Theory of Computing Systems (2021) 65:257–273
https://doi.org/10.1007/s00224-020-10001-0

Exploration of Dynamic Cactuses with Sub-logarithmic Overhead

David Ilcinkas[1] · Ahmed M. Wade[2]

Published online: 14 August 2020

Abstract
We study the problem of exploration by a mobile entity (agent) of a class of dynamic networks, namely constantly connected dynamic graphs. This problem has already been studied in the case where the agent knows the dynamics of the graph and the underlying graph is a ring of n vertices (Ilcinkas and Wade 2018). In this paper, we consider the same problem and we suppose that the underlying graph is a cactus graph (a connected graph in which any two simple cycles have at most one vertex in common). We propose an algorithm that allows the agent to explore these dynamic graphs in at most $O(n\frac{\log n}{\log\log n})$ time units. We show that the lower bound of the algorithm is $\Omega(n\frac{\log n}{(\log\log n)^2})$ time units (for infinitely many n).

Keywords Exploration · Dynamic graphs · Mobile agent · Connectivity over time

1 Introduction

The exploration of a graph by a (physical or software) mobile agent consists of visiting at least once each of the vertices of the graph, starting from a given vertex of the

A (weaker) preliminary version of this paper appeared in the Proceedings of the 21st International Colloquium on Structural Information and Communication Complexity (SIROCCO 2014) [18].

D. Ilcinkas is partially supported by the ANR projects DESCARTES (ANR-16-CE40-0023) and EState (ANR-16-CE25-0009), and the "Investments for the future" Programme IdEx Bordeaux – CPU (ANR-10-IDEX-03-02).
A. M. Wade is partially supported by the African Center of Excellence in Mathematics, Computer Science and ICT (CEA-MITIC).

✉ Ahmed M. Wade
awade@ept.sn

David Ilcinkas
david.ilcinkas@labri.fr

1 LaBRI, CNRS, University of Bordeaux, Bordeaux, France

2 École Polytechnique de Thiès, Thies, Senegal

graph. In practice, many concrete systems can be modeled by graphs. This is what makes the use of graphs very versatile. For example, graphs can be used to model pipeline systems, underground tunnels, roads networks, etc. In this case, the exploration is performed by a mobile robot. Graphs can also be used to model more abstract environments such as computer networks. In this case, the mobile entities used to explore these environments are software agents, that is to say a program running in these environments.

This fundamental problem in distributed computing by mobile agents has been extensively studied since the seminal paper by Claude Shannon [25]. However, the majority of the work concerns static graphs, while new generations of interconnected environments tend to be extremely dynamic. To take into account the dynamism of these extreme environments, for a decade, researchers have begun to model these dynamic environments with dynamic graphs. Several models have been developed. The interested reader may find in [7] a comprehensive overview of the different models and studies of dynamic graphs (see also [21, 22]).

One of the first developed models, and also one of the most classical, is the model of evolving graphs [14]. For simplicity, given a static graph G, called underlying graph, an evolving graph $\mathcal{G}$ based on G is a (possibly infinite) sequence of (spanning but not necessarily connected) subgraphs of G (see Section 2 for the precise definitions). This model is particularly suited for modeling *synchronous* dynamic networks.

In this paper, we study the problem of exploration of dynamic graphs considering the model of constantly connected evolving graphs. An evolving graph $\mathcal{G} = (G_1, G_2, \ldots)$ is called *constantly connected* if each graph G_i which composes it is connected. This class of graphs was used in [24] to study the problem of information dissemination. In 2010, Kuhn, Lynch and Oshman [20] generalize this class of dynamic graphs by introducing the notion of T-interval-connectivity. Roughly speaking, given an integer $T \geq 1$, a dynamic graph is T-interval-connected if for any window of T time units, there is a connected spanning subgraph that is stable throughout the period. (The notion of constant connectivity is equivalent to the notion of 1-interval-connectivity.) This new concept, which captures the connection stability over time, allows to derive interesting results: the T-interval-connectivity allows a savings of a factor about $\Theta(T)$ on the number of messages necessary and sufficient to achieve a complete exchange of information between all vertices [11, 20].

During these last few years, several studies consider constantly connected dynamic graphs where the underlying graph of the dynamic graph is a ring of n vertices. The problem of exploration with termination by a mobile agent is considered in [9, 17, 19]. If the dynamics of the graph is known, [19] shows that a single agent can solve the problem, and $2n - 3$ time units are necessary and sufficient. If the dynamics is not known in advance, [9] shows that two agents knowing an upper bound N on the number of vertices can solve the problem, and $(3N - 6)$ time units are sufficient if all agents are active at each time step, and $O(N^2)$ moves are sufficient if a subset of the agents might be active at each time step. The case when the agent has partial information about network changes is considered in [17]. More precisely, the authors study the exploration time for a single agent which knows the dynamics of the graph for the next S steps in its H-hop neighborhood, for given parameters S and H.

The problem of perpetual exploration is considered in [5, 15]. In [5], the authors consider that all agents are active at each time step and show that to solve the problem, one agent is sufficient in the rings of size two[1], two agents are sufficient in the rings of size three, and three agents are sufficient for all other rings. In [15] the authors consider time varying graphs whose topology is arbitrary and unknown to the agents and investigates the number of agents that are necessary and sufficient to explore such graphs. In addition to the problem of exploration, the problem of dispersion of a team of agents [3], gathering [10] and patrolling by a team of agents [8] are studied, considering constantly connected dynamic graphs based on the ring.

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 [4, 12, 23] or Dynamic Map Visitation Problem [1, 2]. In [2], the case of several agents is considered, while [4, 12, 23] and most of [1] consider the case of a single agent. The problem is shown to be NP-complete, even when the underlying graph has pathwidth 2 and at each time step, the current graph is connected [4]. In the other 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-vertex ring. Inapproximability results for the general case are given in [12, 23].

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 scenario where the dynamic graph is known. The worst-case exploration time of n-vertex static graphs is clearly in $\Theta(n)$ (worst case $2n - 3$). On the other hand, the worst-case exploration time of n-vertex (1-interval-connected) dynamic graphs remains largely unknown. In [12] the authors give a worst-case lower bound in $\Omega(n^2)$ for general graphs and $\Omega(n \log n)$ for degree-bounded graphs. An upper bound in $O(d \log d \cdot \frac{n^2}{\log n})$ for degree-bounded graphs is given in [13].

The goal of this paper is to extend the results obtained in [19] to larger families of underlying graphs. Unfortunately, the problem turns out to be much more difficult than it seems (see [16] for a variant of the problem in dynamic tori). We will see that proving that any dynamic graph based on a tree of cycles (a cactus) can be explored in time $O(n)$ is already a challenging problem.

Our results. We will first give two exploration methods that are efficient for exploring a very large set of constantly connected dynamic graphs based on a cactus, when the agent knows the dynamics of the graph. We will then combine these two exploration methods. We show that the combination of the two methods yields an algorithm that explores all constantly connected dynamic graphs based on a cactus of n vertices in $O(n\frac{\log n}{\log\log n})$ time units, and we derive a lower bound of $\Omega(n\frac{\log n}{(\log\log n)^2})$ time units for the algorithm (for infinitely many n).

[1]In [5], the authors define a ring of size two as a two-node path if the graph is simple, or as two nodes linked by two bidirectional edges otherwise.

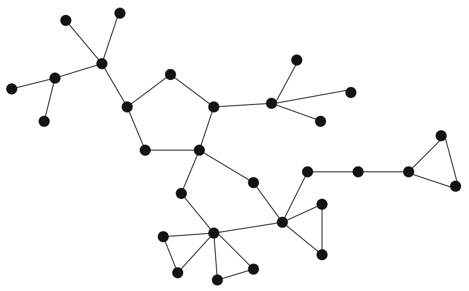
Fig. 1 Example of a cactus

2 Preliminaries

This section provides precise definitions of the concepts and models discussed informally earlier. We also give some previous results from the literature on the problem studied in this paper.

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

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

In this paper, we consider dynamic graphs based on a cactus of size n. We also assume that the agent knows the dynamics of the graph, that is to say, the times of appearance and disappearance of the edges of the dynamic graph.

Definition 3 (Constant connectivity) A dynamic graph is called *constantly connected* if, for any integer i, the static graph $G_i = (V, \mathcal{E}(i))$ is connected.

Definition 4 (Cactus) A *cactus* is a simple graph $G = (V, E)$ in which two connected cycles have at most one vertex in common (see Fig. 1).

A mobile entity, called *agent*, operates on these dynamic graphs. The agent can traverse at most one edge per time unit. It may also stay at the current vertex (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 vertices.

In this article, we will use the following results from the literature.

Theorem 1 *Ilcinkas and Wade* [19] *For every integers $n \geq 3$ and $t \geq 0$, and for every constantly connected dynamic graph based on a ring with n vertices, there exists a vertex $v^{(t)}$ such that an agent starting at time t on $v^{(t)}$ and going in the clockwise*[2]

[2]The actual definition of the "clockwise" direction does not really matter as long as it is any fixed direction.

direction for $n-1$ time units will never be blocked by a missing edge, and thus will explore all vertices within those $n-1$ time units.

Sketch of proof Consider n virtual agents placed on the n vertices (one agent on each vertex). Make all agents move in the clockwise direction for $n-1$ time units from time t. Since at most one edge is removed at a time, it holds that, at each time, at most one such virtual agent is blocked at this time without having been blocked before. Thus, one of the n virtual agents is never blocked during the $n-1$ time units, and the starting vertex of this agent is the vertex $v^{(t)}$ we are looking for. □

Theorem 2 *Kuhn et al.* [20] *For every constantly connected dynamic graph on n vertices, at most $n-1$ time units are sufficient for an agent to go from any vertex to any other vertex in the graph, when the agent knows the dynamics of the graph.*

Sketch of proof Let u be some arbitrary vertex of the dynamic graph. For any integer $i \geq 0$, let V_i be the set of vertices reachable from u in at most i time units. We have that $V_i \subsetneq V_{i+1}$ until V_i contains all the vertices. Indeed, before all vertices are reachable, there exists a vertex not in V_i which is neighbor of a vertex in V_i, because the dynamic graph is constantly connected. □

Theorem 3 *Ilcinkas and Wade* [19] *For every integer $n \geq 3$ and for every constantly connected dynamic graph based on a ring with n vertices, there exists an agent (algorithm),* Explore-ring, *exploring this dynamic graph in time at most $2n-2$ time units. (The agent knows the dynamics of the graph).*

Sketch of proof The algorithm proceeds as follows. Go to vertex $v^{(n-1)}$, whose existence is guaranteed by Theorem 1. This can be done in at most $n-1$ time units, by Theorem 2. At time $n-1$, go clockwise during $n-1$ time units to fully explore the ring, thanks to the properties of $v^{(n-1)}$. □

In this paper, we will only consider asymptotic exploration times. Since exploration of an n-vertex dynamic graph requires at least $n-1$ edge traversals, Theorem 2 implies that requiring the agent to return to the starting vertex can at most double the exploration time. Therefore we will in fact study in this paper the exploration with return problem.

To give a simpler analysis of our algorithms, we consider the tree representation of a cactus given in [6]. For any given cactus, the set of all vertices V is partitioned into three subsets of vertices. Call *C-vertices* the vertices of degree 2 that belong to one and only one cycle, *G-vertices* the vertices that do not belong to any cycle, and H-vertices the other vertices (which belong to at least one cycle and have a degree at least 3), which we also call *attachment vertices*.

A *subtree* is a connected set consisting of H-vertices and G-vertices. A subtree is called *maximal* if the sets of H-vertices and G-vertices that it consists of cannot be extended. A *graft* is a maximal subtree that does not contain two H-vertices belonging to the same cycle. Finally, a *block* is a graft or a cycle.

It is not difficult to see that a cactus is formed by a set of blocks attached via H-vertices (see Fig. 2.(a)).

If we add an edge between the blocks and the H-vertices, we obtain the tree $T_G = (V_G, E_G)$ such that each element of V_G is a block or an H-vertex. Figure 2.(b) gives the tree representation of the cactus shown in Figs. 1 and 2.(a). We say that a cactus is *rooted* if the tree that represents it is rooted.

Given that constantly connected dynamic graphs based on trees (or grafts) are static and thus easy to explore, in this paper we consider cactuses that only consist of cycles and H-vertices. These cactuses will be called *plump cactuses*. Blocks are then always cycles, and we will use the term *cycle* in the sequel. In the following, we will assume that the cactus is rooted at the cycle where the agent starts exploration. If the agent starts on an H-vertex, one of the cycles attached to the H-vertex will be the *root cycle*.

In this paper, we use the classical formalism of static trees. We will talk about degree, child, parent, height or depth of a cycle. Instead of subtree, we will rather use the term *sub-cactus* of a cactus C to denote a cactus C' corresponding to a subtree in the rooted tree representation of C.

3 Chain Method

In this section, we give a simple algorithm inspired by DFS to explore constantly connected dynamic graphs based on a plump cactus of n vertices. The principle of the algorithm is very simple. If the agent enters a cycle it has not visited yet, it visits it using the algorithm EXPLORE-RING for exploring dynamic graphs based on the ring (see Theorem 3), then passes to the attachment vertex of its closest unexplored child and explores it recursively. If all its children have already been explored and there is a cycle not yet explored, then it goes to its parent.

Algorithm 1 CHAIN-METHOD().

```
 1: while not all vertices have been visited do
 2:     if the current cycle is not yet explored then
 3:         EXPLORE-RING (current cycle)
 4:     end if
 5:     if there is a child not yet explored then
 6:         GO-TO-THE-ATTACHMENT-VERTEX (with this child)
 7:     else
 8:         GO-TO-THE-ATTACHMENT-VERTEX (with the parent)
 9:     end if
10: end while
```

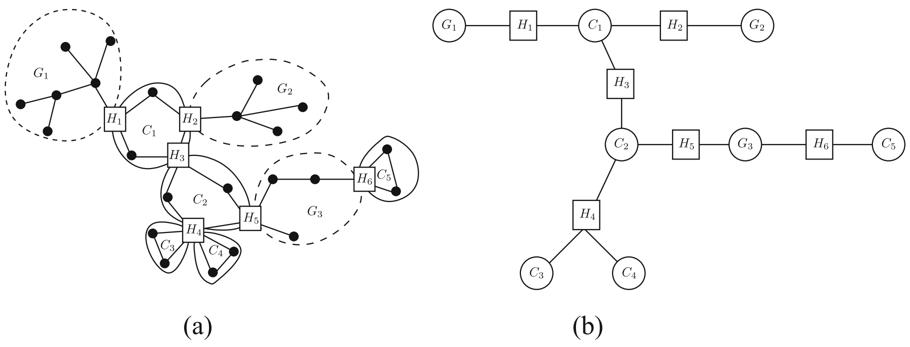

Fig. 2 Tree representation of a cactus

Theorem 4 *For any integer $n \geq 3$, and for any constantly connected dynamic graph based on a plump cactus of n vertices, there is an agent, executing the algorithm* CHAIN-METHOD*, able to explore this dynamic graph in at most $\sum_{i=1}^{k}(d_i+2)(n_i-1)$ time units, where n_i is the size of the cycle i, d_i its degree, and k the number of cycles of the cactus.*

Proof An agent executing the algorithm CHAIN-METHOD pays on each cycle R_{n_i} of the cactus at most $2n_i - 2$ time units to explore it (see Theorem 3). To switch to the attachment vertex of a child or the parent (if it has one), $n_i - 1$ time units are sufficient (see Theorem 2). As the degree of a cycle is equal to the number of its incident edges, then on each cycle R_{n_i} of the cactus, the agent pays at most $(d_i+2)(n_i-1)$ time units. The cactus is composed of k cycles, hence the agent pays at most $\sum_{i=1}^{k}(d_i+2)(n_i-1)$ time units to explore the dynamic graph. □

Note that if the degree of each cycle is constant, then the time to explore the dynamic graph using the CHAIN-METHOD is in $O(n)$, where n is the size of the cactus. Figure 3 presents a plump cactus of size n in which exploration using the CHAIN-METHOD takes time $\Omega(n^2)$. Indeed, any algorithm exploring this graph has to explore the $\Omega(n)$ attached cycles of length 3. However, when the CHAIN-METHOD is used, the order of visit of these cycles is fixed and the adversary may choose the dynamicity of the graph such that going from one attached cycle to the next takes time $\Omega(n)$. Hence the overall exploration time is $\Omega(n^2)$.

4 Star Method

As we have seen in the previous section, the algorithm CHAIN-METHOD is not effective for exploring constantly connected dynamic graphs based on cactuses with cycles of large degree, because the order of visit of the sub-cactuses is fixed, which makes the algorithm spend a lot of time to go from one sub-cactus to the next. On the contrary, the algorithm STAR-METHOD presented in this section focuses on reducing

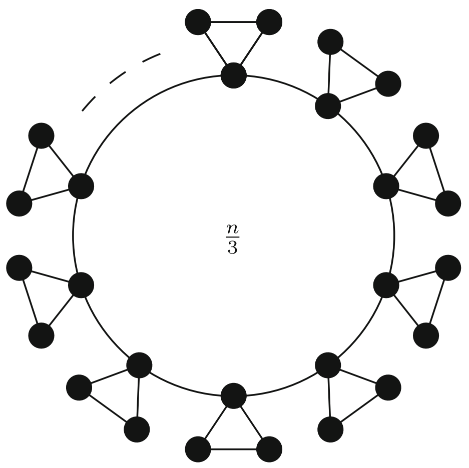

Fig. 3 Difficult graph for the CHAIN-METHOD

these transit times on the root cycle (the cycle where the exploration starts) to the minimum. The idea is to visit the sub-cactuses while exploring the root cycle. Note that directly using the algorithm EXPLORE-RING on the root cycle does not work, because when returning to the attachment vertex after exploring a sub-cactus, the agent cannot continue the exploration according to the algorithm EXPLORE-RING on the root cycle, as the dynamicity has changed on this cycle. To avoid this issue, the algorithm STAR-METHOD uses the algorithm EXPLORE-RING on a carefully chosen virtual dynamic cycle that takes into account both the dynamics of the root cycle and the time needed to recursively explore the sub-cactuses.

Theorem 5 *For any integer $n \geq 3$, and for any constantly connected dynamic graph based on a plump cactus C of n vertices, there is an agent, executing the algorithm* STAR-METHOD*, able to explore this dynamic graph in at most $f_S(C) = 3(n_r - 1)$ time units if C is an n_r-vertex cycle, or $f_S(C) = 3(n_r - 1) + \sum_{i=1}^{\ell} f_S(C_i) + \max_{1\leq i\leq \ell} f_S(C_i)$ time units otherwise, where n_r is the size of the root cycle, $\ell \geq 1$ is the number of sub-cactuses attached to the root cycle, and $f_S(C_i)$ is the recursive exploration cost of the sub-cactus C_i using the same algorithm.*

Proof Let C be a plump cactus, with n vertices, and let $\mathcal{G}$ be a constantly connected dynamic graph based on C. We prove the theorem by induction on the tree structure of the cactus. If C consists of a single ring (base case), then the algorithm STAR-METHOD simply applies the algorithm EXPLORE-RING and returns to the starting vertex, which proves the theorem in this case. Otherwise let n_r be the size of the root cycle, and let $C_1, C_2, \ldots, C_\ell$ be the sub-cactuses attached to this root cycle. Moreover, for a proof by induction, we assume that the theorem holds for these sub-cactuses.

Let f_S be the recursive function defined in the statement of the theorem. The algorithm STAR-METHOD proceeds as follows. First, we introduce the following transformation of $\mathcal{G}$ into another dynamic graph $\mathcal{G}'$, based on a ring $R_{n'}$ of size n'. The dynamic graph $\mathcal{G}'$ is constructed as follows. We retain the root cycle of C and the dynamics of the graph $\mathcal{G}$ on this cycle. We replace every H-vertex of C with two C-vertices linked by a sequence of static paths of length equal to the recursive cost of exploring each subtree attached to the H-vertex. More precisely, the lengths of the added paths are $f_S(C_i)$, for all the sub-cactuses C_i attached to the H-vertex. Thus, we obtain a constantly connected dynamic graph based on a ring of size n' (see Fig. 4). The dynamic graph $\mathcal{G}'$ is indeed constantly connected because we retain the dynamicity of the subgraph of $\mathcal{G}$ based on the root cycle of C, which itself respects the constant connectivity.

We use the fundamental properties behind EXPLORE-RING, namely Theorems 1 and 2, on respectively $\mathcal{G}'$ and $\mathcal{G}$, to obtain a traversal that efficiently explores the root cycle and the sub-cactuses altogether. More precisely, if t is the time after $n_r - 1$ time units elapsed, let $v^{(t)}$ be the vertex of $R_{n'}$ described in Theorem 1. If $v^{(t)}$ does not correspond to a vertex of the root cycle C, then we set v as the H-vertex in C corresponding to the static subpath containing $v^{(t)}$. Otherwise, v is simply the corresponding vertex in C.

Now let Agent B be the virtual agent, starting from the previously defined vertex $v^{(t)}$, that goes in the clockwise direction in $\mathcal{G}'$ without being blocked for $n' - 1$ time units (by Theorem 1). We define the Agent A following the STAR-METHOD as follows.

First Agent A uses $n_r - 1$ time units to reach the previously defined vertex v on C. This is possible thanks to the property from Theorem 2. Now, whenever the (virtual) Agent B stays on a subpath P corresponding to some sub-cactus C_i for at least $f_S(C_i)$ consecutive time units, Agent A uses this time to recursively explore the sub-cactus C_i. If, after completing this exploration, Agent B is still lying on P,

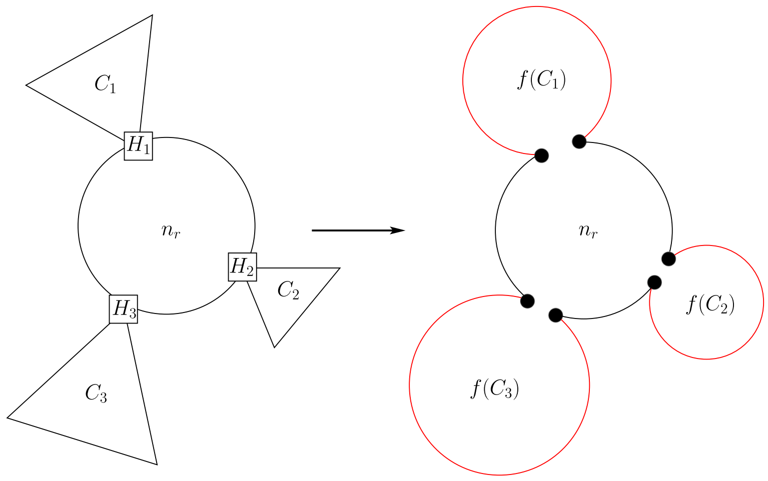

Fig. 4 Correspondence between the dynamic graph based on C and the dynamic graph based on $R_{n'}$

then Agent A simply waits on the attachment vertex. Whenever Agent B lies on the part corresponding to the root cycle (that is outside of the added subpaths), Agent A behaves exactly as Agent B.

This simulation of agent B on $\mathcal{G}$ takes at most $n'-1 = (n_r-1)+\sum_{i=1}^{\ell} f_S(C_i)$ time units. By construction, the root cycle and all sub-cactuses are explored, except for possibly one sub-cactus C_i, if the agent B starts (and ends) inside the static path corresponding to C_i. In this case, the agent A uses at most $f_S(C_i) \leq \max_{1\leq j\leq \ell} f_S(C_j)$ additional time units to explore C_i. Finally, in any case, the agent returns to the starting vertex, using at most $n_r - 1$ time units. Overall, this can be done in $f_S(C) = 3(n_r-1) + \sum_{i=1}^{\ell} f_S(C_i) + \max_{1\leq i\leq \ell} f_S(C_i)$ time units, which concludes the proof of the theorem. □

If the height of the rooted tree of the cactus is constant, then the time to explore the dynamic graph using the STAR-METHOD is $O(n)$ time units, where n is the size of the cactus. However, Fig. 5 presents a plump cactus of size n in which exploration using the STAR-METHOD may take $2^{\Omega(n)}n$ time units for a specific choice of missing edges at each time. Indeed, at each step of the induction, there is only one sub-cactus, and its cost is paid twice, once in the sum, and once in the max (cf. the formula in Theorem 5). The cycle of length $n/2$ to the right needs exploration time $\Omega(n)$. Then, recursively, each additional cycle of size 4 on its left will introduce a multiplicative factor of 2 in the recursive cost of the sub-cactus. As the number of cycles of size 4 is $\Omega(n)$, the overall exploration time is $2^{\Omega(n)}n$.

5 Mixed Method

In a plump cactus C with a root cycle of size n_r and with sub-cactuses $C_1, \ldots, C_\ell$, the recursive exploration time is $f_C(C) = 3(n_r-1) + \sum_{i=1}^{\ell} f_C(C_i) + \ell \cdot (n_r - 1)$ for the algorithm CHAIN-METHOD, and $f_S(C) = 3(n_r-1) + \sum_{i=1}^{\ell} f_S(C_i) + \max_{1\leq i\leq \ell} f_S(C_i)$ for the algorithm STAR-METHOD.

Because both methods presented above may have alone a large exploration time, we introduce in this section a combination of both methods, that is to say, on some sub-cactuses the agent will use the algorithm STAR-METHOD to explore them, and on the remaining sub-cactuses it will use the algorithm CHAIN-METHOD.

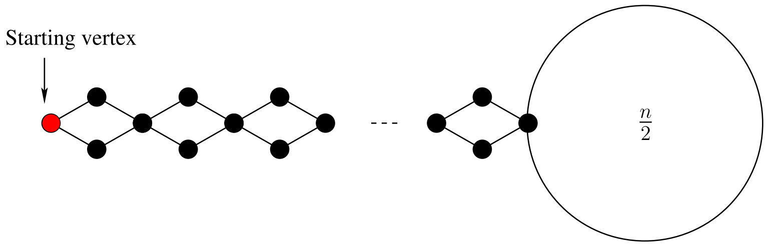

Fig. 5 Difficult graph for the STAR-METHOD

The algorithm MIXED-METHOD is recursively defined as follows, with a cost function f_M.

If the cactus is an n-vertex ring, then the agent uses the algorithm EXPLORE-RING (which is in fact what both methods do), with a cost of at most $f_M(C) = 3(n-1)$. Otherwise, let n_r be the number of vertices of the root cycle, and let $C_1, \dots, C_\ell$ be its sub-cactuses, with $\ell \geq 1$. Assume without loss of generality that the sub-cactuses $C_1, \dots, C_\ell$ are ranked in descending order of their exploration cost $f_M(C_i)$. The agent uses the algorithm CHAIN-METHOD for the k first sub-cactuses, and the algorithm STAR-METHOD for the other sub-cactuses, for a well-chosen k.

More precisely, for each sub-cactus C_i, with $1 \leq i \leq k$, the agent goes to the attachment vertex of C_i in at most $n_r - 1$ time units (Theorem 2), and explores C_i recursively using the algorithm MIXED-METHOD. Then it explores altogether what remains of the cactus (the root cycle and the cactuses C_i, for $i > k$) similarly as in the algorithm STAR-METHOD. The only difference is that the recursive costs $f_M(C_i)$ are used to construct the virtual ring instead of the costs $f_S(C_i)$. As a consequence, the sub-cactuses are recursively explored using the algorithm MIXED-METHOD instead of the algorithm STAR-METHOD.

The resulting cost is then $\sum_{i=1}^{k} \left(n_r - 1 + f_M(C_i)\right) + \left(3(n_r-1) + \sum_{i=k+1}^{\ell} f_M(C_i) + \max_{k+1 \leq i \leq \ell} f_M(C_i)\right)$. Using the fact that the costs $f_M(C_i)$ are decreasing, the preceding bound becomes $3(n_r - 1) + \sum_{i=1}^{\ell} f_M(C_i) + \left(k \cdot (n_r - 1) + f_M(C_{k+1})\right)$.[3] The algorithm MIXED-METHOD chooses k such as to minimize the additional cost $k \cdot (n_r - 1) + f_M(C_{k+1})$. To summarize, the exploration cost of the algorithm MIXED-METHOD is $f_M(C) = 3(n_r - 1) + \sum_{i=1}^{\ell} f_M(C_i) + \min_{1 \leq k \leq \ell}(k \cdot (n_r - 1) + f_M(C_{k+1}))$.

5.1 Upper Bound for the Algorithm MIXED-METHOD

In this section, we give an upper bound on the complexity of the algorithm MIXED-METHOD. The term $k \cdot (n_r - 1) + f_M(C_{k+1})$ is a priori not monotone with respect to k. Therefore, it is not clear how to handle the min in the formula defining the function f_M. To circumvent this issue, we study a variant of the algorithm MIXED-METHOD which chooses k more consistently.

Theorem 6 *An agent executing the algorithm* MIXED-METHOD *requires at most* $O\left(n \frac{\log n}{\log \log n}\right)$ *time units to explore any constantly connected dynamic graph based on a plump cactus of n vertices.*

Proof Fix an arbitrary constantly connected dynamic graph based on a plump cactus C of n vertices. In order to study the exploration cost of the algorithm MIXED-METHOD, we will discuss another algorithm, denoted EXPLORE-CACTUS, which

[3] To simplify the notation, we define $f_M(C_i)$ as 0 when $i > \ell$.

is less efficient but easier to analyze. The upper bound obtained for this less efficient algorithm will also give us a valid upper bound for the MIXED-METHOD. The algorithm EXPLORE-CACTUS is defined as the algorithm MIXED-METHOD except that the number k of sub-cactuses on which it uses the algorithm CHAIN-METHOD is always $\min\{\ell, 2c\}$, with $c = \frac{\log n}{\log\log n}$.

Therefore, the exploration cost $f_E(C)$ of the algorithm EXPLORE-CACTUS in a plump cactus C having a root cycle of size n_r and sub-cactuses $C_1, \ldots, C_\ell$ is at most $3(n_r - 1) + \sum_{i=1}^{\ell} f_E(C_i) + 2c(n_r - 1) + f_E(C_{2c+1})$. Among the first $2c$ sub-cactuses, there are at least c sub-cactuses whose number of vertices is at most n/c, where n is the total number of vertices in C. Also, all these sub-cactuses have a cost larger than $f_E(C_{2c+1})$. It is therefore possible to charge the additional cost $f_E(C_{2c+1})$ to these sub-cactuses. We obtain the upper bound $f_E(C) \le 3(n_r - 1) + \sum_{i=1}^{\ell} f_E(C_i) + 2c(n_r - 1) + \frac{1}{c}\sum_{|C_i| \le \frac{n}{c}} f_E(C_i)$.

Differently speaking, there is a multiplicative factor $1 + \frac{1}{c}$ in front of $f_E(C_i)$ for the sub-cactuses C_i such that $|C_i| \le \frac{n}{c}$. Since the number of vertices is divided by c, there can be at most $\log_c n$ such factors stacking in a branch of the cactus. Developing the recursive cost, we thus obtain a total exploration time of at most $n(1 + \frac{1}{c})^{\log_c n}(2c + 3)$. Using the fact that $\lim_{c \to +\infty} (1 + \frac{1}{c})^c = e$, if we replace c with its value, we obtain the claimed bound. This concludes the proof of the theorem. □

5.2 Lower Bound for the Algorithm MIXED-METHOD

It turns out that the algorithm MIXED-METHOD does not explore all constantly connected dynamic graphs based on a cactus of size n in $O(n)$ time units. We have the following theorem to prove it.

Theorem 7 *For infinitely many n, there is a constantly connected dynamic graph based on a plump cactus of n vertices such that the exploration of the dynamic graph by an agent executing the algorithm* MIXED-METHOD *takes at least* $\Omega\left(n\frac{\log n}{(\log\log n)^2}\right)$ *time units.*

Proof Let d be the triple of a sufficiently large power of 2 and let $h = \frac{1}{2}d\log d$. We construct a particular rooted plump cactus C_d for which the exploration cost $f_M(C_d)$ of the algorithm MIXED-METHOD is large, namely in $\Omega(d \cdot |C_d|)$.

To do so, we use the following transformation. Given an integer $i \ge 1$ and a cactus C, the cactus $\mathrm{sub}_i(C)$ is defined as the cactus C in which all edges have been subdivided in i edges. Note that, in particular, $\mathrm{sub}_1(C) = C$.

We now define C_d via an inductive construction. More precisely, we define the cactuses $C_{d,i}$, for $0 \le i \le h$ by induction on i. We denote by $m_{d,i}$, $r_{d,i}$, and $t_{d,i}$, the number of edges, the size of the root cycle, and the recursive cost $f_M(C_{d,i})$, of the cactus $C_{d,i}$.

First, let $C_{d,0}$ be a ring of $m_{d,0} = r_{d,0} = (d+3)/3$ edges (which is an integer by definition of h). The exploration cost of this cactus is $t_{d,0} = f_M(C_{d,0}) = 3(r_{d,0} - 1) = d$. For $i \geq 1$, we define inductively $C_{d,i}$ as a cactus with a root cycle of size $r_{d,i} = t_{d,i-1} + 1$ on which are attached on d different vertices the cactuses $\mathrm{sub}_1(C_{d,i-1})$, $\mathrm{sub}_2(C_{d,i-1})$, up to $\mathrm{sub}_d(C_{d,i-1})$ (see Fig. 6). Finally, C_d is defined as $C_{d,h}$.

The number of edges of $C_{d,i}$ is $m_{d,i} = r_{d,i} + \sum_{j=1}^{d}(j \cdot m_{d,i-1})$. Recall that the algorithm Mixed-method chooses the number k of sub-cactuses to explore with the algorithm Chain-method such as to minimize the additional cost $k \cdot (r_{d,i} - 1) + f_M(\mathrm{sub}_{d-k}(C_{d,i-1}))$. On one hand, we have $r_{d,i} - 1 = t_{d,i-1}$ by definition. On the other hand, the recursive exploration cost of each sub-cactus $\mathrm{sub}_j(C_{d,i-1})$ is $j \cdot t_{d,i-1}$. Therefore, the additional cost does not depend on k and is always equal to $d \cdot t_{d,i-1}$. In other words, the cactus is constructed in such a way that all the algorithms Chain-method, Star-method, and Mixed-method have the same exploration cost. Therefore, the exploration cost $f_M(C_{d,i})$ of $C_{d,i}$ is $t_{d,i} = 3(r_{d,i} - 1) + \sum_{j=1}^{d}(j \cdot t_{d,i-1}) + d \cdot t_{d,i-1} = (3 + d(d+1)/2 + d) \cdot t_{d,i-1} = ((d^2 + 3d + 6)/2) \cdot t_{d,i-1}$.

To simplify the notation, we now remove the first index d. Also, let $\alpha = (d^2 + 3d + 6)/2)$ and $\beta = d(d+1)/2$. To summarize, we have

$$\begin{aligned} t_0 &= d \\ r_0 &= (d+3)/3 \\ m_0 &= (d+3)/3 \end{aligned}$$

and, for $1 \leq i \leq h$,

$$\begin{aligned} t_i &= \alpha \cdot t_{i-1} \\ r_i &= t_{i-1} + 1 \\ m_i &= r_i + \beta \cdot m_{i-1} \,. \end{aligned}$$

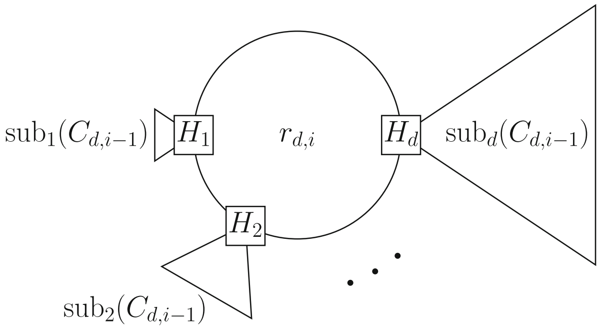
$\mathrm{sub}_1(C_{d,i-1})$
H_1
$r_{d,i}$
H_d
$\mathrm{sub}_d(C_{d,i-1})$
H_2
$\mathrm{sub}_2(C_{d,i-1})$

Fig. 6 Inductive construction of the cactus $C_{d,i}$

Solving the recurrences and setting $\gamma = \alpha/\beta$, we obtain

$$\begin{aligned}
t_h &= \alpha^h \cdot t_0 \\
r_h &= \alpha^{h-1} \cdot t_0 + 1 \\
m_h &= \beta^h \cdot m_0 + \sum_{i=1}^{h} \beta^{h-i} \cdot r_i \\
&= \beta^h \cdot m_0 + \sum_{i=1}^{h} (\alpha^{i-1} \cdot t_0 + 1)\beta^{h-i} \\
&= \beta^h \cdot m_0 + t_0 \cdot \sum_{i=0}^{h-1} \alpha^i \beta^{h-1-i} + \sum_{i=0}^{h-1} \beta^{h-1-i} \\
&= \beta^h \cdot m_0 + \frac{\beta^h - 1}{\beta - 1} + t_0 \cdot \beta^{h-1} \cdot \frac{\gamma^h - 1}{\gamma - 1}\,.
\end{aligned}$$

We now prove that the last term is somehow predominant. Indeed, we have

$$\frac{\beta^h \cdot m_0 + \frac{\beta^h - 1}{\beta - 1}}{t_0 \cdot \beta^{h-1} \cdot \frac{\gamma^h - 1}{\gamma - 1}} \le \frac{2\beta \cdot m_0}{t_0} \cdot \frac{\gamma - 1}{\gamma^h - 1} \le \beta \cdot \frac{\gamma - 1}{\gamma^h - 1}\,.$$

Besides, we have

$$\gamma = \frac{d^2 + 3d + 6}{d^2 + d} = 1 + \frac{2d + 6}{d(d+1)} = 1 + \frac{2}{d} + \frac{4}{d(d+1)}\,.$$

Plugging the last equation into the previous one, we obtain

$$\begin{aligned}
\frac{\beta^h \cdot m_0 + \frac{\beta^h - 1}{\beta - 1}}{t_0 \cdot \beta^{h-1} \cdot \frac{\gamma^h - 1}{\gamma - 1}} &\le \beta \cdot \frac{\frac{2}{d} + \frac{4}{d(d+1)}}{(1 + \frac{2}{d} + \frac{4}{d(d+1)})^h - 1} \\
&\le \beta \cdot \frac{\frac{3}{d}}{\left((1 + \frac{2}{d})^{\frac{d}{2}}\right)^{\frac{2h}{d}} - 1} \\
&\le \frac{3\beta}{d} \cdot \frac{1}{2^{logd} - 1} \\
&\le 2\,,
\end{aligned}$$

where the penultimate inequality uses the fact that $\lim_{x \to +\infty}(1 + 1/x)^x = e$ and the definition of h.

We are now ready to derive a lower bound on the exploration time t_h.

$$\begin{aligned} t_h &\geq \frac{\alpha^h \cdot t_0}{3t_0 \cdot \beta^{h-1} \cdot \frac{\gamma^h - 1}{\gamma - 1}} \cdot m_h \geq \frac{\alpha}{3} \cdot \gamma^{h-1} \cdot \frac{\gamma - 1}{\gamma^h - 1} \cdot m_h \\ &\geq \frac{\alpha}{3} \cdot \frac{\gamma - 1}{\gamma} \cdot m_h \geq \frac{\alpha}{3d} \cdot m_h \\ &\geq \frac{d}{6} \cdot m_h . \end{aligned}$$

It remains now to express d and m_h as a function of the number n of vertices of the cactus C_d.

$$2^d \leq \beta^h \leq \frac{1}{2} m_h \leq n \leq m_h \leq \beta^{2h} \leq d^{6h}$$

The inequality $2^d \leq n$ implies that $\log d \leq \log \log n$, while $d^{6h} = d^{3d \log d} \geq n$ allows to derive that $3d \log^2 d \geq \log n$ and thus that $d \geq \frac{1}{3} \frac{\log n}{(\log \log n)^2}$. Finally, we obtain $t_h \geq \frac{1}{18} n \frac{\log n}{(\log \log n)^2}$, which concludes the proof of the theorem. $\square$

6 Conclusion

In this paper, we studied the time complexity for exploring constantly connected dynamic graphs based on cactuses, under the assumption that the agent knows the dynamics of the graph. We gave an exploration algorithm for dynamic graphs that we called MIXED-METHOD, and we have shown that for exploring the whole class of constantly connected dynamic graphs based on cactuses of n vertices, with this algorithm, $\Omega(n \frac{\log n}{(\log \log n)^2})$ time units are necessary (for infinitely many n), and $O(n \frac{\log n}{\log \log n})$ time units are sufficient. This study opens several perspectives.

In the short term, it would be interesting to find a new method in order to obtain a better upper bound on the exploration time of dynamic graphs based on cactuses. At a second stage, an interesting question to investigate would be if T-interval-connectivity (for $T > 1$) allows to save a significant factor in the exploration time of the cactuses. A natural further objective is to extend the family of underlying graphs. Note that the families of underlying graphs considered so far (rings and cactuses) have the property that at most one edge can be absent at a given time in every bi-connected component. Studying families of underlying graphs that do not possess this property seems to be a challenging problem.

A further perspective is to consider the exploration problem of dynamic graphs using more than one agent, assuming standard models of communication between the agents. The objective would be to study whether dynamic graph exploration can be performed more efficiently by using more than one agent.

Acknowledgements The authors would like to thank Arnaud Casteigts for insightful and valuable discussions regarding the topic of this paper.

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, vol. 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, vol. 8847, pp. 22–38 (2014)
3. Agarwalla, A., Augustine, J., Moses, W.K. Jr., Madhav, S.K., Sridhar, A.K.: Deterministic dispersion of mobile robots in dynamic rings. In: 19th international conference on distributed computing and networking, ICDCN 2018, pp. 19:1–19:4 (2018)
4. Bodlaender, H.L., van der Zanden, T.C.: On exploring always-connected temporal graphs of small pathwidth In. Inform Process Lett **142**, 68–71 (2019)
5. Bournat, M., Dubois, S., Petit, F.: Computability of perpetual exploration in highly dynamic rings. In: 37th IEEE international conference on distributed computing systems (ICDCS), IEEE computer society, pp. 794–804 (2017)
6. Burkard, R., Krarup, J.: A linear algorithm for the Pos/Neg-Weighted 1-Median problem on a cactus. In Comput **60**(3), 193–216 (1998)
7. Casteigts, A., Flocchini, P., Quattrociocchi, W., Santoro, N.: Time-varying graphs and dynamic networks. In: Int J Parall, Emerg Distrib Syst **27**(5), 387–408 (2012)
8. Das, S., Di Luna, G.A., Gasieniec, L.A.: Patrolling on dynamic ring networks. In: 45th international conference on current trends in theory and practice of computer science (SOFSEM), LNCS, vol. 11376, pp. 150–163 (2019)
9. Di Luna, G.A., Dobrev, S., Flocchini, P., Santoro, N.: Distributed exploration of dynamic rings. Distribut Comput **33**(1), 41–67 (2020)
10. Di Luna, G.A., Flocchini, P., Pagli, L., Prencipe, G., Santoro, N., Viglietta, G.: Gathering in dynamic rings. In theoretical computer science **811**, 79–98 (2020)
11. Dutta, C., Pandurangan, G., Rajaraman, R., Sun, Z., Viola, E.: On the complexity of information spreading in dynamic networks. In: Proceedings of the twenty-fourth annual ACM-SIAM symposium on discrete algorithms (SODA), pp. 717–736 (2013)
12. Erlebach, T., Hoffmann, M., Kammer, F.: On temporal graph exploration. In: 42nd international colloquium on automata, languages, and programming (ICALP), LNCS, vol. 9134, pp. 444–455 (2015)
13. Erlebach, T., Spooner, J.T.: Faster exploration of degree-bounded temporal graphs. In: 43rd international symposium on mathematical foundations of computer science (MFCS), pp. 36:1–36:13 (2018)
14. Ferreira, A.: Building a reference combinatorial model for dynamic networks: initial results in evolving graphs. INRIA RR-5041 (2003)
15. Gotoh, T., Flocchini, P., Masuzawa, Santoro, N.: Tight bounds on distributed exploration of temporal graphs. In: 23rd international conference on principles of distributed systems (OPODIS), vol. 153, pp. 22:1–22:16 (2019)
16. Gotoh, T., Sudo, Y., Ooshita, F., Kakugawa, H., Masuzawa, T.: Group Exploration of Dynamic Tori. In: 38th IEEE international conference on distributed computing systems (ICDCS), IEEE computer society, pp. 775–785 (2018)
17. Gotoh, T., Sudo, Y., Ooshita, F., Masuzawa, T.: Dynamic Ring Exploration with (H,S) View In. Algorithms **13**(6), 141 (2020)
18. 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, vol. 8576, pp. 250–262 (2014)
19. Ilcinkas, D., Wade, A.M.: Exploration of the T-Interval-Connected dynamic graphs: the case of the ring. In: Theory of Computing Systems, vol. 62, pp. 1144–1160 (2018)
20. 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)
21. Kuhn, F., Oshman, R.: Dynamic networks: models and algorithms. In ACM SIGACT News **42**(1), 82–96 (2011)
22. Michail, O.: An introduction to temporal graphs: an algorithmic perspective. In Int Math **12**(4), 239–280 (2016)

23. Michail, O., Spirakis, P.G.: Traveling salesman problems in temporal graphs. In: Theoretical Computer Science, vol. 634, pp. 1–23 (2016)
24. O'Dell, R., Wattenhofer, R.: Information dissemination in highly dynamic graphs. In: DIALM-POMC, pp. 104–110 (2005)
25. Shannon, C.E.: Presentation of a maze-solving machine. In: 8th Conference of the Josiah Macy Jr. Found. (Cybernetics), pp. 173–180 (1951)