Tight Bounds on Distributed Exploration of Temporal Graphs

Tsuyoshi Gotoh
Osaka University, Japan

Paola Flocchini
University of Ottawa, Canada

Toshimitsu Masuzawa
Osaka University, Japan

Nicola Santoro
Carleton University, Canada

Abstract

Temporal graphs (or evolving graphs) are time-varying graphs where time is assumed to be discrete. In this paper, we consider for the first time the problem of exploring temporal graphs of *arbitrary unknown topology*. We study the feasibility of exploration, under both the FSYNC and SSYNC schedulers, focusing on the number of agents necessary and sufficient to explore such graphs.

We first consider the minimal (i.e., less restrictive) assumption on the dynamics of the graph under which exploration is still feasible: *temporal connectivity*. Let $\mathcal{H}$ be the class of temporally connected graphs; we show that for any temporal graph $\mathcal{G} \in \mathcal{H}$ 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. More precisely, any $\mathcal{G} \in \mathcal{H}$ can be explored by a team of $k \geq 2\eta(\mathcal{G}) + 1$ agents; this bound is tight as we prove there are $\mathcal{G} \in \mathcal{H}$ that cannot be explored by $2\eta(\mathcal{G})$ agents.

We then turn our attention to the well-known stronger assumption on the dynamics of the graph, called *1-interval connectivity*: the graph is connected at any time step. Let $\mathcal{W} \subset \mathcal{H}$ be the class of these always-connected temporal graphs. For this class, we prove the existence of a difference between FSYNC and SSYNC when there is a bound ℓ on the number of edges missing at each time. In fact, we show a tight bound of $2\ell + 1$ on the number of agents necessary and sufficient in SSYNC, and a smaller tight bound of 2ℓ in FSYNC. As a corollary, we re-establish two recently published bounds for 1-interval connected rings.

2012 ACM Subject Classification Theory of computation → Distributed algorithms; Theory of Computation → Distributed computing models;

Keywords and phrases Distributed algorithm, Mobile agents, Exploration of dynamic networks, Arbitrary footprint

Digital Object Identifier 10.4230/LIPIcs.OPODIS.2019.23

Acknowledgements This research was partly supported by NSERC through the Discovery Grant program, by Prof. Flocchini's University Research Chair, ...

1 Introduction

1.1 Framework and Background

The *graph exploration* problem (EXPLORATION), first introduced by Shannon [34], 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 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 networks with mobile entities (e.g., software agents,

23rd International Conference on Principles of Distributed Systems (OPODIS 2019).
Editors: Pascal Felber, Roy Friedman, Seth Gilbert, and Avery Miller; Article No. 23; pp. 23:1–23:16
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

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 labelings or are anonymous, on the type of communication mechanisms available to the agents, on the degree of synchronization of the network, on the level of knowledge the agents have about the graph, on their memory, etc. (e.g., see [1, 8, 7, 10, 13, 14, 21, 22, 33, 35], and [9] for a recent survey). In spite of all the differences, the existing literature has until very recently made a common assumption: the graph is *static*, i.e., the link structure does not change during the exploration.

Recently, researchers in the distributed computing community have started to investigate *highly dynamic graphs*, that is graphs where the topological changes are not sporadic or anomalous, but rather inherent in the nature of the network. Various models have been proposed to describe highly dynamic networks, under a variety of names. A model that describes them in a simple and natural way is the one of *time-varying graphs*, formally defined in [6], where main classes of systems studied in the literature and their computational relationship were identified. When time is assumed to be *discrete*, the evolution of these systems can be equivalently described as a sequence of static graphs, called *evolving graph* or *temporal graph*, a model suggested in [25], formalized in [17].

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, there is temporal reachability between any two nodes (e.g., [5]). Stronger assumptions include *1-interval connectivity* : the graph is always connected (e.g., [24, 30, 31]); and *T-interval connectivity* : the graph is always connected and every $T > 1$ consecutive rounds contain the same spanning-tree (e.g., [28, 30]). A classification of the most common assumption was done in [6].

While there are several studies on computations by mobile agents moving in temporal graphs (for a recent survey see [11]), the results on the *exploration of temporal graphs* are rather limited. On the probabilistic side, there is an early seminal work on random walks [2]. On the deterministic side there are: the study of the complexity of computing a foremost exploration schedule under the 1-interval-connectivity assumption [32], generalized and extended in [15] and then in [16]; the computation of an exploration schedule for *rings* under the stronger T-interval-connectivity assumption [28]; the computation of an exploration schedule for *cactuses* under the 1-interval-connectivity assumption [26]. These studies are however *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. *Distributed* approaches have 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 [18, 19, 27, 28]. Exploration with termination of 1-interval connected *rings* by two and three agents under both synchronous and semi-synchronous schedulers has been considered in [12]. Perpetual exploration by three agents on temporally connected *rings* has been studied in [4, 5]. Perpetual exploration 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 [23].

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.

1.2 Contributions

In this paper we consider perpetual exploration of time varying graphs whose topology ia arbitrary and unknown to the agents. We focus on solvability of the exploration of such dynamic graphs and we determine 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*, perpetual exploration is trivially impossible to achieve. We thus start our investigation with the class $\mathcal{H}$ of temporally connected temporal graphs. We show that for the graphs $\mathcal{G} \in \mathcal{H}$, the number of agents sufficient to perform exploration is related to the evanescence $\eta(\mathcal{G})$ of the graph, that is the number of transient edges. More precisely, any $\mathcal{G} \in \mathcal{H}$ can be explored by a team of $k \geq 2\eta(\mathcal{G}) + 1$ agents; this bound is tight as we prove 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). On the other hand, the proposed exploration algorithm, based on the rotor router technique, works under very weak conditions (SSYNC scheduler, anonymous agents, no knowledge of topological parameters).

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. For this class, we first show a tight bound of $2\ell + 1$ under the SSYNC scheduler on the number of agents. We then prove the existence of a difference between FSYNC and SSYNC if the network size and the number of agents are known. In fact, in this case, while the bound for SSYNC remains unchanged, we prove a tight bound of 2ℓ for FSYNC. Moreover, we show that if $2\ell + 1$ agents are available in SSYNC, the exploration with termination is possible As a corollary of these results, we re-establish a recently published bound for temporally-connected rings [5] and one for 1-interval connected rings [12].

Note that, 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 differs in the case of FSYNC, showing that the stronger connectivity assumption of $\mathcal{W}$ does not influence the solvability bound in case of semi-synchronous schedulers, but does have an impact for fully synchronous ones.

2 The Model

2.1 The Network

The system 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.

In this paper we consider *discrete* time; that is, $\mathbb{T} = \mathbb{Z}^+$. Since time is discrete, the dynamics of the system can be viewed also in terms of a sequence of static graphs: $\mathcal{S}_\mathcal{G} = G_0, G_1, \ldots, G_t, \ldots$, where $G_t = (V_t, E_t)$ is the graph of the edges present at time t (also called *snapshot* at time t). The TVG in this case is called *temporal graph* (or *evolving graph*). 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.

In a temporal graph, the edge set E can be partition into the set of recurrent edges E^*, and the one of transient edges E^-. Formally, a *recurrent edge* $e^* \in E^*$ is such that $\forall t \in \mathbb{Z}^+, \exists t' > t : \rho(e^*, t') = 1$. In other words, a recurrent edge appears infinitely often. On the other hand, a *transient edge* $e^- \in E^-$ is such that $\exists t \in \mathbb{Z}^+, \forall t' \geq t : \rho(e^-, t') = 0$. In other words, a transient edge eventually ceases to exist forever.

The *solidity* of $\mathcal{G}$ is defined as the number $\sigma(\mathcal{G})$ of recurrent edges, and the *evanescence* of $\mathcal{G}$, denoted by $\eta(\mathcal{G})$, as the number of transient edges (i.e., $\eta(\mathcal{G}) = |E| - \sigma(\mathcal{G})$).

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 $t' \geq t$.

A particularly important class of temporal graphs are *temporally connected* ones:

▶ **Definition 1** (Temporally connected). *A TVG $\mathcal{G}$ 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 tasks; in particular, perpetual exploration (i.e., requiring every node to be visited infinitely often) is trivially impossible if the graph 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 in a particular temporally connected graph, where connectivity is actually guaranteed at every time (*always connected* or *1-interval connected* temporal graphs); 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.

The nodes of $\mathcal{G}$ are anonymous (i.e., they have no IDs) and each node provides a constant amount of local memory called *whiteboard*. Each edge incident to node v is locally labeled by a bijection $\lambda_v : E \rightarrow \{0, \ldots, \delta_v - 1\}$; no other assumptions are made about the labels. Every node v has ports p_i for $0 \leq i \leq \delta_v$ which are used to store at most one agent trying to move through e such that $\lambda_v(e) = i$.

2.2 Mobile agents

A set $A = \{a_0, a_1, \ldots, a_{k-1}\}$ of k agents operate on the network, initially occupying arbitrary positions. Agents are anonymous and have access to their private notebook (local memory) and to whiteboards (memory of nodes).

The agents operate in synchronous rounds, and each round is composed by three phases: Look, Compute, and Move, during which they execute the following actions [20]:

Look: Agent a_i observes the content of its own notebook and of the whiteboard of the node it occupies, and it checks, for each port of the node, if there are other agents at the same node.

COMPUTE: On the basis of the information obtained in the LOOK phase, agent a_i decides whether to move or not. It can write information on the whiteboard[1] and, 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 occupies a port, it tries to move. If the corresponding edge exists, a_i reaches the other side, otherwise it stays on the port to try again at the next round. If a_i does not occupy a port, it does not 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, we need to specify the behavior of the agents that fall asleep on a port when the corresponding edge is missing. In this paper, we assume the weakest rule, called *eventual transport rule* [12], in which the agent sleeping at a port will eventually be activated at a time when the edge corresponding to the port is present. This prevents the adversary from using semi-synchronicity to block an agent forever on a recurrent edge.

2.3 Configuration and execution

A configuration C_t is defined by: the contents of the whiteboards, the local memory of the agents, the locations of the agents, and the snapshot G_t of the temporal graph in the sequence $\mathcal{S}_\mathcal{G}$, at 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 $\mathcal{E}^\mathcal{A}$.

2.4 The Exploration problem

We say that a node v is visited by round t if there exists a round t' $(0 \le t' < t)$ such that an agent occupies v at time t'. We say that the network is explored by round t if every node has been visited by round t.

A *perpetual* exploration algorithm is one where, in every execution, every node is visited infinitely often. An exploration *with termination* algorithm is one where the agents terminate when all nodes have been visited at least once. In this paper we are concerned with *perpetual exploration*.

3 Exploration of temporally connected TVGs

In this section, we show that the feasibility of exploration of temporally connected TVGs is related to their evanescence.

[1] Access to the whiteboard is done in fair mutual exclusion

3.1 Impossibility

Let $\mathcal{H}(\ell) = \{\mathcal{G} \in \mathcal{H} : \eta(\mathcal{G}) \leq \ell\}$ 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 have some topological knowledge (n, m or 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 former type of agents are called (with respect to the number of changes of a selected edge) *finite* and the latter *infinite*.

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

Each component is connected as follows. For S_i^{inf} $(0 \leq i \leq 2\ell - 1)$ and u, each $b_{(i,j)}^{\mathsf{inf}}$ $(0 \leq j \leq 2)$ is connected with u by edge $(b_{(i,j)}^{\mathsf{inf}}, u)$. For S_i^{fin} $(0 \leq i \leq 2\ell - 1)$ and u, each $b_{(i,j)}^{\mathsf{fin}}$ $(j = 0$ or $1)$ is connected with u by edge $(b_{(i,j)}^{\mathsf{fin}}, u)$. In addition to that, for $0 \leq i \leq l - 1$, $b_{(2i,2)}^{\mathsf{fin}}$ and $b_{(2i+1,2)}^{\mathsf{fin}}$ are connected by $(b_{(2i,2)}^{\mathsf{fin}}, b_{(2i+1,2)}^{\mathsf{fin}})$. A graph for $l = 2$ $(k = 4)$ is depicted in Figure 1.

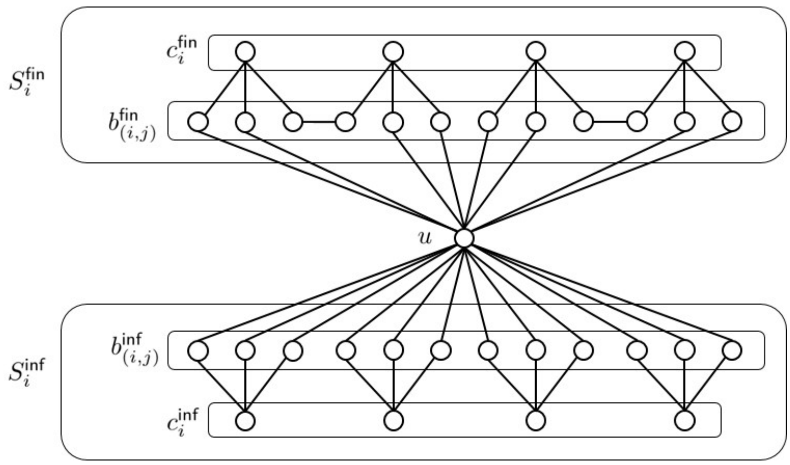

Figure 1 Example of a graph for $\ell = 2$ and $k = 2\ell = 4$. There are four stars S_i^{fin} (S_i^{inf}) for $0 \leq i \leq 3$ on the top (bottom) of the figure. Each star S_i^{fin} (S_i^{inf}) has one center node c_i^{fin} (c_i^{fin}) and three leaf nodes $\{b_{(i,0)}^{\mathsf{fin}}, b_{(i,1)}^{\mathsf{fin}}, b_{(i,2)}^{\mathsf{fin}}\}$ ($\{b_{(i,0)}^{\mathsf{fin}}, b_{(i,1)}^{\mathsf{fin}}, b_{(i,2)}^{\mathsf{fin}}\}$).

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 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^{inf} 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 edge $(b_{(i,j)}^{\mathsf{inf}}, u)$ whenever a_i is on $b_{(i,j)}^{\mathsf{inf}}$. Clearly, this prevents any agent executing $\mathcal{A}$ to visit 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 $\mathcal{E}_2$. To decide the initial configuration of $\mathcal{E}_2$, 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 $(0 \le f \le k-1)$ be the number of *finite agents*. In the following, *finite agents* are denoted by $a_0^{\mathsf{fin}}, \ldots, a_{f-1}^{\mathsf{fin}}$, and the *infinite agents* are denoted by $a_0^{\mathsf{inf}}, \ldots, a_{k-f-1}^{\mathsf{inf}}$. W.l.o.g., we assume that $a_i^{\mathsf{fin}} = a_i$, i.e, a_i^{fin} is the agent starting from c_i^{inf} in $\mathcal{E}_1$.

The adversary decides the initial configuration of $\mathcal{E}_2$ as follows: each a_i^{inf} $(0 \le i \le k-f-1)$ is put on the same node as in the initial configuration of $\mathcal{E}_1$, while each a_i^{fin} $(0 \le i \le f-1)$ is put on c_i^{fin}.

Then, the adversary changes the assignment of the port labels and the node ID (if any) of c_i^{fin}, $b_{(i,0)}^{\mathsf{fin}}$, $b_{(i,1)}^{\mathsf{fin}}$, and, $b_{(i,2)}^{\mathsf{fin}}$ in S_i^{fin} so that a_i^{fin} cannot distinguish $\mathcal{E}_1$ and $\mathcal{E}_2$. Let $v_i = b_{(i,x)}^{\mathsf{inf}}$ be the node where $a_i = a_i^{\mathsf{fin}}$ finally waits a missing edge permanently in $\mathcal{E}_1$. For $b_{(i,2)}^{\mathsf{fin}}$, the assignment of the port labels and the node ID (if any) are copied from v_i. The ones of c_i^{fin} are copied from c_j^{inf}. The ones of $b_{(i,0)}^{\mathsf{fin}}$ and $b_{(i,1)}^{\mathsf{fin}}$ are copied from each of $b_{(i,y)}^{\mathsf{inf}}$ for $y \neq x$.

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 (resp, u or $S_{i'}^{\mathsf{fin}}$ for $i' \neq i$) when $a_{i''}^{\mathsf{inf}} = a_i$ (resp, a_i^{fin}) exists on $b_{(i,j)}^{\mathsf{inf}}$ (resp, $b_{(i,j)}^{\mathsf{fin}}$). Obviously, every agent cannot distinguish $\mathcal{E}_1$ and $\mathcal{E}_2$, since the difference between $\mathcal{E}_1$ and $\mathcal{E}_2$ is only the order of the port labeling (and the node degrees are fixed). Thus, $\mathcal{G}$ cannot be explored since u is not visited by any agent in $\mathcal{E}_2$.

Finally, we show that, in $\mathcal{E}_2$, $\eta(\mathcal{G})$ is at most ℓ. To prevent infinite agents, no transient edge is necessary; in fact, an infinite agent eventually changes its selected edge if it is kept missing, and no two infinite agents wait on the same edge (otherwise, the edge may be transient). For finite agents, by construction, a_{2i}^{fin} and a_{2i+1}^{fin} for $0 \le i \le (f-1)/2$ eventually wait for the same edge $(b_{(2i,2)}^{\mathsf{fin}}, b_{(2i+1,2)}^{\mathsf{fin}})$ (when f is odd, only a_{f-1} waits for $(b_{(f-1,2)}^{\mathsf{fin}}, b_{(f,2)}^{\mathsf{fin}})$). Since f is at most $k = 2\ell$, at most ℓ edges are necessary to prevent finite agents. ◀

3.2 Semi Synchronous Exploration by $2\eta(\mathcal{G}) + 1$ agents

In this section, we show that every temporally connected time-varying network $\mathcal{G} \in \mathcal{H}$ 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., [3, 29, 35]).

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

```
if not on a port then
    i ← 0
    p ← pointer_v
    while i < δ_v ∧ port p is occupied do
        p ← (p + 1) mod δ_v
        i ← i + 1
    if i < δ_v then
        pointer_v ← (p + 1) mod δ_v
        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, $2l+1$ agents achieve perpetual exploration using Algorithm 1 (Theorem).

▶ **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 within finite time 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 definition of transient edges, there exists a time step t_e such that, for any transient edge e, $\rho(e,t)=0$ 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. Let $x(t)$ be the sum of the number of visits over all the nodes 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 infinitely often. We prove it by contradiction. Suppose that a neighbor u of v connected by a recurrent edge is visited only a finite number of times and let t' be the last round when u is visited. Since v is visited infinitely often 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 is 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 $\mathcal{H}(\ell)$ is possible iff*

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

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

▶ **Theorem 7.** *Every temporally connected temporal graph can be explored by $2(m - n) + 3$ agents.*

4 Exploration of 1-interval connected temporal graphs with bounded missing edges

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, 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 exists TVGs that cannot be explored by 2ℓ 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 nodes and/or agents have distinct IDs and the agents have some topological knowledge (n, m or k).*

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 $\mathcal{A}$, possibily 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^{inf} for $0 \leq i \leq k - 1 = 2\ell - 1$ in 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 $t \equiv i (\mathrm{mod}\ k)$. When the adversary activates a_i and a_i exists on $b^{\mathsf{inf}}_{(i,j)}$, the adversary deletes $(b^{\mathsf{inf}}_{(i,j)}, u)$ 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*, since the eventual transport rule is not violated even in $\mathcal{E}_1$, 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^{fin}_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 c^{fin}_i, $b^{\mathsf{fin}}_{(i,0)}$, $b^{\mathsf{fin}}_{(i,1)}$, and, $b^{\mathsf{fin}}_{(i,2)}$ 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 $S_{i'}$ for $i' \neq i$ whenever a_i is on $b^{\mathsf{fin}}_{(i,j)}$. After some round t from when every agent a_i does not change its selected edge at $b^{\mathsf{fin}}_{(i,2)}$ for $0 \le i \le 2l$, the adversary deletes $(b^{\mathsf{fin}}_{(2j,2)}, b^{\mathsf{fin}}_{(2j+1,2)})$ for $0 \le j \le l-1$ 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 eventually 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.*

From Theorems 8 and 9 it follows that:

▶ **Theorem 10.** *Under a semi-syncrhonous scheduler, exploration of all ℓ-bounded 1-interval connected TVG $\mathcal{G}$ 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-boundend 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 exists TVGs that cannot be explored by $2\ell-1$ agents, even if the agents know n, m, and k.

▶ **Theorem 11.** *There exist 1-interval ℓ-bounded 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 have some topological knowledge (n, m, k).*

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): $E^{(0)}_{2\ell}, E^{(1)}_{2\ell}, \ldots, E^{(2\ell-2)}_{2\ell}$. For example, the following separation leads to disjoint independent edge sets: each $E^{(i)}_{2\ell}$ has ℓ edges, $(v_i, v_{2\ell-1})$, (v_{i-1}, v_{i+1}), (v_{i-2}, v_{i+2}), $\ldots$, (v_{i-l+1}, v_{i+l-1}), see Figure 2 (for simplicity, mod 2ℓ is omitted).

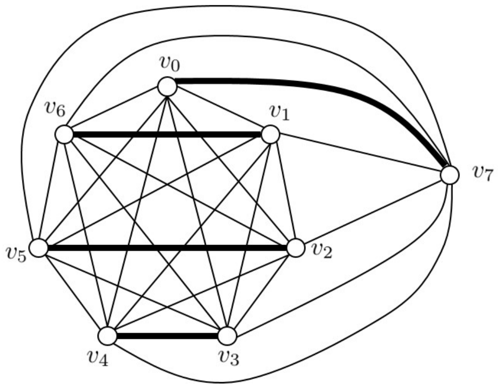

Figure 2 Example of coloring for the proof of Lemma 11. The bold lines are the edges of $E_8^{(0)}$.

The execution where v_{2l-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 $0 \le j \le 2\ell - 1$ 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 its 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. ◀

4.2.2 Bound on Exploration time

Let $\mathcal{G} \in \mathcal{W}(\ell)$. Since $\mathcal{W}(\ell) \subset H(\ell)$, we can clearly execute 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 $2\ell + 1$ 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.

◀

To show the upper-bound on time complexity, we introduce the notions of *augmented configuration* and *augmented execution*.

In an augmented configuration C_t^{aug}, a new variable $\mathsf{visited}_v$ written and read only by an external observer, is added to each node v. The initial value of $\mathsf{visited}_v$ is 0. When v is visited, $\mathsf{visited}_v$ is set to 1 by the external observer. 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 $\mathcal{E}^{\mathsf{aug}} = C_0^{\mathsf{aug}} C_1^{\mathsf{aug}} \dots C_r^{\mathsf{aug}}$ 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 $2\ell + 1$ agents executing one round of Algorithm 1 fully-synchronously, with the action of the adversary deciding which edges are missing; C_r^{aug} is a unique terminal configuration in $\mathcal{E}^{\mathsf{aug}}$. Note that the agents keep executing Algorithm 1 after round r, but augmented configurations after round r are ignored in $\mathcal{E}^{\mathsf{aug}}$. For $\mathcal{E}^{\mathsf{aug}}$, the following lemma holds.

▶ **Lemma 13.** *In an augmented execution 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 where no agents move between C_t^{aug} and C_{t+1}^{aug}. Suppose that there exist two augmented configurations C_t^{aug} and $C_{t'}^{\mathsf{aug}}$ for $t < t'$ in an augmented execution $\mathcal{E}^{\mathsf{aug}}$. Let $\mathcal{E}_{t,t'}^{\mathsf{aug}} = C_t^{\mathsf{aug}} C_{t+1}^{\mathsf{aug}} \cdots C_{t'-1}^{\mathsf{aug}}$ be a subsequence of $\mathcal{E}^{\mathsf{aug}}$. In this case, the adversary can create an infinite augmented execution from $\mathcal{E}^{\mathsf{aug}}$ by repeating $\mathcal{E}_{t,t'}^{\mathsf{aug}}$, 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. ◀

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 directions of pointers (i.e., $\Delta^{|V'|}$) and all of the 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 visited nodes 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 $1 \leq |V'| \leq n - 1$, i.e., $\alpha \leq \Sigma_{|V'|=1}^{n-1} \Delta^{|V'|}(|V'|(\Delta + 1))^k \leq \Delta^n(\Delta + 1)^k(n - 1)^k$ rounds. ◀

It then follows that:

▶ **Theorem 15.** *Under a fully-synchronous scheduler, 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-bouned 1-interval connected TVGs.

▶ **Theorem 16.** *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 $\Delta^n(\Delta+1)^{2\ell+1}(n-1)^{2\ell+1}$ rounds by $2\ell + 1$ agents under the fully synchronous scheduler.*

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 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 to a port. When 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 port.

Algorithm 2 Computation at node v

```
if on a port then
    Waiting ← Waiting + 1
    if Waiting > kT then
        exit the current port
if not on a port then
    Waiting ← 0
    i ← 0
    p ← pointer_v
    while i < δ_v ∧ port p is occupied do
        p ← (p + 1) mod δ_v
        i ← i + 1
    if i < δ_v then
        pointer_v ← (p + 1) mod δ_v
        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 the other $2\ell-1$ agents complete the exploration.*

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 keep waiting without satisfying predicate $\mathsf{Waiting} > kT$ at its chosen edge, if it 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 $kT/T \geq k$. 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 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) keep waiting their chosen edge without satisfying $\mathsf{Waiting} > kT$ if the 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, $2(l - 1) + 1 = 2\ell - 1$ agents complete the exploration of the $(\ell - 1)$-bounded TVGs in these T rounds. This means that the $2\ell - 1$ agents other than a complete the exploration of the network without e during those T rounds, because none of them starts to wait for e at v during that time by assumption. Thus, the lemma holds. ◀

▶ **Theorem 18.** *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, under a fully-synchronous scheduler.*

Proof. The proof follows the same lines of Theorem 5. We first show that, executing Algorithm 2 , there exists at least one node v which is visited infinitely often, and we then show that all the nodes are visited infinitely often. Let $x(t)$ be the sum of the number of visits over all the node from the beginning of the execution up to time t and $V_A^{(t)}$ be a node set such that there exists at least one agent on every $w \in V_A^{(t)}$ at round t.

We show that $x(t)$ goes to infinity (for $t \to \infty$), which leads to the existence of a node v visited infinitely often. We consider the configuration at round t and show that after t rounds, $x(t)$ eventually increases. Two cases are considered: *Case 1)* there exists a node $\hat{v} \in V_A^{(t)}$ with δ_v or more agents and *Case 2)* there does not exist such a node.

Case 1) Suppose that there exists a node $\hat{v}$ with $\delta_{\hat{v}}$ or more agents at round t. Note that at least one of the edges incident to $\hat{v}$ exists at round t because the network is 1-interval connected. In this case, at least one of the agents on $\hat{v}$ succeeds to move because all the ports of $\hat{v}$ are occupied. Therefore, $x(t)$ increases.

Case 2) Suppose that there does not exist a node $\hat{v}$ with $\delta_{\hat{v}}$ or more agents. We show that $x(t)$ increases within finite rounds from t by contradiction. We assume that no agent moves out of its current node after t. Clearly, there exists a node $\tilde{v} \in V_A^{(t)}$ which has a neighbor $\tilde{u}$ with no agent (otherwise, the exploration would have been completed). An agent changes its port if it is blocked by the same missing edge for kT rounds by Algorithm 2; an agent $\tilde{a}$ on $\tilde{v}$ eventually chooses $(\tilde{v}, \tilde{u})$ to move from $\tilde{v}$. At this round, the adversary must prevent $\tilde{a}$ from moving by deleting $(\tilde{v}, \tilde{u})$. This means that the adversary must prevent $2(\ell - 1) + 1 = 2\ell - 1$ agents from moving by deleting $\ell - 1$ edges, which is impossible. This leads to a contradiction. Therefore, $x(t)$ increases and goes to infinity for $t \to \infty$, and thus a node (say v) visited infinitely often exists.

We now show that all the neighbors of v are also visited by agents infinitely often. We prove it by contradiction. Suppose that a neighbor u of v is visited only a finite number of times and let t' be the last round when u is visited. Since v is visited infinitely often 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 chose 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' rounds, 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 infinitely often) to all the nodes, proving the theorem. ◀

From Theorems 11 and 18, we have:

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

5 Conclusion

In this paper, we considered perpetual exploration of temporal graphs with arbitrary 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.

This is the first study on distributed exploration of temporal graphs with *arbitrary topology* and it has considered only temporally connected and 1-interval connected networks: the investigation of other connectivity classes of temporal graphs with arbitrary topology is the main research direction left open.

In this paper the focus was exclusively on feasibility of exploration; clearly, an important avenue of investigation is also the design of efficent solutions, whenever they exist.

References

1. S. Albers and M. Henzinger. Exploring unknown environments. *SIAM Journal on Computing*, 29(4):1164–1188, 2000.
2. C. Avin, M. Koucky, and Z. Lotker. How to explore a fast-changing world. In *Proc. of the 35th Int. Colloquium on Automata, Languages and Programming (ICALP)*, pages 121–132, 2008.
3. E. Bampas, L. Gasieniec, N. Hanusse, D. Ilcinkas, R. Klasing, A. Kosowski, and T. Radzik. Robustness of the rotor-router mechanism. *Algorithmica*, 78(3):869–895, 2017.
4. M. Bournat, A.K. Datta, and S. Dubois. Self-stabilizing robots in highly dynamic environments. *Theoretical Computer Science*, 772:88–110, 2019.
5. M. Bournat, S. Dubois, and F. Petit. Computability of perpetual exploration in highly dynamic rings. In *Proc. IEEE 37th Int. Conference on Distributed Computing Systems (ICDCS)*, pages 794–804, 2017.
6. 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.
7. J. Chalopin, P. Flocchini, B. Mans, and N. Santoro. Network exploration by silent and oblivious robots. In *Proc. of 36th International Workshop on Graph Theoretic Concepts in Computer Science (WG)*, pages 208–219, 2010.
8. R. Cohen, P. Fraigniaud, D. Ilcinkas, A. Korman, and D. Peleg. Label-guided graph exploration by a finite automaton. *ACM Trans. Algorithms*, 4(4):1–18, 2008.
9. S. Das. *Graph Exploration with Mobile Agents*. Chapter 16 of [20], pages 403–422, 2019.
10. X. Deng and C. H. Papadimitriou. Exploring an unknown graph. *Journal of Graph Theory*, 32(3):265–297, 1999.
11. G.A. Di Luna. *Mobile Agents on Dynamic Graphs*. Chapter 20 of [20], pages 549–584, 2019.
12. G.A. Di Luna, S. Dobrev, P. Flocchini, and N. Santoro. Live exploration of dynamic rings. In *Proc. IEEE 36th International Conference on Distributed Computing Systems (ICDCS)*, pages 570–579, 2016.
13. Y. Dieudonné and A. Pelc. Deterministic network exploration by anonymous silent agents with local traffic reports. *ACM Transactions on Algorithms*, 11(2):Article No. 10, 2014.
14. S. Dobrev, L. Narayanan, J. Opatrny, and D. Pankratov. Exploration of high-dimensional grids by finite automata. In *Proc. 46th Int. Colloquium on Automata, Languages, and Programming (ICALP)*, pages 1–16, 2019.
15. T. Erlebach, M. Hoffmann, and F. Kammer. On temporal graph exploration. In *Proc. of 42th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 444–455, 2015.

16 T. Erlebach and J. T. Spooner. Faster exploration of degree-bounded temporal graphs. In *Proc. of 43rd International Symposium on Mathematical Foundations of Computer Science (MFCS)*, pages 1–13, 2018.

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

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

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

20 P. Flocchini, G. Prencipe, and N. Santoro (Eds). *Distributed Computing by Mobile Entities*. Springer, 2019.

21 P. Fraigniaud, D. Ilcinkas, G. Peer, A. Pelc, and D. Peleg. Graph exploration by a finite automaton. *Theoretical Computer Science*, 345(2–3):331–344, 2005.

22 P. Fraigniaud, D. Ilcinkas, and A. Pelc. Impact of memory size on graph exploration capability. *Discrete Applied Mathematics*, 156(12):2310–2319, 2008.

23 T. Gotoh, Y. Sudo, F. Ooshita, H. Kakugawa, and T. Masuzawa. Group exploration of dynamic tori. In *Proc. IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, pages 775–785, 2018.

24 B. Haeupler and F. Kuhn. Lower bounds on information dissemination in dynamic networks. In *Proc. 26th International Symposium on Distributed Computing (DISC)*, pages 166–180, 2012.

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

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

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

28 D. Ilcinkas and A.M. Wade. Exploration of the T-interval-connected dynamic graphs: the case of the ring. *Theory Computing Systems*, 62(4):1144–1160, 2018.

29 A. Kosowski and D. Pajak. Does adding more agents make a difference? a case study of cover time for the rotor-router. *J. Comput. Syst. Sci.*, 106:80–93, 2019.

30 F. Kuhn, N. A. Lynch, and R. Oshman. Distributed computation in dynamic networks. In *Proc. 42nd ACM Symposium on Theory of Computing (STOC)*, pages 513–522, 2010.

31 F. Kuhn and R. Oshman. Coordinated consensus in dynamic networks. In *Proc. 30th Symposium on Principles of Distributed Computing (PODC)*, pages 1–10, 2011.

32 O. Michail and P. Spirakis. Traveling salesman problems in temporal graphs. *Theoretical Computer Science*, 634:1–23, 2016.

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

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

35 V. Yanovsky, A. Bruckstein, and I. Wagner. A distributed ant algorithm for efficently patrolling a network. *Algorithmica*, 37(3):165–186, 2003.