Theory Comput Syst (2018) 62:1144–1160
DOI 10.1007/s00224-017-9796-3

Exploration of the T-Interval-Connected Dynamic Graphs: the Case of the Ring

David Ilcinkas[1] · **Ahmed M. Wade**[2]

Published online: 14 July 2017

Abstract In this paper, we study the T-interval-connected dynamic graphs from the point of view of the time necessary and sufficient for their exploration by a mobile entity (agent). A dynamic graph (more precisely, an evolving graph) is T-interval-connected ($T \geq 1$) if, for every window of T consecutive time steps, there exists a connected spanning subgraph that is stable (always present) during this period. This property of connection stability over time was introduced by Kuhn, Lynch and Oshman (Kuhn et al. 2010) (STOC 2010). We focus on the case when the underlying graph is a ring of size n, and we show that the worst-case time complexity for the exploration problem is $2n - T - \Theta(1)$ time units if the agent knows the dynamics of the graph, and $n + \frac{n}{\max\{1,T-1\}}(\delta - 1) \pm \Theta(\delta)$ time units otherwise, where δ is the maximum time between two successive appearances of an edge.

Keywords Exploration · Dynamic graphs · T-interval-connectivity · Mobile agent

A preliminary version of this paper appeared in the Proceedings of the 20th International Colloquium on Structural Information and Communication Complexity (SIROCCO 2013) [13]. Partially supported by the ANR projects DISPLEXITY (ANR-11-BS02-014) and MACARON (ANR-13-JS02-002).

✉ Ahmed M. Wade
awade@ept.sn

David Ilcinkas
david.ilcinkas@labri.fr

[1] LaBRI, CNRS, Univ. Bordeaux, France

[2] École Polytechnique de Thiès, Thiès, Senegal

1 Introduction

Partly due to the very important increase of the number of communicating objects that we observe today, the distributed computing systems are becoming more and more dynamic. The computational models for static networks are clearly not sufficient anymore to capture the behavior of these new communication networks. One can nevertheless consider the appearances and disappearances of nodes or edges due to the dynamic nature of the network as (topological) failures. But even the computational models that take into account a certain degree of fault tolerance become insufficient for some very dynamic networks. Indeed, the classical models of fault tolerance either assume that the frequency of fault occurrences is small, which gives enough time to the algorithm to adapt to the changes, or that the system stabilizes after a certain amount of time (as in the self-stabilizing systems for example). Therefore, in the last decade or so, many more or less equivalent models have been developed that take into account the extreme dynamism of some communication networks. An interested reader will find in [4] a very complete overview of the different models and studies of dynamic graphs (see also [15] and [16]).

One of the first developed models, and also one of the most standard, is the model of evolving graphs [7]. To simplify, given a static graph G, called the underlying graph, an evolving graph based on G is a (possibly infinite) sequence of spanning but not necessarily connected subgraphs of G (see Section 2 for precise definitions). Differently speaking, the node set does not change but edges can appear or disappear at each time instant. This model is particularly well adapted for modeling dynamic synchronous networks.

In all its generality, the model of evolving graphs allows to consider an extremely varied set of dynamic networks. Therefore, to obtain interesting results, it is often required to make assumptions that reduce the possibilities of dynamic graphs generated by the model. One example is the assumption of connectivity over time, introduced in [7], which states that there is a journey (path over time) from any vertex to any other vertex. Another example is the assumption of constant connectivity, for which the graph must be connected at all times. This latter assumption, which is very usual, has been recently generalized in a paper by Kuhn, Lynch and Oshman [14] by the notion of T-interval-connectivity (see also [17] for other kinds of generalizations). Roughly speaking, given an integer $T \geq 1$, a dynamic graph is T-interval-connected if, for any window of T consecutive time steps, there exists a connected spanning subgraph which is stable throughout the period. (The notion of constant connectivity is thus equivalent to the notion of 1-interval-connectivity). This new notion, which captures the connection stability over time, allows the finding of interesting results: the T-interval-connectivity allows to reduce by a factor of about $\Theta(T)$ the number of messages that is necessary and sufficient to perform a complete exchange of information between all the vertices [14] (gossip problem).

In this paper, we carry on the study of these T-interval-connected dynamic graphs by considering the problem of exploration. A mobile entity (called agent), moving from node to node along the edges of a dynamic graph, must traverse/visit each of its

vertices at least once (the traversal of an edge takes one time unit).[1] This fundamental problem in distributed computing by mobile agents has been widely studied in static graphs since the seminal paper by Claude Shannon [19].

As far as highly dynamic graphs are concerned, only the case of periodically-varying graphs had been studied before the preliminary version of this paper [13], both in the absence [10, 12] and in the presence of harmful nodes [8, 9]. Since then, several works also considered the problem of exploring highly dynamic networks.

The most related papers are [11], a generalization of our paper to the case when the underlying graph is a tree of cycles (a cactus), and [6], a part of which is a generalization to the cases of the graphs with bounded treewidth, in general and for some specific subclasses, like the rings. Note that our results and those of [6] on the rings were proved independently. In [5], the authors study the impact that synchrony, anonymity and topological knowledge have on the computability and complexity of the deterministic multi-agent exploration with termination of the 1-interval-connected dynamic graphs based on the ring, in the case when the agents do not know the dynamics of the graph. Finally, [3] presents and proves the correctness of a self-stabilizing algorithm allowing three robots to perpetually explore a dynamic graph based on the ring in which each node can reach infinitely often any other node.

Besides, several papers focus on the complexity of computing the optimal exploration time of a dynamic graph given as (a centralized) input, in a similar manner as in the Traveling Salesman Problem for static graphs. In the dynamic case, the problem is called Temporal Graph Exploration Problem [6, 18] or Dynamic Map Visitation Problem [1, 2]. In [2], the case of several agents is considered, while [6, 18] and most of [1] consider the case of a single agent. In these papers, several polynomial-time algorithms are given, either exact algorithms for specific graph classes, or approximation algorithms for the general cases. In particular, [1] gives an $O(n^2)$ algorithm to compute the optimal exploration time of a given 1-interval-connected dynamic graph based on the n-node ring. Inapproximability results for the general case are given in [6, 18].

We focus here on the (worst-case) time complexity of this problem, namely the number of time units used by the agent to solve the problem in the T-interval-connected dynamic graphs. The problem of exploration, in addition to its theoretical interests, can be applied for instance to the network maintenance, where a mobile agent has to control the proper functioning of each vertex of the graph.

We consider the problem in two scenarios. In the first one, often referred as the offline scenario, the agent knows entirely and exactly the dynamic graph it has to explore. This situation corresponds to predictable dynamic networks such as transportation networks for example. In the second scenario, often referred as the online scenario, the agent does not know the dynamics of the graph, that is the times

[1] Note that several specializations of this problem exist, depending on whether the agent has to eventually detect termination (exploration with stop), return to its starting position (exploration with return), or even visit each vertex infinitely often (perpetual exploration). The rest of the paper just considers the general version of the problem.

of appearance and disappearance of the edges. This case typically corresponds to networks whose changes are related to frequent and unpredictable failures. In this second scenario, Kuhn, Lynch and Oshman [14] noted that the exploration problem is impossible to solve under the single assumption of 1-interval-connectivity. In fact, it is quite easy to convince oneself that by adding the assumption that each edge of the underlying graph must appear infinitely often, the exploration problem becomes possible, but the time complexity remains unbounded. In this article, and only for the second scenario, we therefore add the assumption of δ-recurrence, for some integer $\delta \geq 1$: each edge of the underlying graph appears at least once every δ time units.

It turns out that the problem of exploration is much more complex in dynamic graphs than in static graphs. Indeed, let us consider for example the first scenario (known dynamic graph). The worst-case exploration time of n-node static graphs is clearly in $\Theta(n)$ (worst case $2n-3$). On the other hand, the worst-case exploration time of n-node (1-interval-connected) dynamic graphs remains largely unknown. No lower bound better than the static bound is known, while the best known upper bound is quadratic, and directly follows from the fact that the temporal diameter of these graphs is bounded by n. Also, without the 1-interval-connectivity assumption, it is already NP-complete to decide whether exploration is feasible at all, while with this assumption the problem remains hard to approximate (see [6, 18]). In this paper, we focus on the study of T-interval-connected dynamic graphs whose underlying graph is a ring. Note that, in this particular case, the T-interval-connectivity property, for $T \geq 1$, implies that at most one edge can be absent at a given time.

Our results We determine in this paper the exact time complexity of the exploration problem for the n-node T-interval-connected dynamic graphs based on the ring, when the agent knows the dynamics of the graph. This is essentially $2n - T - 1$ time units (see Section 3 for details). When the agent does not know the dynamics of the graph, we add the assumption of δ-recurrence, and we show that the complexity is $n + \frac{n}{\max\{1,T-1\}}(\delta - 1) \pm \Theta(\delta)$ time units in this case (see Section 4 for details).

2 Model and Definitions

This section gives the precise definitions of the concepts and models informally mentioned in the introduction. Some definitions are similar or even identical to the definitions given in [14].

Definition 1 (Evolving graph) An *evolving graph* is a pair $\mathscr{G} = (V, \mathscr{E})$, where V is a static set of n vertices, and $\mathscr{E}$ is a function which maps to every integer $i \geq 0$ a set $\mathscr{E}(i)$ of undirected edges on V.

Definition 2 (Underlying graph) Given an evolving graph $\mathscr{G} = (V, \mathscr{E})$, the static graph $G = (V, \bigcup_{i=0}^{\infty} \mathscr{E}(i))$ is called the *underlying graph* of $\mathscr{G}$. Conversely, the evolving graph $\mathscr{G}$ is said to be *based* on the static graph G.

In this article, we consider the evolving graphs based on the n-node ring, denoted C_n. Since the cases $n = 1$ and $n = 2$ are trivial and somehow degenerated, we will assume $n \geq 3$.

Definition 3 (T-interval-connectivity) An evolving graph $\mathscr{G} = (V, \mathscr{E})$ is *T-interval-connected*, for an integer $T \geq 1$, if for every integer $i \geq 0$, the static graph $G_{[i,i+T[} = (V, \bigcap_{j=i}^{i+T-1} \mathscr{E}(j))$ is connected.

Definition 4 (δ-recurrence) An evolving graph is *δ-recurrent* if every edge of the underlying graph is present at least once every δ time steps.

Definition 5 (Temporal diameter) The *temporal diameter* of an evolving graph is the maximum time needed to go from any node to any other node starting at any time when at most one edge can be traversed at each time unit.

Note that the temporal diameter of any 1-interval-connected evolving graph is at most $n - 1$.

A mobile entity, called *agent*, operates on these dynamic graphs. We do not assume any limitation in terms of computational capabilities or memory. Nevertheless, the agent can traverse at most one edge per time unit. It may also stay at the current node (typically to wait for an incident edge to appear). We say that an agent *explores* the dynamic graph if and only if it visits all the nodes.

3 The Agent knows the Dynamics of the Graph

In this section, we assume that the agent perfectly knows the dynamic graph to be explored.

3.1 Upper Bound

The theorem presented in this subsection, Theorem 1, shows that the worst-case exploration time is actually small, bounded by $2n$, when the underlying graph is a ring. Furthermore, it shows that the agent can benefit from the T-interval-connectivity to spare an additive term T (cf. Fig. 1). Note that our upper bound is constructive, and tight (cf. Theorem 2).

The proof of Theorem 1 being quite long and technical, we first present a simpler and elegant proof, but giving a less precise upper bound, namely $2n - 2$ for any value of T. Nevertheless, we believe that this simplified result and its proof are of independent interest, because of the simplicity of the proof and the fact that the presented algorithm visits every node in the last $n - 1$ time units. Note that this implies that the agent is not changing direction and is never blocked during these $n - 1$ steps.

Proposition 1 *For every integers $n \geq 3$ and $T \geq 1$, and for every T-interval-connected dynamic graph based on C_n, there exists an agent (algorithm) exploring this*

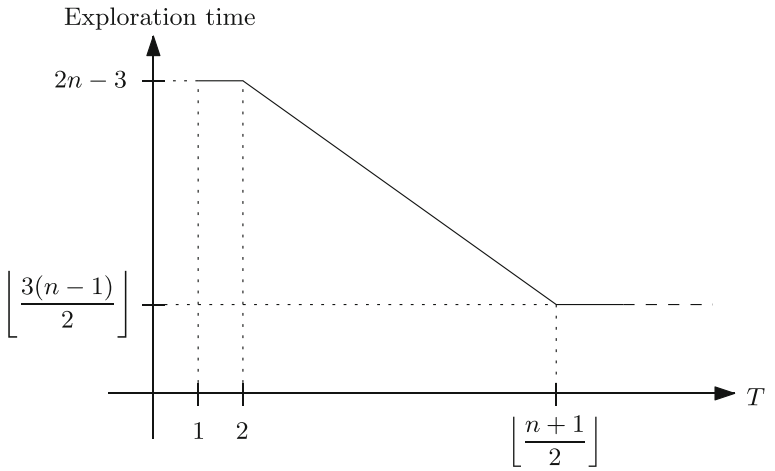

Fig. 1 Worst-case exploration time of the T-interval-connected dynamic graphs based on C_n as a function of T

dynamic graph in time at most $2n-2$ *such that this algorithm visits every node in the last* $n-1$ *time units.*

Proof We first prove that, for any time t, there exists a node $v(t)$ such that an agent starting from $v(t)$ at time t and moving in the clockwise direction is not blocked in the $n-1$ following time units, implying that such an agent has visited all nodes by time $t+n-1$. Indeed, consider that n virtual agents are placed on the n nodes of the graph at time t (one virtual agent on each node). Further consider that all these virtual agents go clockwise from time t on. At the first round (from time t), all virtual agents are trying to traverse different edges, so at most one virtual agent may be blocked. More generally, at each round, only one additional virtual agent can be blocked. Since there are n virtual agents, there exists at least one of them which is not blocked from time t to time $t+n-1$. Its starting node $v(t)$ satisfy the desired properties.

We can now describe the algorithm satisfying the statement of the proposition. Let u be the starting node of the agent and let v be the node $v(n-1)$ described in the first part of this proof. The algorithm simply consists in going from u to v in time at most $n-1$ (recall that the temporal diameter of any 1-interval-connected dynamic graph is at most $n-1$, so this is always possible), then in waiting at v until time $n-1$ (if v is reached sooner), and finally in going clockwise from v for $n-1$ time units. The proposition follows from the properties of the node v. □

Before proceeding with the formal theorem and its proof, let us informally describe the key ingredients of the proof of the most general case.

We consider two algorithms, being the algorithms always going in the clockwise, resp. counter-clockwise, direction, traversing edges as soon as the dynamic graph allows it. At the beginning of the process, the two agents executing these algorithms, starting from the same initial position, try to traverse distinct edges and thus, at each

time step, at least one of them progresses. During this phase, the average speed of the two agents is thus 1/2 (edge traversals per time unit). However, when the agents are about to meet each other on an edge or on a node (thus after time at most n), their progression may be stopped by the absence of a unique edge e.

If this edge e is absent for at least $n-1$ time steps, then any agent has enough time to change its direction and to explore all the nodes of the graph in the other direction, hence completing exploration within $2n$ steps, see Fig. 2.

If the edge e does not stay absent long enough and reappears at time t, we modify the two algorithms as follows. The agent previously progressing in the clockwise, resp. counter-clockwise, direction, starts now by exploring the ring in the opposite direction, before going back in the usual direction the latest possible so that it reaches the edge e at time at most t. At time t, the two modified algorithms cross each other, and then continue their progression in their usual direction until one of them terminates the exploration. Note that, after time t, we have again the property that, at each time step, at least one agent progresses. See Fig. 3.

Globally, except during the period when e is absent, the average speed of the two agents is 1/2. Besides, the modification of the algorithms generally allows each of the agents to explore an additional part of the ring. Unfortunately, these parts of the ring are traversed twice instead of once. Nevertheless, in the general case, the speed of both the modified agents is 1 during the period when e is absent. This compensates the loss induced by traversing twice some parts of the ring. Overall, the average speed is thus globally of at least 1/2, which implies that at least one of the two modified agents performs exploration within time $2n$.

In order to obtain a better upper bound thanks to the T-interval connectivity, we use the following observation.

Observation When a dynamic graph based on a ring is T-interval connected, all edges are present during $T-1$ steps between the removal of two different edges.

We use this observation to gain an additive term of $T-1$ on the exploration time, yielding to a time of roughly $2n-T$. A much more precise analysis of the modified algorithms allows us to obtain the exact claimed bounds.

Theorem 1 *For every integers $n \geq 3$ and $T \geq 1$, and for every T-interval-connected dynamic graph based on C_n, there exists an agent (algorithm) exploring this dynamic graph in time at most*

$$\begin{cases} 2n-3 & \text{if } T=1 \\ 2n-T-1 & \text{if } 2 \leq T \leq (n+1)/2 \\ \left\lfloor \frac{3(n-1)}{2} \right\rfloor & \text{if } T > (n+1)/2 \end{cases}$$

Proof Fix $n \geq 3$ and an arbitrary dynamic graph based on the ring C_n. Let $v_0, \cdots, v_{n-1}$ be the vertices of C_n in clockwise order. Assume that the agent starts exploration from v_0 at time 0. In order to prove this theorem, we will describe various algorithms, and we will show that at least one of them will allow the agent to perform exploration within the claimed time bound. Fix T to be any positive integer, and let $\mathscr{T}$ be this bound.

First assume that at most one edge e is absent during the time interval $[0, \mathscr{T})$. Then, an agent going to the closest extremity of e (in time at most $\lfloor (n-1)/2 \rfloor$) and then changing direction (for $n-1$ steps) will explore all nodes of the ring in time at most $\lfloor 3(n-1)/2 \rfloor \leq \mathscr{T}$ (see Fig. 1). So let us assume from now on that at least two different edges are absent at least once each during the time interval $[0, \mathscr{T})$.

Before proceeding with the rest of the proof, we introduce the following notations. Given a time interval I and two algorithms A and B, let d_A^I be the number of edge traversals performed by agent A during the time interval I, let α_A^I, resp. $\alpha_{A,B}^I$, be the number of time steps in I for which agent A, resp. both agents A and B, do(es) not move. (For all the algorithms we will consider, the reason why an agent will not move will always be the same: the edge it wants to traverse is absent.) Finally, let β^I be the number of time steps in I for which no edges are absent.

Let us now consider two simple algorithms. L, respectively R, is the algorithm always going in the counter-clockwise, resp. clockwise, direction, traversing edges as soon as the dynamic graph allows it. Now consider the sum of the number of edges traversed by each of the two algorithms until some time t. Since only one edge can be absent at a given time, this sum increases by at least one (and obviously by at most two) at each time step, until this sum is larger than or equal to $n-1$. So let e be the unique unexplored edge when this sum reaches $n-1$. If the sum jumps directly from $n-2$ to n, then fix e to be any of the last two unexplored edges. In both cases, let t_1 be the first time one of the two agents reaches one extremity of e. We consider two cases.

Case 1. The edge e is absent during the whole interval $[t_1, t_1 + n - 1)$.

In this case, the first agent to reach an extremity of e, at time t_1, goes back in the opposite direction and explores the ring in $n-1$ further steps. This gives an exploration time of at most $t_1 + n - 1$. Let $I_1 = [0, t_1)$. We have

$$t_1 = \begin{cases} d_L^{I_1} + \alpha_L^{I_1} & (1) \\ d_R^{I_1} + \alpha_R^{I_1} & (2) \end{cases}$$

and, since L and R are always trying to traverse distinct edges during I_1 and at most one edge may be removed at any time, we also have

$$\alpha_L^{I_1} + \alpha_R^{I_1} + \beta^{I_1} \leq t_1. \quad (3)$$

$$\text{Besides, by definition of } t_1, \text{ we have } d_L^{I_1} + d_R^{I_1} \leq n - 1. \quad (4)$$

Recall that we are considering the case when there are at least two removed different edges during the whole interval $[0, t_1 + n - 1)$. As mentioned before, when the dynamic graph is T-interval connected, all edges must be present during $T-1$ steps between the removal of two different edges. This implies that during the whole interval $[0, t_1 + n - 1)$, there are at least $T-1$ steps when no edges are absent. By definition of Case 1, these steps must occur before time t_1. Thus, we have

$$\beta^{I_1} \geq T - 1 \quad (5)$$

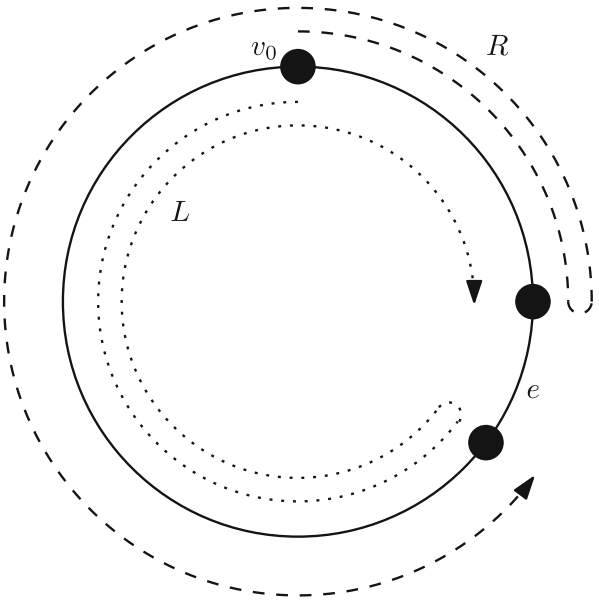

Fig. 2 In the case when edge e is absent for a long time (Case 1 in the proof), one of the dotted and dashed trajectories explores all nodes in the desired time

Summing the first five (in)equalities, we obtain

$$(1) + (2) + (3) + (4) + (5) \rightarrow t_1 + T \leq n,$$

or equivalently $t_1 + n - 1 \leq 2n - T - 1$, which gives the exact claimed bound for the general case $T \geq 2$. Figure 2 illustrates the trajectories analyzed in Case 1.

For $T = 1$, this bound is one unit larger than the claimed bound. So let us further study the case $T = 1$. If the inequality (4) is in fact strict, then the correct bound is obtained. Otherwise, it means that at time $t_1 - 1$, both agents were free to move. This implies that either $\beta^{t_1} \geq 1$ (the inequality (5) is strict) or that the inequality (3) is strict. In both cases, this also gives the correct bound.

Case 2. The edge e is not absent during the whole interval $[t_1, t_1 + n - 1)$.

Then let t_2 be the time such that $t_2 + 1$ is the first time at which every edge has been explored by L or R (or both). Note that this definition implies that $t_2 \geq t_1$. We now define two new algorithms, one of which will explore the dynamic graph within time $\mathscr{T}$.

Let L' be the algorithm that is equal to L until some time t, at which L' goes back in the other direction forever. More precisely, L' is the algorithm for which t is the largest possible value such that L' and R share the same position at time t_2 (intuitively, L' just has time to catch back R at time at most t_2). Similarly, let R' be the algorithm that is equal to R until some time t, at which R' goes back in the other direction forever. More precisely, R' is the algorithm for which t is the largest possible value such that R' and L share the same position at time t_2.

In order to analyze the algorithms L' and R', we introduce two other algorithms. Let L'', respectively R'' be the algorithm defined as L', resp. R', but turning back exactly one time unit later than L', resp. R'.

Finally, let $\mathscr{T}_{exp}$ be the exploration time of the first between L' and R' exploring the dynamic graph, and let $I_1 = [0, t_1)$, $I_2 = [t_1, t_2)$, $I_{1,2} = [0, t_2)$, $I_3 = [t_2, \mathscr{T}_{exp})$, and $I = [0, \mathscr{T}_{exp})$.

As in the first case, we have

$$t_1 = \begin{cases} d_L^{I_1} + \alpha_L^{I_1} & (6) \\ d_R^{I_1} + \alpha_R^{I_1} & (7) \end{cases}$$

On I_1, we have

$$\alpha_{L''}^{I_1} + \alpha_{R''}^{I_1} - \alpha_{L'',R''}^{I_1} + \beta^{I_1} \leq t_1. \quad (8)$$

Besides, L and R are always trying to traverse distinct edges during I_1. Moreover, by definition, the algorithm L'', resp. R'', does not catch R, resp. L, before time t_2 (and thus t_1). This gives

$$\alpha_L^{I_1} + \alpha_R^{I_1} + \alpha_{L'',R''}^{I_1} + \beta^{I_1} \leq t_1 \quad (9)$$

$$(6) + (7) + (8) + (9) \rightarrow \alpha_{L''}^{I_1} + \alpha_{R''}^{I_1} + 2\beta^{I_1} \leq d_L^{I_1} + d_R^{I_1} \quad (10)$$

On $I_{1,2}$, we have

$$t_2 = \begin{cases} d_{L''}^{I_{1,2}} + \alpha_{L''}^{I_{1,2}} & (11) \\ d_{R''}^{I_{1,2}} + \alpha_{R''}^{I_{1,2}} & (12) \end{cases}$$

Note that, by definition of L'' and R''

$$d_{L''}^{I_{1,2}} \leq d_{L'}^{I_{1,2}} + 1 \quad (13)$$

$$d_{R''}^{I_{1,2}} \leq d_{R'}^{I_{1,2}} + 1 \quad (14)$$

$$(11) + (12) + (13) + (14) \rightarrow 2t_2 \leq d_{L'}^{I_{1,2}} + d_{R'}^{I_{1,2}} + \alpha_{L''}^{I_{1,2}} + \alpha_{R''}^{I_{1,2}} + 2 \quad (15)$$

Note that, by definition of t_1 and t_2, the edge e is absent during the whole interval I_2. Besides, neither L' nor R' reaches an extremity of the edge e before turning back, because otherwise it would reach the other extremity too late, namely at time at least $t_1 + n - 1$, which is larger than t_2 by definition of Case 2. (By the way, this proves that $\mathscr{T}_{exp} > t_2$.) This implies that L'' and/or R'' may reach an extremity of e (at time t_1) but in this case turn back immediately before trying to traverse it. Moreover, L'' and R'' cannot reach an extremity of edge e while going clockwise, resp. counter-clockwise, before time t_2. This means that they are never blocked during the time interval I_2. This translates into

$$\alpha_{L''}^{I_{1,2}} = \alpha_{L''}^{I_1} \quad (16)$$

$$\alpha_{R''}^{I_{1,2}} = \alpha_{R''}^{I_1} \quad (17)$$

$$(10) + (15) + (16) + (17) \rightarrow 2t_2 + 2\beta^{I_1} \leq d_L^{I_1} + d_R^{I_1} + d_{L'}^{I_{1,2}} + d_{R'}^{I_{1,2}} + 2 \quad (18)$$

Starting from time $t_2 + 1$, the algorithms L' and R' are always trying to traverse distinct edges. Since L' and R' are not blocked at time t_2, this means that, on I_3, we have

$$\alpha_{L'}^{I_3} + \alpha_{R'}^{I_3} + \beta^{I_3} \leq \mathscr{T}_{exp} - t_2 \tag{19}$$

and

$$\mathscr{T}_{exp} - t_2 = \begin{cases} d_{L'}^{I_3} + \alpha_{L'}^{I_3} & (20) \\ d_{R'}^{I_3} + \alpha_{R'}^{I_3} & (21) \end{cases}$$

$$(19) + (20) + (21) \rightarrow \mathscr{T}_{exp} - t_2 + \beta^{I_3} \leq d_{L'}^{I_3} + d_{R'}^{I_3} \tag{22}$$

$$(22) + \frac{1}{2}(18) \rightarrow$$

$$\mathscr{T}_{exp} + \beta^{I_1} + \beta^{I_3} \leq \frac{1}{2}(d_L^{I_1} + d_R^{I_1} + d_{L'}^{I_{1,2}} + d_{R'}^{I_{1,2}}) + d_{L'}^{I_3} + d_{R'}^{I_3} + 1 \tag{23}$$

Let x, resp. y, be the number of edges traversed by L', resp. R', before turning back. Then

$$d_{L'}^{I_{1,2}} = 2x + d_R^{I_{1,2}} \tag{24}$$

$$d_{R'}^{I_{1,2}} = 2y + d_L^{I_{1,2}} \tag{25}$$

Counting the number of edges that still need to be traversed by L' and R' until exploration is performed, we obtain

$$d_{L'}^{I_3} \leq n - 1 - x - d_R^{I_{1,2}} \tag{26}$$

$$d_{R'}^{I_3} \leq n - 1 - y - d_L^{I_{1,2}} \tag{27}$$

Similarly as in Case 1, because of the T-interval connectivity and the hypotheses that at least two different edges are removed, we have

$$\beta^I \geq T - 1. \tag{28}$$

Besides, note that $\beta^{I_1} + \beta^{I_3} = \beta^I$.
Finally, since $d_L^{I_1} - d_L^{I_{1,2}}$ and $d_R^{I_1} - d_R^{I_{1,2}}$ are less than or equal to 0, we get

$$(23) + \frac{1}{2}(24) + \frac{1}{2}(25) + (26) + (27) + (28) \rightarrow \mathscr{T}_{exp} \leq 2n - T.$$

In fact, we claim that this last inequality is strict. We will prove this claim by contradiction, assuming that inequalities (19), (26), (27), and (28) are in fact equalities. The equalities for (26) and (27) imply that the algorithms L' and R' are not blocked during the last step of I_3, which allow them to simultaneously terminate exploration at this last step. Equation (19) being an equality, this implies that this last step is counted in β^{I_3}. However, this step where no edges are absent being the last one of I, it cannot belong to the $T - 1$ consecutive steps with no absent edges that occur between the removal of two different edges in the same interval I.

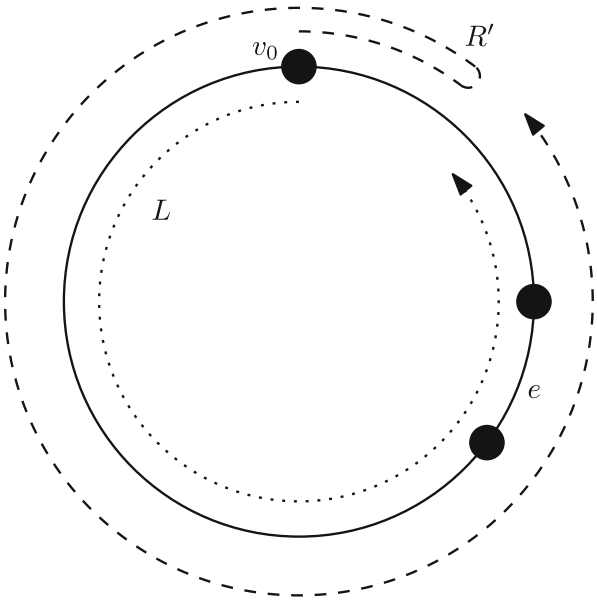

Fig. 3 In the case when edge e is absent for a short time (Case 2 in the proof), the dashed trajectory is used (or its equivalent in the other direction). The turning time of the dashed trajectory is defined at the latest possible time such that both the dashed and the dotted trajectories traverse edge e at the same time

This contradicts the fact that (28) is an equality, concluding the proof for $T \geq 2$. Figure 3 illustrates the trajectories analyzed in Case 2.

For $T = 1$, the bound obtained so far is one unit larger than the claimed bound. For the purpose of contradiction, assume that $\mathscr{T}_{exp} = 2n - 2$. This implies that all inequalities are in fact equalities except exactly one of the inequalities (19), (26), (27), and (28). In the latter case, we proved more precisely that $\beta^I = 1$ because $\beta^{I_3} = 1$. This implies that $\beta^{I_1} = 0$ in all four cases. We will now come to a contradiction by proving that one of the inequalities (8), (9), (13), or (14) must be strict.

The only way for Eq. (13), resp. (14), to be an equality is that L'', resp. R'', traverses an edge just before turning back, that is at the step, say $t_{L'}$, resp. $t_{R'}$, when L', resp. R', turns back. Differently speaking, Eq. (13), resp. (14), being an equality implies that L'' and thus L, resp. R'' and thus R, are not blocked at step $t_{L'}$, resp. $t_{R'}$. Since (8) is assumed to be an equality, and because $\beta^{I_1} = 0$, at least one of L'' and R'' is blocked at each time step of the interval I_1. This is in particular true for the times $t_{L'}$ and $t_{R'}$. Therefore, the two times $t_{L'}$ and $t_{R'}$ must be different. Without loss of generality, assume that $t_{L'} < t_{R'}$. Let us now consider the step at time $t_{R'}$. At this step, R'' and thus R as well are not blocked. This implies that L must be blocked at this step because of (9) being an equality. However, L'' has already turned back and does not travel with L anymore, and thus cannot be blocked at this step. This would lead to (8) being strict, the contradiction concluding this proof. □

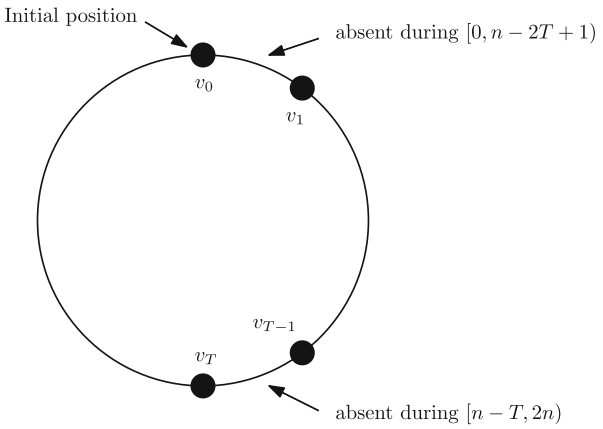

Fig. 4 T-interval-connected dynamic graph based on C_n achieving the worst-case exploration time, for $2 \leq T \leq \lceil (n+1)/2 \rceil$

3.2 Lower Bound

We now prove that the precise bound given in Section 3.1 is actually the exact worst-case time complexity of the exploration problem.

Theorem 2 *For every integers $n \geq 3$ and $T \geq 1$, there exists a T-interval-connected dynamic graph based on C_n such that any agent (algorithm) needs at least*

$$\begin{cases} 2n-3 & \text{if } T = 1 \\ 2n-T-1 & \text{if } 2 \leq T \leq (n+1)/2 \\ \left\lfloor \frac{3(n-1)}{2} \right\rfloor & \text{if } T > (n+1)/2 \end{cases}$$

time units to explore it.

Proof For any integers $n \geq 3$, and $2 \leq T \leq \lceil (n+1)/2 \rceil$, we define a T-interval-connected dynamic graph $\mathscr{G}_{n,T}$ based on C_n. Let $v_0, v_1, \cdots, v_{n-1}$ be the vertices of C_n in clockwise order. Assume that the exploration starts from v_0 at time 0. In $\mathscr{G}_{n,T}$, the edge $\{v_0, v_1\}$, respectively $\{v_{T-1}, v_T\}$, is absent in the time interval $[0, n-2T+1)$, respectively $[n-T, 2n)$. See Fig. 4. Note that this dynamic graph is indeed T-interval-connected.

Consider any agent (algorithm). We will now prove that the time it uses to explore $\mathscr{G}_{n,T}$ is at least $2n-T-1$. Since the agent must explore all vertices, it must in particular explore both v_{T-1} and v_T. We consider two cases.

Case 1. v_{T-1} is explored before v_T.

To visit v_{T-1} without going through v_T, the agent must traverse the edge $\{v_0, v_1\}$. By construction, this edge is absent until time $n-2T+1$. Moreover, the length of

the path between v_0 and v_{T-1} without going through v_T is $T-1$. Thus the agent needs at least $n-T$ time units to reach v_{T-1} for the first time. Since the edge $\{v_{T-1}, v_T\}$ is absent in the time interval $[n-T, 2n)$, the fastest way of reaching v_T is to traverse the whole ring through v_0, inducing $n-1$ additional time units. So in this first case, the agent needs at least $2n-T-1$ time units to explore $\mathcal{G}_{n,T}$.

Case 2. v_T is explored before v_{T-1}.

To visit v_T without going through v_{T-1}, the agent must use the path v_0, v_{n-1}, up to v_T, which is of length $n-T$. When at node v_T, and since the edge $\{v_{T-1}, v_T\}$ is absent in the time interval $[n-T, 2n)$, the fastest way of reaching v_{T-1} is to traverse the whole ring through v_0, inducing $n-1$ additional time units. Thus also in the second case, the agent needs at least $2n-T-1$ time units to explore $\mathcal{G}_{n,T}$.

This proves the theorem for values of T in $[2, \lceil (n+1)/2 \rceil]$. In fact, this also proves the theorem for $T=1$ because $\mathcal{G}_{n,2}$ is obviously also 1-interval-connected, and thus the bound 2n-3 proved for $T=2$ is also valid for $T=1$. Besides, note that only one edge is ever removed in $\mathcal{G}_{n,\lceil (n+1)/2 \rceil}$. This dynamic graph is therefore T-interval-connected for any T, and thus the theorem is also proved for values of T larger than $(n+1)/2$. □

4 The Agent does not know the Dynamics of the Graph

In this section, we assume that the agent does not know the dynamics of the graph, i.e., it does not know the times of appearance and disappearance of the edges. As explained in the introduction, we assume here the δ-recurrence property, for a given $\delta \geq 1$, in order for the problem to be solvable in bounded time.

4.1 Upper Bound

We first prove that there exists a very simple algorithm that is able to explore all the δ-recurrent T-interval-connected dynamic graphs based on the ring. This algorithm consists in moving as much and as soon as possible in a fixed arbitrary direction, see Algorithm 1.

Algorithm 1 STUBBORN-TRAVERSAL(`dir`)

```
Require: a direction dir
    for each time step do
        if the edge in the dir direction is present then
            traverse it
        else
            wait
        end if
    end for
```

Theorem 3 *For every integers $n \geq 3$, $T \geq 1$ and $\delta \geq 1$, and for any direction* `dir`*, Algorithm* STUBBORN-TRAVERSAL(`dir`) *explores any δ-recurrent T-interval-connected dynamic graph based on C_n in time at most*

$$n - 1 + \left\lceil \frac{n-1}{\max\{1, T-1\}} \right\rceil (\delta - 1).$$

Proof Fix an arbitrary direction `dir` and let us analyze the algorithm STUBBORN-TRAVERSAL(`dir`). Note first that it will complete exploration after traversing exactly $n - 1$ edges. To bound its exploration time, it thus remains to bound the number of time steps when the agent cannot move.

Since the dynamic graph is δ-recurrent, an edge cannot be absent for more than $\delta - 1$ consecutive time steps. Furthermore, since the dynamic graph is T-interval-connected, two time steps in which two different edges are absent must be separated by at least $T - 1$ time steps in which all edges are present. Therefore, the agent can traverse at least $\max\{1, T - 1\}$ edges between two consecutive blocks at different nodes. To summarize, the agent can be blocked at most $\left\lceil \frac{n-1}{\max\{1,T-1\}} \right\rceil$ times during at most $\delta - 1$ time steps.

Putting everything together, the agent will perform edge traversals for $n - 1$ time steps and will wait for at most $\left\lceil \frac{n-1}{\max\{1,T-1\}} \right\rceil (\delta - 1)$ time steps, which gives the claimed bound. □

4.2 Lower Bound

It turns out that the simple and natural Algorithm 1, described and analyzed in Section 4.1, is almost optimal, up to an additive term proportional to δ.

Theorem 4 *For every integers $n \geq 3$, $T \geq 1$, and $\delta \geq 1$, and for every agent (algorithm), there exists a δ-recurrent T-interval-connected dynamic graph based on C_n such that this agent needs at least*

$$n - 1 + \left\lfloor \frac{n-3}{\max\{1, T-1\}} \right\rfloor (\delta - 1)$$

time units to explore it.

This result holds even if the agent knows n, T and δ.

Proof Let $n \geq 3$, $T \geq 1$, and $\delta \geq 1$. Fix an arbitrary agent (algorithm) A. We construct as follows the δ-recurrent T-interval-connected dynamic graph $\mathcal{G}_{n,T,\delta}(A)$ based on C_n that this agent will fail to explore in less than the claimed bound.

Let $v_0, v_1, \cdots, v_{n-1}$ be the vertices of C_n in clockwise order. Assume that the agent starts exploration from v_0 at time 0. For any integer $1 \leq i \leq n - 1$, if the node v_i is explored by going from v_0 in the counter-clockwise direction, then node v_i is denoted v_{i-n}. Finally, let $\tilde{T} = \max\{1, T - 1\}$.

In the dynamic graph $\mathcal{G}_{n,T,\delta}(A)$, only the edges $\{v_{\tilde{T}+1}, v_{\tilde{T}+2}\}$, $\{v_{2\tilde{T}+1}, v_{2\tilde{T}+2}\}$, and so on, and $\{v_0, v_{-1}\}$, $\{v_{-\tilde{T}}, v_{-\tilde{T}-1}\}$, $\{v_{-2\tilde{T}}, v_{-2\tilde{T}-1}\}$, and so on, may be absent. The actual times of appearance and disappearance of these edges depend on the algorithm A. For any integer $i \geq 0$, each time the agent arrives at node $v_{-i\tilde{T}}$ in the counter-clockwise direction, the edge $\{v_{-i\tilde{T}}, v_{-i\tilde{T}-1}\}$ is removed until either the δ-recurrence forces the edge to reappear or the agent leaves the node $v_{-i\tilde{T}}$ to go on $v_{-i\tilde{T}+1}$. Similarly, for any integer $i \geq 1$, each time the agent arrives at node $v_{i\tilde{T}+1}$ in the clockwise direction, the edge $\{v_{i\tilde{T}+1}, v_{i\tilde{T}+2}\}$ is removed until either the δ-recurrence forces the edge to reappear or the agent leaves the node $v_{i\tilde{T}+1}$ to go on $v_{i\tilde{T}}$. Note that between two time steps with two different absent edges, there are at least $T-1$ time steps for which no edges are absent. The dynamic graph is therefore T-interval-connected. It is also δ-recurrent by construction.

By definition of the dynamics of the graph, the agent needs to wait $\delta-1$ time units to go from $v_{-i\tilde{T}}$ to $v_{-i\tilde{T}-1}$, for $i \geq 0$, or to go from $v_{i\tilde{T}+1}$ to $v_{i\tilde{T}+2}$, for $i \geq 1$. Also, except near the origin in the clockwise direction, the agent cannot traverse more than $\tilde{T}$ new edges before having to traverse such a blocking edge. Hence, to explore all the vertices, the agent needs to perform at least $\left\lfloor \frac{n-3}{\tilde{T}} \right\rfloor$ such traversals. This lower bound is obtained in the case when the agent explores the ring by always going clockwise (starting in the counter-clockwise direction and/or changing direction during the exploration do not help). The waiting time of the agent is thus at least $\left\lfloor \frac{n-3}{\tilde{T}} \right\rfloor (\delta-1)$. Since the agent needs also at least $n-1$ time units to traverse enough edges so that all vertices are explored, we obtain the claimed bound. □

5 Conclusion

We studied in this paper the problem of exploration of the T-interval-connected dynamic graphs based on the ring in two scenarios, when the agent is specific to the dynamic graph, and when the agent does not know the dynamics of the graph. The next objective is obviously to extend these results to larger families of underlying graphs. Unfortunately, this problem is much more difficult than it seems: proving that any dynamic graph based on a tree of cycles (a cactus) can be explored in time $O(n)$ is already a challenging open problem.

References

1. Aaron, E., Krizanc, D., Meyerson, E.: DMVP: Foremost Waypoint Coverage of Time-Varying Graphs. In: 40th International Workshop on Graph-Theoretic Concepts in Computer Science (WG), LNCS 8147, pp. 29–41 (2014)
2. Aaron, E., Krizanc, D., Meyerson, E.: Multi-Robot Foremost Coverage of Time-Varying Graphs. In: 10th International Symposium on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics (ALGOSENSORS), LNCS 8847, pp. 22–38 (2014)
3. Bournat, M., Datta, A.K., Dubois, S.: Self-Stabilizing Robots in Highly Dynamic Environments. In: 18th International Symposium on Stabilization, Safety, and security of distributed systems (SSS 2016), LNCS 10083, pp. 54–69 (2016)

4. Casteigts, A., Flocchini, P., Quattrociocchi, W., Santoro, N.: Time-varying graphs and dynamic networks. Int. J. Parallel Emergent Distrib. Syst., 27(5) (2012)
5. Di Luna, G.A., Dobrev, S., Flocchini, P., Santoro, N.: Live exploration of dynamic rings. In: IEEE 36th International Conference on Distributed Computing Systems (ICDCS), pp. 570–579 (2016)
6. Erlebach, T., Hoffmann, M., Kammer, F.: On Temporal Graph Exploration. In: 42nd International Colloquium on Automata, Languages, and Programming (ICALP), LNCS 9134, pp. 444–455 (2015)
7. Ferreira, A.: Building a reference combinatorial model for MANETs. Network, IEEE **18**(5), 24–29 (2004)
8. Flocchini, P., Kellett, M., Mason, P.C., Santoro, N.: Searching for black holes in subways. Theory of Computing Systems **50**(1), 158–184 (2012)
9. Flocchini, P., Kellett, M., Mason, P.C., Santoro, N.: Finding Good Coffee in Paris. In: 6th International Conference on Fun with Algorithms (FUN), LNCS 7288, pp. 154–165 (2012)
10. Flocchini, P., Mans, B., Santoro, N.: On the exploration of time-varying networks. Theor. Comput. Sci. **469**, 53–68 (2013)
11. Ilcinkas, D., Klasing, R., Wade, A.M.: Exploration of Constantly Connected Dynamic Graphs Based on Cactuses. In: 21st International Colloquium on Structural Information and Communication Complexity (SIROCCO), LNCS 8576, pp. 250–262 (2014)
12. Ilcinkas, D., Wade, A.M.: On the Power of Waiting when Exploring Public Transportation Systems. In: 15th International Conference On Principles Of Distributed Systems (OPODIS), LNCS 7109, pp. 451–464 (2011)
13. Ilcinkas, D., Wade, A.M.: Exploration of the T-Interval-Connected Dynamic Graphs: the Case of the Ring. In: 20th International Colloquium on Structural Information and Communication Complexity (SIROCCO), LNCS 8179, pp. 13–23 (2013)
14. Kuhn, F., Lynch, N.A., Oshman, R.: Distributed computation in dynamic networks. In: 42nd ACM symposium on Theory of computing (STOC), pp. 513–522 (2010)
15. Kuhn, F., Oshman, R.: Dynamic networks: models and algorithms. ACM SIGACT News **42**(1), 82–96 (2011)
16. Michail, O.: An introduction to temporal graphs: an algorithmic perspective. Internet Math. **12**(4), 239–280 (2016)
17. Michail, O., Chatzigiannakis, I., Spirakis, P.G.: Causality, influence, and computation in possibly disconnected synchronous dynamic networks. J. Parallel Distrib. Comput. **74**(1), 2016–2026 (2014)
18. Michail, O., Spirakis, P.G.: Traveling salesman problems in temporal graphs. Theor. Comput. Sci. **634**, 1–23 (2016)
19. Shannon, C.E.: Presentation of a maze-solving machine. 8th Conf. of the Josiah Macy Jr. Found. (Cybernetics), 173–180 (1951)