wait for e at v in $\mathcal{I}$. Since e is missing during $\mathcal{I}$, the network (without e) can be considered as a $(\ell-1)$-bounded 1-interval connected TVG during $\mathcal{I}$. By Theorem 28, $2\ell-2 = 2(\ell-1)$ agents with one leader complete the exploration of the $(\ell-1)$-bounded TVGs in the T successive rounds. This means that every node of the network (without e) is visited by an agent during $\mathcal{I}$ at least once because none of the non-leaders starts to wait for e at v during that time by assumption. Thus, the lemma holds. □

Lemma 32. *Let the leader execute Algorithm 3 and the $2\ell-2$ non-leaders execute Algorithm 5. If there is a node visited at only a finite number of rounds (by the leader or non-leaders) and there is another node visited at only a finite number of rounds by non-leaders, every node is visited at only a finite number of rounds by non-leaders.*

Proof. Suppose that a node v is visited at only a finite number of rounds by agents and another node u is visited at only a finite number of rounds by non-leaders. Let t_1 be the last round when v is visited by an agent or u is visited by non-leaders.

We first show that all the neighbors of u are visited at only a finite number of rounds by non-leaders. We prove this by contradiction, assuming that a neighbor w of u is visited at an infinite number of rounds by non-leaders. Eventually, an agent a_1 at w chooses (w,u) to move after t_1. Since u is never visited at any time after t_1, a_1 is kept blocked on w at the port of (w,u) for $(k-1)T$ rounds. By Lemma 31, however, another non-leader visits u or every node is visited by an agent at least once after t_1. Both cases contradict the assumption and thus all the neighbors of u are visited at only a finite number of rounds by non-leaders. Since the network is connected, we can apply inductively the claim to all the nodes, proving the lemma. □

Theorem 33. *In* FSYNC, *with knowledge of n and k, if $\ell \geq 2$, any ℓ-bounded 1-interval connected temporal graph $\mathcal{G} \in \mathcal{W}(\ell)$ can be explored by $k = 2\ell - 1$ agents with one leader.*

Proof. Consider the leader executing Algorithm 3 and non-leaders executing Algorithm 5. The proof follows the same lines of the one of Theorem 18. There clearly exists at least one node v which is visited at an infinite number of rounds. We then show that all the nodes are visited at an infinite number of rounds. Two cases are considered: *Case a)* v is visited at an infinite number of rounds by non-leaders and *Case b)* v is visited at only a finite number of rounds by non-leaders.

Case a) Suppose that v is visited at an infinite number of rounds by non-leaders. We show that all the neighbors of v are also visited at an infinite number of rounds by agents. We prove it by contradiction, assuming that a neighbor u of v is visited at only a finite number of rounds by agents and letting t_1 be the last round when u is visited by an agent. Since v is visited at an infinite number of rounds by the non-leaders executing Algorithm 5, some non-leader a_i visiting v eventually chooses (v,u) to move. If (v,u) appears within $(k-1)T$ rounds, a_i visits u in the period, which is a contradiction. Otherwise, another agent visits u by Lemma 31. It follows that u is eventually visited after t_1, which is a contradiction.

Case b) Suppose that v is visited at only a finite number of rounds by non-leaders. We show by contradiction that all the neighbor of v are visited at an infinite number of rounds. Assume that a neighbor of v is visited at only a finite number of rounds by agents. It follows by Lemma 32 that every node is visited at only a finite number of rounds by non-leaders. This means that no non-leader exists in the network by the definition. This is a contradiction since $\ell \geq 2$ and at least two non-leaders exist in the network.

In either case, all the neighbors of v are visited at an infinite number of rounds. Since the network is connected, we can apply inductively the claim to all the nodes, proving the theorem. □

From Theorems 24 and 33, we have:

Theorem 34. *Under the fully-synchronous scheduler, with knowledge of n and k and the existence of a leader, if $\ell \geq 2$, the exploration of all ℓ-bounded 1-interval connected TVGs is possible iff $k \geq 2\ell - 1$.*

6. Conclusion

In this paper, we considered perpetual exploration of temporal graphs with arbitrary and unknown topology, focusing on the number of agents that are necessary and sufficient to perform the task. We considered two common dynamic models: temporally connected networks, and 1-interval connected (or always connected) networks with a bounded number of missing edges at each round. We derived tight bounds for both models under fully synchronous and semi-synchronous settings, both when the agents are anonymous and when there is a leader.

Our algorithms use at each node v a rotor-router mechanism; this can be implemented with either a constant number of movable tokens that can be placed on the ports of v, or with a whiteboard of size $O(\log \delta_v)$ bits. It would be interesting if the same tight bounds on the number of agents could be obtained using a different (perhaps more complex) mechanism requiring less memory.

In this paper the focus has been on optimality in terms of number of agents. The investigation on the amount of time and the number of moves required by an agent-optimal solution to visit all the nodes at least once during the perpetual exploration, is another open research direction.

This is the first study on distributed exploration of temporal graphs with arbitrary topology, and it has considered so far temporally connected and 1-interval connected networks. The investigation of other classes of temporal dynamics in networks of arbitrary topology is the main research direction left open.

CRediT authorship contribution statement

All authors have contributed to all aspects of the paper: design, ideas, writing, revisions.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research was partly supported by NSERC through the Discovery Grant program Grant numbers A2415 and 203254, and by JSPS KAKENHI Grant Number 19H04085.

References

[1] C. Shannon, Presentation of a maze-solving machine, in: Proc. of the 8th Conf. of the Josiah Macy Jr. Foundation (Cybernetics), 1951, pp. 173–180.
[2] S. Albers, M. Henzinger, Exploring unknown environments, SIAM J. Comput. 29 (4) (2000) 1164–1188.
[3] R. Cohen, P. Fraigniaud, D. Ilcinkas, A. Korman, D. Peleg, Label-guided graph exploration by a finite automaton, ACM Trans. Algorithms 4 (4) (2008) 1–18.
[4] J. Chalopin, P. Flocchini, B. Mans, N. Santoro, Network exploration by silent and oblivious robots, in: Proc. of 36th International Workshop on Graph Theoretic Concepts in Computer Science (WG), 2010, pp. 208–219.
[5] X. Deng, C.H. Papadimitriou, Exploring an unknown graph, J. Graph Theory 32 (3) (1999) 265–297.
[6] Y. Dieudonné, A. Pelc, Deterministic network exploration by anonymous silent agents with local traffic reports, ACM Trans. Algorithms 11 (2) (2014) 1–29.
[7] S. Dobrev, L. Narayanan, J. Opatrny, D. Pankratov, Exploration of high-dimensional grids by finite automata, in: Proc. 46th Int. Colloquium on Automata, Languages, and Programming (ICALP), 2019, pp. 1–16.
[8] P. Fraigniaud, D. Ilcinkas, G. Peer, A. Pelc, D. Peleg, Graph exploration by a finite automaton, Theor. Comput. Sci. 345 (2–3) (2005) 331–344.
[9] P. Fraigniaud, D. Ilcinkas, A. Pelc, Impact of memory size on graph exploration capability, Discrete Appl. Math. 156 (12) (2008) 2310–2319.
[10] P. Panaite, A. Pelc, Exploring unknown undirected graphs, J. Algorithms 33 (1999) 281–295.
[11] S. Das, Graph Exploration with Mobile Agents, Chapter 16 of [40] (2019) 403–422.
[12] A. Casteigts, P. Flocchini, W. Quattrociocchi, N. Santoro, Time-varying graphs and dynamic networks, Int. J. Parallel Emerg. Distrib. Syst. 27 (5) (2012) 387–408.
[13] F. Harary, G. Gupta, Dynamic graph models, Math. Comput. Model. 25 (7) (1997) 79–88.
[14] A. Ferreira, Building a reference combinatorial model for MANETs, IEEE Netw. 18 (5) (2004) 24–29.
[15] M. Bournat, S. Dubois, F. Petit, Computability of perpetual exploration in highly dynamic rings, in: Proc. IEEE 37th Int. Conference on Distributed Computing Systems (ICDCS), 2017, pp. 794–804.
[16] A. Casteigts, P. Flocchini, B. Mans, N. Santoro, Deterministic computations in time-varying graphs: broadcasting under unstructured mobility, in: Proc. of IFIP International Conference on Theoretical Computer Science (TCS), 2010, pp. 111–124.
[17] A. Casteigts, P. Flocchini, B. Mans, N. Santoro, Measuring temporal lags in delay-tolerant networks, IEEE Trans. Comput. 63 (2) (2014) 397–410.
[18] T. Erlebach, J.T. Spooner, A game of cops and robbers on graphs with periodic edge-connectivity, in: 46th International Conference on Current Trends in Theory and Practice of Informatics (SOFSEM), 2020, pp. 64–75.
[19] P. Flocchini, M. Kellett, P. Mason, N. Santoro, Searching for black holes in subways, Theory Comput. Syst. 50 (1) (2012) 158–184.
[20] P. Flocchini, B. Mans, N. Santoro, On the exploration of time-varying networks, Theor. Comput. Sci. 469 (2013) 53–68.
[21] D. Ilcinkas, A. Wade, On the power of waiting when exploring public transportation systems, in: Proc. 15th International Conference on Principles of Distributed Systems (OPODIS), 2011, pp. 451–464.
[22] D. Ilcinkas, A. Wade, Exploration of the T-interval-connected dynamic graphs: the case of the ring, Theory Comput. Syst. 62 (4) (2018) 1144–1160.
[23] B. Haeupler, F. Kuhn, Lower bounds on information dissemination in dynamic networks, in: Proc. 26th International Symposium on Distributed Computing (DISC), 2012, pp. 166–180.
[24] F. Kuhn, N.A. Lynch, R. Oshman, Distributed computation in dynamic networks, in: Proc. 42nd ACM Symposium on Theory of Computing (STOC), 2010, pp. 513–522.
[25] F. Kuhn, R. Oshman, Coordinated consensus in dynamic networks, in: Proc. 30th Symposium on Principles of Distributed Computing (PODC), 2011, pp. 1–10.
[26] G. Di Luna, Mobile Agents on Dynamic Graphs, Chapter 20 of [40] (2019) 549–584.
[27] M. Bournat, S. Dubois, F. Petit, Gracefully degrading gathering in dynamic rings, in: 20th International Symposium on Stabilization, Safety, and Security of Distributed Systems, 2018, pp. 349–364.
[28] G. Di Luna, P. Flocchini, L. Pagli, G. Prencipe, N. Santoro, G. Viglietta, Gathering in dynamic rings, Theor. Comput. Sci. 811 (2020) 79–98.
[29] A. Agarwalla, J. Augustine, W. Moses, S. Madhav, A. Sridhar, Deterministic dispersion of mobile robots in dynamic rings, in: 19th International Conference on Distributed Computing and Networking, 2018, pp. 19:1–19:4.
[30] O. Michail, P. Spirakis, Traveling salesman problems in temporal graphs, Theor. Comput. Sci. 634 (2016) 1–23.
[31] T. Erlebach, M. Hoffmann, F. Kammer, On temporal graph exploration, in: Proc. of 42nd International Colloquium on Automata, Languages, and Programming (ICALP), 2015, pp. 444–455.
[32] T. Erlebach, J.T. Spooner, Faster exploration of degree-bounded temporal graphs, in: Proc. of 43rd International Symposium on Mathematical Foundations of Computer Science (MFCS), 2018, pp. 1–13.
[33] T. Erlebach, F. Kammer, K. Luo, A. Sajenko, J. Spooner, Two moves per time step make a difference, in: Proc. 46th International Colloquium on Automata, Languages, and Programming (ICALP), vol. 141, 2019, pp. 1–14.
[34] D. Ilcinkas, R. Klasing, A. Wade, Exploration of constantly connected dynamic graphs based on cactuses, in: Proc. 21st International Colloquium Structural Information and Communication Complexity (SIROCCO), 2014, pp. 250–262.
[35] C. Avin, M. Koucky, Z. Lotker, How to explore a fast-changing world, in: Proc. of the 35th Int. Colloquium on Automata, Languages and Programming (ICALP), 2008, pp. 121–132.
[36] M. Bournat, A. Datta, S. Dubois, Self-stabilizing robots in highly dynamic environments, Theor. Comput. Sci. 772 (2019) 88–110.
[37] G. Di Luna, S. Dobrev, P. Flocchini, N. Santoro, Distributed exploration of dynamic rings, Distrib. Comput. 33 (2020) 41–67.
[38] T. Gotoh, Y. Sudo, F. Ooshita, H. Kakugawa, T. Masuzawa, Group exploration of dynamic tori, in: Proc. IEEE 38th International Conference on Distributed Computing Systems (ICDCS), 2018, pp. 775–785.
[39] T. Gotoh, Y. Sudo, F. Ooshita, T. Masuzawa, Exploration of dynamic ring networks by a single agent with the h-hops and s-time steps view, in: International Symposium on Stabilizing, Safety, and Security of Distributed Systems, Springer, 2019, pp. 165–177.

[40] P. Flocchini, G. Prencipe, N.S. (Eds.), Distributed Computing by Mobile Entities, Springer, 2019.
[41] Y. Afek, E. Gafni, Distributed algorithms for unidirectional networks, SIAM J. Comput. 23 (6) (1994) 1152–1178.
[42] E. Bampas, L. Gasieniec, N. Hanusse, D. Ilcinkas, R. Klasing, A. Kosowski, T. Radzik, Robustness of the rotor-router mechanism, Algorithmica 78 (3) (2017) 869–895.
[43] A.S. Fraenkel, Economic traversal of labyrinths, Math. Mag. 43 (1970) 125–130.
[44] A. Kosowski, D. Pajak, Does adding more agents make a difference? A case study of cover time for the rotor-router, J. Comput. Syst. Sci. 106 (2019) 80–93.
[45] V. Yanovsky, A. Bruckstein, I. Wagner, A distributed ant algorithm for efficiently patrolling a network, Algorithmica 37 (3) (2003) 165–186.