Exploration of Constantly Connected Dynamic Graphs Based on Cactuses

David Ilcinkas, Ralf Klasing, Ahmed Mouhamadou Wade

Exploration of Constantly Connected Dynamic Graphs Based on Cactuses*

David Ilcinkas, Ralf Klasing, and Ahmed Mouhamadou Wade

LaBRI, CNRS & Bordeaux University
{ilcinkas,klasing,wade}@labri.fr

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 [5]. 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 $2^{O(\sqrt{\log n})}n$ time units. We show that the lower bound of the algorithm is $2^{\Omega(\sqrt{\log n})}n$ time units.
Keywords: Exploration, Dynamic graphs, Mobile agent, Connectivity over time

1 Introduction

Exploration of a graph by a mobile agent (physical or software) is the task that the mobile agent, starting at a vertex of the graph, visits all vertices at least once. 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 the environment.

This fundamental problem in distributed computing by mobile agents has been extensively studied since the seminal paper by Claude Shannon [12]. 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 [2] a comprehensive overview of the different models and studies of dynamic graphs (see also [7]).

* Partially supported by the ANR project DISPLEXITY (ANR-11-BS02-014). This study has been carried out in the frame of "the Investments for the future" Programme IdEx Bordeaux – CPU (ANR-10-IDEX-03-02).

One of the first models developed, and also one of the most classic, is the model of evolving graphs [4]. 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 $\mathcal{G}$ is called *constantly connected* if each graph $\mathcal{G}_i$ which composes it is connected. This class of graphs was used in [10] to study the problem of information dissemination. In 2010, Kuhn, Lynch and Oshman [6] 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 [3, 6].

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-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 follows directly from the fact that the temporal diameter of these graphs is bounded by n.

The problem of exploration of constantly connected dynamic graphs has already been studied in the case where the underlying graph of the dynamic graph is a ring of n vertices [5]. That article shows that if the agent knows the dynamics of the graph, $2n - 3$ units of time are necessary and sufficient to solve the problem. The goal of this paper is to extend these results to larger families of underlying graphs. Unfortunately, the problem turns out to be much more difficult than it seems. 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. The difficulty of the exploration problem in general dynamic graphs is further underlined by the fact that the exploration problem for static graphs is the well-known Graph TSP problem (see e.g. [8, 9, 11]), which is already APX hard in general graphs.

Our results. At a first instance, we will 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 $2^{O(\sqrt{\log n})}n$ time units, and we derive a lower bound of $2^{\Omega(\sqrt{\log n})}n$ time units for the algorithm.

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.

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 to every integer $i \geq 0$ 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 graph $G = (V, E)$ in which two connected cycles have at most one vertex in common (see Figure 1).*

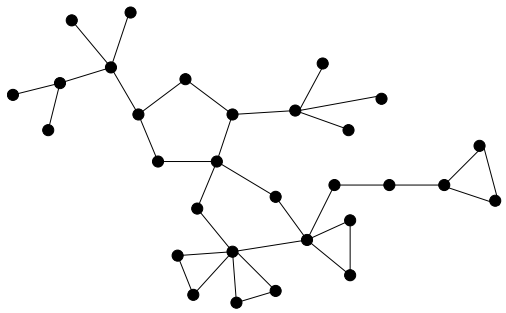

Fig. 1: Example of a cactus

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 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.

Theorem 1. *[5] 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*, capable of exploring this dynamic graph in at most $2n-3$ time units, when the agent knows the dynamics of the graph.*

Theorem 2. *[6] 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.*

Corollary 1. *For every constantly connected dynamic graph on n vertices, there exists an agent (algorithm) capable of exploring this dynamic graph in $O(n^2)$ time units, when the agent knows the dynamics of the graph.*

To give a simpler analysis of our algorithms, we consider the tree representation of a cactus given in [1].

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 ≥ 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 Figure 2.(a)).

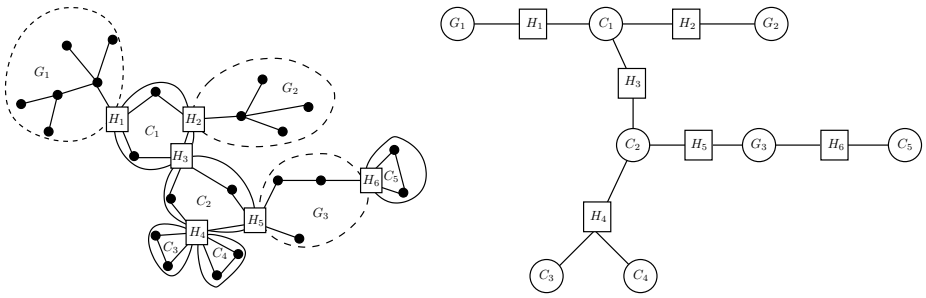

Fig. 2: Tree representation of a cactus

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 Figure 1. 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, in this paper we consider cactuses that only consist of cycles and H-vertices. In the following, we will assume that the cactus is rooted at the block where the agent starts exploration. If the agent starts on an H-vertex, one of the blocks attached to the H-vertex will be the starting block.

In this paper, we use the classical formalism of static trees. We will talk about degree, child, parent, height or depth of a block.

3 Chain method

In this section, we give a simple algorithm inspired by DFS to explore constantly connected dynamic graphs based on a cactus of n vertices. The principle of the algorithm is very simple. If the agent enters a ring it has not visited yet, it visits it using the algorithm Explore-ring for exploring dynamic graphs based on the ring (see Theorem 1), then passes to the point of attachment of its closest unexplored child and explores it recursively. If all its children have already been explored and there is a ring 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 ring is not yet explored then
3:       Explore-ring (current ring)
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
```

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

Proof. An agent executing the algorithm Chain-method pays on each ring R_{n_i} of the cactus at most $2n_i - 3$ units of time to explore it (see Theorem 1). To switch to the point of attachment 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 block is equal to the number of incident edges, then on each ring R_{n_i} of the cactus, the agent pays at most $(d_i+2)n_i-(d_i+3)$ units of time. The cactus is composed of k rings, hence the agent pays at most $\sum_{i=1}^{k}((d_i+2)n_i-(d_i+3))$ units of time to explore the dynamic graph. □

Note that if the degree of each ring 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 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 adversary may choose the dynamicity of the graph such that changing from one attached cycle to another takes time $\Omega(n)$, hence the overall exploration time is $\Omega(n^2)$.

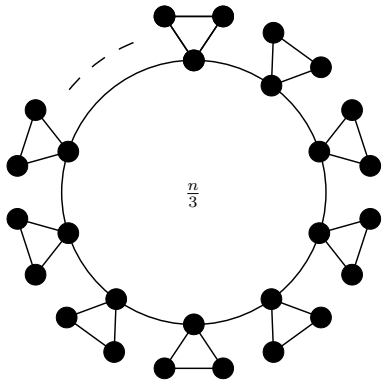

Fig. 3: Difficult graph for the CHAIN-METHOD

4 Star method

Because the exploration method that we gave earlier is not effective for exploring constantly connected dynamic graphs based on cactuses with rings of large degree, this section provides an exploration technique to overcome this.

The algorithm we give here uses a similar technique as the exploration algorithm for dynamic graphs based on the ring. Assume that the agent starts exploring from some vertex of some constantly connected dynamic graph $\mathcal{G}$ based on a cactus C of n vertices. From the starting point, the agent explores the starting ring. The major difference with the exploration algorithm for dynamic graphs based on the ring is that when an agent arrives at a vertex where an unexplored subtree is attached, it explores the subtree recursively and then it returns to the point of attachment and continues its exploration. However, when returning to the point of attachment, the problem is that the agent cannot continue the exploration according to the basic exploration algorithm on the starting ring, as the dynamicity has changed on the ring.

In order to cope with this dynamicity problem, we need to refine the approach appropriately. We take into account the time needed to recursively explore the

sub-cactuses by introducing the following transformation of $\mathcal{G}$ into another dynamic graph $\mathcal{G}'$, based on a ring $R_{n'}$ of larger size n'. The dynamic graph $\mathcal{G}'$ is constructed as follows. We retain the starting ring of C and the dynamics of the graph $\mathcal{G}$ based on this part. We replace every H-vertex of C with two C-vertices by adding a static path of length equal to twice the recursive cost of exploring the subtree attached to the H-vertex. Thus, we obtain a constantly connected dynamic graph based on a ring of size n' (see Figure 4). The dynamic graph $\mathcal{G}'$ is constantly connected because we retained the dynamics of the subgraph of $\mathcal{G}$ based on the starting ring of C, which respects the constant connectivity.

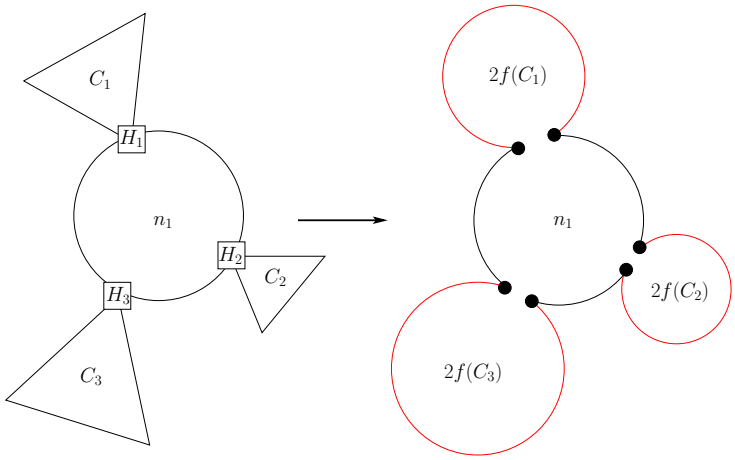

Fig. 4: Correspondence between the dynamic graph based on C and the dynamic graph based on $R_{n'}$

Theorem 4. *For any integer $n \geq 3$ and for any constantly connected dynamic graph based on a cactus C, there is an agent (algorithm) capable to explore this dynamic graph in at most $\sum_{i=1}^{k} 2^{p_i}(3n_i - 3)$ time units, where p_i is the depth of the ring i in the rooted tree, n_i is the size of the ring i in the rooted tree, k the number of rings of the cactus, and $n = \sum_{i=1}^{k} n_i - k + 1$ the number of vertices of the cactus.*

Proof. For some $n \geq 3$, let C be a cactus with n vertices and let $\mathcal{G}$ be a constantly connected dynamic graph based on C. Let us first determine the size of the dynamic graph $\mathcal{G}'$ based on $R_{n'}$ which is obtained from $\mathcal{G}$ by the above construction.

Suppose that C is rooted at the starting block. By construction, the size n' of $\mathcal{G}'$ is the sum of the size of the root ring plus the sum of twice the costs of the recursive exploration of the sub-cactuses that are attached, using the Star-method.

Denote by $f(C)$ the cost of exploring any constantly connected dynamic graph based on the cactus C using the Star-method. If C is reduced to a ring of size n, then $f(C) = 3n - 4$, because to explore a ring of size n and return to the starting vertex, an agent executing the algorithm Explore-ring needs at most $3n - 4$ time units. Otherwise let n_1 be the size of the root ring, and let $C_1, C_2, \ldots, C_\ell$ be the sub-cactuses attached to the root, then we have

$$f(C) = 3(n_1 - 1) + 2\sum_{i=1}^{\ell} f(C_i). \tag{1}$$

In order to obtain the recursive cost (1), we use the following algorithm for exploring a dynamic ring. For a constantly connected dynamic graph based on a ring R_N, one virtually deploys one agent on each vertex of the ring R_N, using $N - 1$ time units. The virtual agents then move in clockwise direction along the ring whenever they can. As there are N agents and in each round, only one agent can be held up by the adversary, after $N - 1$ rounds there is one (virtual) agent that has never been held up, hence this agent explores the ring in $N - 1$ additional time units. This agent is chosen as the actual exploration algorithm.

We consider a slightly modified version of this algorithm to explore the transformed dynamic graph $\mathcal{G}'$. Instead of allocating $n' - 1$ time units for the deployment phase, we assume that $n_1 - 1$ time units are sufficient. Now let Agent B be the virtual agent that is never held up in $\mathcal{G}'$. We define the Agent A following the Star-method as follows.

First Agent A uses $n_1 - 1$ time units to reach the starting node v of Agent B. If v is not a node of the starting ring, then Agent A goes to the attachment node in C corresponding to the static subpath containing v.

Now, whenever the (virtual) Agent B stays on a subpath P corresponding to some sub-cactus C_i for at least $f(C_i)$ 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, then Agent A simply waits on the attachment node. Whenever Agent B lies on the part corresponding to the starting ring (that is outside of the added subpaths), Agent A behaves exactly as Agent B. This part of the exploration of $\mathcal{G}$ takes at most $(n_1 - 1) + 2\sum_{i=1}^{\ell} f(C_i)$ time units.

After that, Agent A returns to its starting position. This takes at most $n_1 - 1$ time units.

Solving recurrence (1), we obtain the bound announced in 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. Figure 5 presents a cactus of size n in which exploration using the Star-method takes time $2^{\Omega(n)}n$. Indeed, when using the Star-method, from the starting point, the agent explores the starting cycle. When it reaches the rightmost vertex of the starting cycle, it explores the sub-cactus attached to the right recursively. However, the time allocated by the Star-method to do so corresponds to twice the exploration time of the sub-cactus. Hence, recursively, each additional cycle of length 4 will introduce an additional

factor of 2 in the cost. As the number of cycles of length 4 is $\Omega(n)$ and the cycle of length $n/2$ to the right needs exploration time $\Omega(n)$, the overall exploration time is $2^{\Omega(n)}n$.

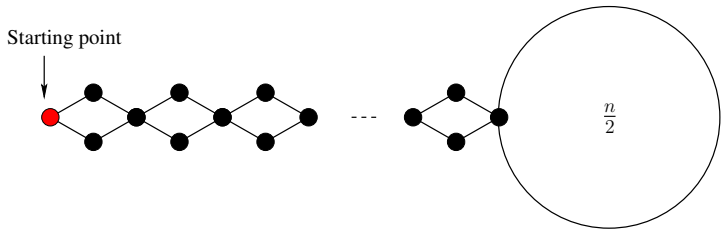

Fig. 5: Difficult graph for the Star-method

5 Mixed method

Note that if the agent is on a block that has a subtree attached to it, then the extra cost of exploring the block plus the subtree is equal to the block size minus one if the agent uses the Chain-method, and it is equal to the cost of exploring the subtree if the agent uses the Star-method. Because none of the two methods presented above alone allows to have a bound of $O(n)$ without further assumptions, in this section we introduce a combination of both methods, that is to say, on some blocks the agent will use the Star-method to explore, and on the remaining blocks it will use the Chain-method. The use of the two methods is as follows. If the agent is on a block that has no child, then it uses the ring exploration algorithm. Otherwise, on a block and a given subtree, in order to choose its method of exploration, the agent will compare the cost of exploring the subtree with the block size. If the block size is greater than the cost of exploring the subtree, then the agent uses the Star-method to explore the block and the subtree, otherwise it uses the Chain-method to explore them. In the following, we call this exploration algorithm Mixed-method.

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.

Theorem 5. *An agent executing the algorithm* Mixed-method *needs at most* $2 \cdot 2^{2\sqrt{\log n}} \cdot n$ *time units to explore any constantly connected dynamic graph based on a cactus of* n *vertices.*

Proof. Fix an arbitrary constantly connected dynamic graph based on a cactus C of n vertices. In order to study the exploration used by the MIXED-METHOD, we will discuss another algorithm, denoted EXPLORE-CACTUS, which 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. Given a parent ring R_{n_1} in the cactus, let $C_1, \ldots, C_\ell$ be its sub-cactus children. The MIXED-METHOD chooses for each child the best of the STAR-METHOD and the CHAIN-METHOD in terms of the time for exploring the sub-cactus and the size of the parent. The algorithm EXPLORE-CACTUS itself chooses the method to be used according to the criteria below. Assume without loss of generality that the sub-cactuses $C_1, \ldots, C_\ell$ are ranked in descending order of their number of vertices. The algorithm EXPLORE-CACTUS chooses the CHAIN-METHOD for the sub-cactuses $C_1, \ldots, C_{c-1}$, and the STAR-METHOD for the sub-cactuses $C_c, \ldots, C_\ell$, where $c = 2^{\sqrt{\log n}}$. According to the ordering of the sub-cactuses, the number of vertices of each sub-cactus $C_c, \ldots, C_\ell$ cannot exceed a fraction $1/c$ of the total number of vertices of the cactus rooted at the parent R_{n_1}. Therefore, a ring cannot have more than $\log_c n$ ancestors (potentially including itself) for which the STAR-METHOD was chosen. In summary, the total time used by the algorithm EXPLORE-CACTUS on the dynamic graph based on C is at most $2^{\log_c n}(c - 1 + 3)n \leq 2 \cdot 2^{2\sqrt{\log n}} n$ by definition of c. 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 6. *There is a constantly connected dynamic graph based on a cactus of n vertices such that the exploration of the dynamic graph by an agent executing the algorithm* MIXED-METHOD *takes at least $1/2 \cdot 2^{\sqrt{\log n}} \cdot n$ time units.*

Proof. Let h be an arbitrary even integer. Let $d = 2^{h+1}$. Consider a cactus based on a rooted complete d-ary tree of height h, that is to say all internal vertices have exactly d children and all of whose leaves are at distance h from the root (i.e. at depth h). For p between 0 and h, let $f_h(p) = d(2d+3)^{h-p} - \sum_{i=0}^{h-p-1}(2d+3)^i$. Any internal vertex of depth p is a ring of size $f_h(p+1)+1$. The leaves are cycles of size $\frac{d+4}{3}$ (which is an integer by definition of h). For any cycle, the points in common with the parent cycle and with each of the d child cycles, if they exist, are all different (see Figure 6). Let $t_h(p)$ be the time that algorithm MIXED-METHOD uses on a sub-cactus rooted at a cycle of depth $p \leq h$. We now prove that for any $p \leq h$, we have $t_h(p) = f_h(p)$. The proof is by induction on $(h-p)$. By Theorem 1, for $p = h$, we have $t_h(h) = 3\frac{d+4}{3} - 4 = d = f_h(h)$. Fix p such that $1 \leq p \leq h$ and suppose by induction hypothesis that $t_h(p) = f_h(p)$. At a cycle of depth $p-1$, for each of its children, the two methods are equivalent. Hence, the time used by the algorithm MIXED-METHOD will be $t_h(p-1) =$

$2\left(f_h(p)+1+d\cdot t_h(p)\right)-3+f_h(p)+1-1$. After simplification, and using the induction hypothesis, we obtain $t_h(p-1)=(2d+3)f_h(p)-1$, which is equal to $f_h(p-1)$. This concludes the proof by induction. Hence, the total exploration time of the cactus by the algorithm MIXED-METHOD is $f_h(0)$. We now compute a lower bound on $f_h(0)$. We have

$$\begin{aligned}
f_h(0) &= d(2d+3)^h - \sum_{i=0}^{h-1}(2d+3)^i \\
&= d(2d+3)^h - \frac{(2d+3)^h-1}{2d+2} \\
&\geq \frac{d-1}{2d+2}\cdot(2d+3)^h \\
&\geq 2d^2\cdot(2d+3)^{h-1} \\
&\geq 2^h\cdot d^{h+1} \\
&\geq \frac{d}{2}\cdot d^{h+1}
\end{aligned}$$

We now calculate the total number n of vertices of the cactus. According to the definition of the cactus, we have $n=\sum_{p=0}^{h-1}\left(d^p\cdot f_h(p+1)\right)+d^h\cdot\frac{d+4}{3}+1$. Therefore,

$$\begin{aligned}
n &\leq \sum_{p=0}^{h-1}\left(d^p\cdot d(2d+3)^{h-p-1}\right)+d^{h+1}/3 \\
&\leq d(2d+3)^{h-1}\sum_{p=0}^{h-1}(d/(2d+3))^p+d^{h+1}/3 \\
&\leq 2d(2d+3)^{h-1}+d^{h+1}/3 \\
&\leq (2d)^h(1+3/(2d))^{h-1}+d^{h+1}/3 \\
&\leq d^{h+1}/2\cdot 4/3+d^{h+1}/3 \\
&\leq d^{h+1}.
\end{aligned}$$

From this, we deduce that $d\geq 2^{\sqrt{\log n}}$. Combining all these bounds, we obtain $f_h(0)\geq 1/2\cdot 2^{\sqrt{\log n}}\cdot n$, which concludes the proof. □

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,

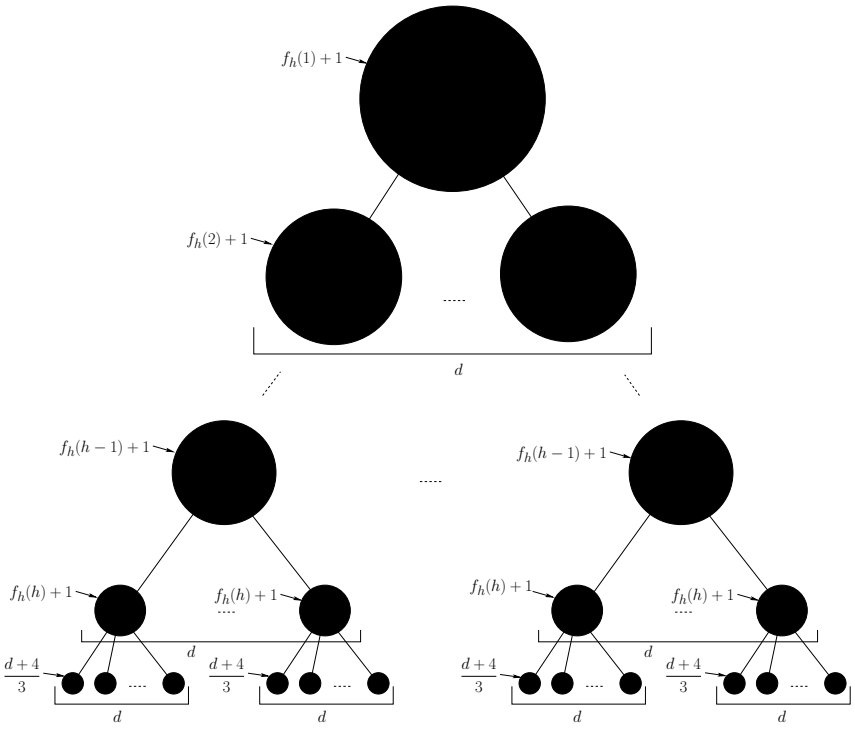

Fig. 6: Lower bound for the MIXED-METHOD

with this algorithm, $2^{\Theta(\sqrt{\log n})} \cdot n$ units of time are necessary and 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 (ring 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 more general objective is to establish whether there is an agent which knows the dynamics of the graph and which is able to explore all T-interval-connected dynamic graphs where the underlying graph has m edges in time $O(m)$, or even $o(m)$. A further perspective is to consider the exploration problem of T-interval-connected 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. Finally, the computational complexity of the exploration problem for dynamic graphs is largely unknown. As noted in the Introduction, the exploration problem for static graphs is already APX HARD in general graphs, hence the exploration problem for dynamic graphs is at least APX HARD in general graphs. However, it is not known whether this non-approximability result for dynamic graphs is tight, and whether efficient approximation algorithms for the exploration problem in dynamic graphs can be derived.

References

1. R. Burkard and J. Krarup: A Linear Algorithm for the Pos/Neg-Weighted 1-Median Problem on a Cactus. *Computing*, volume 60(3), pages 193–216, 1998.
2. A. Casteigts, P. Flocchini, W. Quattrociocchi, and N. Santoro: Time-varying graphs and dynamic networks. *International Journal of Parallel, Emergent and Distributed Systems*, volume 27(5), 2012.
3. C. Dutta, G. Pandurangan, R. Rajaraman, and Z. Sun: Information spreading in dynamic networks. *CoRR*, abs/1112.0384, 2011.
4. A. Ferreira: Building a Reference Combinatorial Model for Dynamic Networks: Initial Results in Evolving Graphs. INRIA, RR-5041 (2003)
5. D. Ilcinkas and A.M. Wade. Exploration of the T-Interval-Connected Dynamic Graphs: the Case of the Ring. In *Structural Information and Communication Complexity (SIROCCO)*, LNCS 8179, pages 13–23, 2013.
6. F. Kuhn, N.A. Lynch, and R. Oshman: Distributed computation in dynamic networks. In *42nd ACM Symposium on Theory of Computing (STOC)*, pages 513–522, 2010.
7. F. Kuhn and R. Oshman, Dynamic networks: models and algorithms. *ACM SIGACT News*, volume 42(1), pages 82–96, 2011.
8. T. Mömke and O. Svensson: Approximating Graphic TSP by Matchings. In *52nd IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 560–569, 2011.
9. M. Mucha: 13/9-approximation for Graphic TSP. In *29th Int. Symposium on Theoretical Aspects of Computer Science (STACS)* pages 30–41, 2012.
10. R. O'Dell and R. Wattenhofer: Information dissemination in highly dynamic graphs. In *DIALM-POMC*, pages 104–110, 2005.
11. A. Sebö, J. Vygen: Shorter Tours by Nicer Ears: 7/5-approximation for graphic TSP, 3/2 for the path version, and 4/3 for two-edge-connected subgraphs. *Combinatorica*, to appear.
12. C.E. Shannon: Presentation of a maze-solving machine. In *8th Conf. of the Josiah Macy Jr. Found. (Cybernetics)*, pages 173–180, 1951.