



HAL
open science

Exploration of Dynamic Cactuses with Sub-logarithmic Overhead

David Ilcinkas, Ahmed M Wade

► **To cite this version:**

David Ilcinkas, Ahmed M Wade. Exploration of Dynamic Cactuses with Sub-logarithmic Overhead. Theory of Computing Systems, 2021, 65, pp.257-273. 10.1007/s00224-020-10001-0 . hal-03036749

HAL Id: hal-03036749

<https://hal.science/hal-03036749>

Submitted on 2 Dec 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

1 **Exploration of Dynamic Cactuses with Sub-logarithmic**
2 **Overhead**

3 **David Ilcinkas · Ahmed M. Wade**

4
5 Received: date / Accepted: date

6 **Abstract** We study the problem of exploration by a mobile entity (agent) of a class
7 of dynamic networks, namely constantly connected dynamic graphs. This problem
8 has already been studied in the case where the agent knows the dynamics of the
9 graph and the underlying graph is a ring of n vertices [19]. In this paper, we consider
10 the same problem and we suppose that the underlying graph is a cactus graph (a
11 connected graph in which any two simple cycles have at most one vertex in common).
12 We propose an algorithm that allows the agent to explore these dynamic graphs in
13 at most $O(n \frac{\log n}{\log \log n})$ time units. We show that the lower bound of the algorithm is
14 $\Omega(n \frac{\log n}{(\log \log n)^2})$ time units (for infinitely many n).

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

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).

D. Ilcinkas
LaBRI, CNRS & Univ. Bordeaux, France
Tel.: +33 5 4000 69 12
E-mail: david.ilcinkas@labri.fr

A. M. Wade
École Polytechnique de Thiès, Senegal
Tel.: +221 77 142 66 70
E-mail: awade@ept.sn

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 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 [28]. 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]).

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 $\mathcal{G} = (G_1, G_2, \dots)$ is called *constantly connected* if each graph G_i which composes it is connected. This class of graphs was used in [26] 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

61 information about network changes is considered in [17]. More precisely, the authors
 62 study the exploration time for a single agent which knows the dynamics of the graph
 63 for the next S steps in its H -hop neighborhood, for given parameters S and H .

64 The problem of perpetual exploration is considered in [5, 15]. In [5], the authors
 65 consider that all agents are active at each time step and show that to solve the problem,
 66 one agent is sufficient in the rings of size two¹, two agents are sufficient in the rings
 67 of size three, and three agents are sufficient for all other rings. In [15] the authors
 68 consider time varying graphs whose topology is arbitrary and unknown to the agents
 69 and investigates the number of agents that are necessary and sufficient to explore
 70 such graphs. In addition to the problem of exploration, the problem of dispersion of a
 71 team of agents [3], gathering [10] and patrolling by a team of agents [8] are studied,
 72 considering constantly connected dynamic graphs based on the ring.

73 Besides, several papers focus on the complexity of computing the optimal explo-
 74 ration time of a dynamic graph given as (a centralized) input, in a similar manner
 75 as in the Traveling Salesman Problem for static graphs. In the dynamic case, the
 76 problem is called Temporal Graph Exploration Problem [4, 12, 23] or Dynamic Map
 77 Visitation Problem [1, 2]. In [2], the case of several agents is considered, while [4,
 78 12, 23] and most of [1] consider the case of a single agent. The problem is shown to
 79 be NP-complete, even when the underlying graph has pathwidth 2 and at each time
 80 step, the current graph is connected [4]. In the other papers, several polynomial-time
 81 algorithms are given, either exact algorithms for specific graph classes, or approxi-
 82 mation algorithms for the general cases. In particular, [1] gives an $O(n^2)$ algorithm to
 83 compute the optimal exploration time of a given 1-interval-connected dynamic graph
 84 based on the n -vertex ring. Inapproximability results for the general case are given
 85 in [12, 23].

86 It turns out that the problem of exploration is much more complex in dynamic
 87 graphs than in static graphs. Indeed, let us consider for example the scenario where
 88 the dynamic graph is known. The worst-case exploration time of n -vertex static graphs
 89 is clearly in $\Theta(n)$ (worst case $2n - 3$). On the other hand, the worst-case exploration
 90 time of n -vertex (1-interval-connected) dynamic graphs remains largely unknown.
 91 In [12] the authors give a worst-case lower bound in $\Omega(n^2)$ for general graphs and
 92 $\Omega(n \log n)$ for degree-bounded graphs. An upper bound in $O(d \log d \cdot \frac{n^2}{\log n})$ for degree-
 93 bounded graphs is given in [13].

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

98 *Our results.* We will first give two exploration methods that are efficient for exploring
 99 a very large set of constantly connected dynamic graphs based on a cactus, when the
 100 agent knows the dynamics of the graph. We will then combine these two exploration
 101 methods. We show that the combination of the two methods yields an algorithm that
 102 explores all constantly connected dynamic graphs based on a cactus of n vertices in

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.

103 $O(n \frac{\log n}{\log \log n})$ time units, and we derive a lower bound of $\Omega(n \frac{\log n}{(\log \log n)^2})$ time units for
 104 the algorithm (for infinitely many n).

105 2 Preliminaries

106 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
 107 studied in this paper.
 108

109 **Definition 1 (Dynamic graph)** A *dynamic graph* is a pair $\mathcal{G} = (V, \mathcal{E})$, where V is a
 110 static set of n vertices, and \mathcal{E} is a function which maps every integer $i \geq 0$ to a set
 111 $\mathcal{E}(i)$ of undirected edges on V .

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

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

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

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

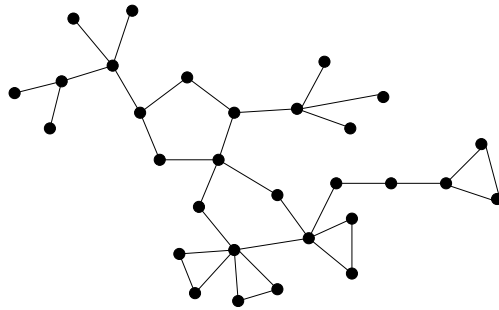


Fig. 1 Example of a cactus.

122 A mobile entity, called *agent*, operates on these dynamic graphs. The agent can
 123 traverse at most one edge per time unit. It may also stay at the current vertex (typically
 124 to wait for an incident edge to appear). We say that an agent *explores* the dynamic
 125 graph if and only if it visits all the vertices.

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

127 **Theorem 1** [19] For every integers $n \geq 3$ and $t \geq 0$, and for every constantly con-
 128 nected dynamic graph based on a ring with n vertices, there exists a vertex $v^{(t)}$ such
 129 that an agent starting at time t on $v^{(t)}$ and going in the clockwise² direction for $n - 1$
 130 time units will never be blocked by a missing edge, and thus will explore all vertices
 131 within those $n - 1$ time units.

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

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

141 *Sketch of proof.* Let u be some arbitrary vertex of the dynamic graph. For any integer
 142 $i \geq 0$, let V_i be the set of vertices reachable from u in at most i time units. We have that
 143 $V_i \subsetneq V_{i+1}$ until V_i contains all the vertices. Indeed, before all vertices are reachable,
 144 there exists a vertex not in V_i which is neighbor of a vertex in V_i , because the dynamic
 145 graph is constantly connected. \diamond

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

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

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

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

165 A *subtree* is a connected set consisting of *H-vertices* and *G-vertices*. A subtree is
 166 called *maximal* if the sets of *H-vertices* and *G-vertices* that it consists of cannot be

² The actual definition of the ‘‘clockwise’’ direction does not really matter as long as it is any fixed direction.

167 extended. A *graft* is a maximal subtree that does not contain two H -vertices belonging
 168 to the same cycle. Finally, a *block* is a graft or a cycle.

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

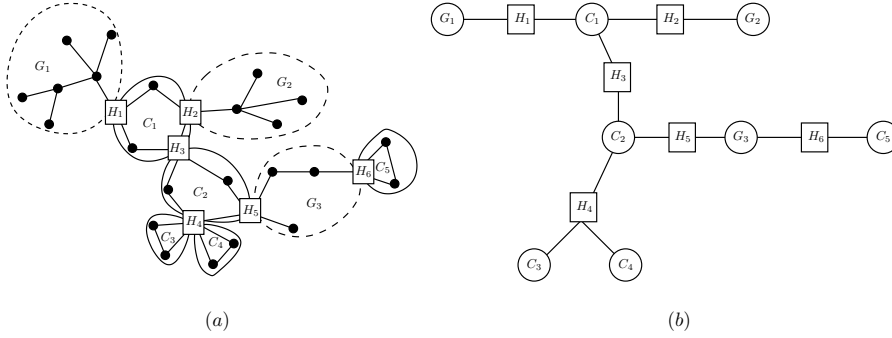


Fig. 2 Tree representation of a cactus.

171 If we add an edge between the blocks and the H -vertices, we obtain the tree
 172 $T_G = (V_G, E_G)$ such that each element of V_G is a block or an H -vertex. Figure 2.(b)
 173 gives the tree representation of the cactus shown in Figures 1 and 2.(a). We say that
 174 a cactus is *rooted* if the tree that represents it is rooted.

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

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

186 3 Chain method

187 In this section, we give a simple algorithm inspired by DFS to explore constantly
 188 connected dynamic graphs based on a plump cactus of n vertices. The principle of
 189 the algorithm is very simple. If the agent enters a cycle it has not visited yet, it visits
 190 it using the algorithm EXPLORE-RING for exploring dynamic graphs based on the
 191 ring (see Theorem 3), then passes to the attachment vertex of its closest unexplored
 192 child and explores it recursively. If all its children have already been explored and
 193 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

```

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

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

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

214 **4 Star method**

215 As we have seen in the previous section, the algorithm CHAIN-METHOD is not ef-
216 fective for exploring constantly connected dynamic graphs based on cactuses with
217 cycles of large degree, because the order of visit of the sub-cactuses is fixed, which
218 makes the algorithm spend a lot of time to go from one sub-cactus to the next. On the
219 contrary, the algorithm STAR-METHOD presented in this section focuses on reducing
220 these transit times on the root cycle (the cycle where the exploration starts) to the
221 minimum. The idea is to visit the sub-cactuses while exploring the root cycle. Note
222 that directly using the algorithm EXPLORE-RING on the root cycle does not work,
223 because when returning to the attachment vertex after exploring a sub-cactus, the
224 agent cannot continue the exploration according to the algorithm EXPLORE-RING on

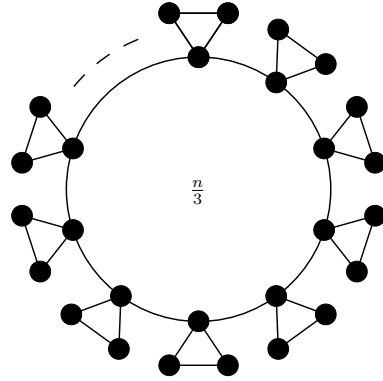


Fig. 3 Difficult graph for the CHAIN-METHOD.

225 the root cycle, as the dynamicity has changed on this cycle. To avoid this issue, the
 226 algorithm STAR-METHOD uses the algorithm EXPLORE-RING on a carefully chosen
 227 virtual dynamic cycle that takes into account both the dynamics of the root cycle and
 228 the time needed to recursively explore the sub-cactuses.

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

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

243 Let f_S be the recursive function defined in the statement of the theorem. The al-
 244 gorithm STAR-METHOD proceeds as follows. First, we introduce the following trans-
 245 formation of \mathcal{G} into another dynamic graph \mathcal{G}' , based on a ring $R_{n'}$ of size n' . The
 246 dynamic graph \mathcal{G}' is constructed as follows. We retain the root cycle of C and the
 247 dynamics of the graph \mathcal{G} on this cycle. We replace every H -vertex of C with two
 248 C -vertices linked by a sequence of static paths of length equal to the recursive cost
 249 of exploring each subtree attached to the H -vertex. More precisely, the lengths of the
 250 added paths are $f_S(C_i)$, for all the sub-cactuses C_i attached to the H -vertex. Thus,
 251 we obtain a constantly connected dynamic graph based on a ring of size n' (see Fig-
 252 ure 4). The dynamic graph \mathcal{G}' is indeed constantly connected because we retain the

253 dynamicity of the subgraph of \mathcal{G} based on the root cycle of C , which itself respects
 254 the constant connectivity.

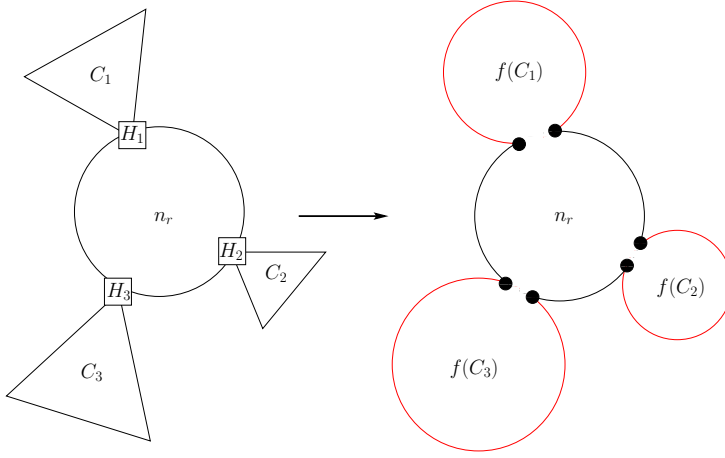


Fig. 4 Correspondence between the dynamic graph based on C and the dynamic graph based on $R_{n'}$.

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

262 Now let Agent B be the virtual agent, starting from the previously defined vertex
 263 $v^{(t)}$, that goes in the clockwise direction in \mathcal{G}' without being blocked for $n' - 1$
 264 time units (by Theorem 1). We define the Agent A following the STAR-METHOD as
 265 follows.

266 First Agent A uses $n_r - 1$ time units to reach the previously defined vertex v
 267 on C . This is possible thanks to the property from Theorem 2. Now, whenever the
 268 (virtual) Agent B stays on a subpath P corresponding to some sub-cactus C_i for at
 269 least $f_S(C_i)$ consecutive time units, Agent A uses this time to recursively explore the
 270 sub-cactus C_i . If, after completing this exploration, Agent B is still lying on P , then
 271 Agent A simply waits on the attachment vertex. Whenever Agent B lies on the part
 272 corresponding to the root cycle (that is outside of the added subpaths), Agent A be-
 273 haves exactly as Agent B .

274 This simulation of agent B on \mathcal{G} takes at most $n' - 1 = (n_r - 1) + \sum_{i=1}^{\ell} f_S(C_i)$
 275 time units. By construction, the root cycle and all sub-cactuses are explored, except
 276 for possibly one sub-cactus C_i , if the agent B starts (and ends) inside the static path
 277 corresponding to C_i . In this case, the agent A uses at most $f_S(C_i) \leq \max_{1 \leq j \leq \ell} f_S(C_j)$
 278 additional time units to explore C_i . Finally, in any case, the agent returns to the start-

ing 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. \square

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, Figure 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 $2^{\Omega(n)}n$.

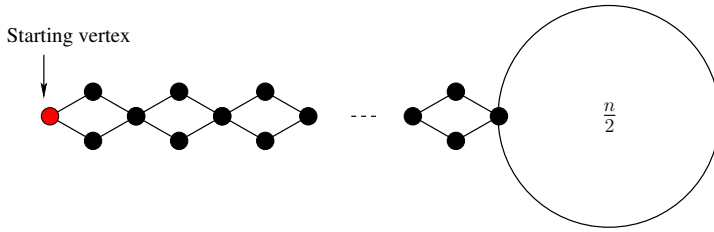


Fig. 5 Difficult graph for the STAR-METHOD.

5 Mixed method

In a plump cactus C with a root cycle of size n_r and with sub-cactuses C_1, \dots, C_ℓ , 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. 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_ℓ be its sub-cactuses, with $\ell \geq 1$. Assume without loss of generality that the sub-cactuses C_1, \dots, C_ℓ 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 .

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

317 The resulting cost is then $\sum_{i=1}^k (n_r - 1 + f_M(C_i)) + (3(n_r - 1) + \sum_{i=k+1}^{\ell} f_M(C_i) +$
 318 $\max_{k+1 \leq i \leq \ell} f_M(C_i))$. Using the fact that the costs $f_M(C_i)$ are decreasing, the preced-
 319 ing bound becomes $3(n_r - 1) + \sum_{i=1}^{\ell} f_M(C_i) + (k \cdot (n_r - 1) + f_M(C_{k+1}))$.³ The algo-
 320 rithm MIXED-METHOD chooses k such as to minimize the additional cost $k \cdot (n_r -$
 321 $1) + f_M(C_{k+1})$. To summarize, the exploration cost of the algorithm MIXED-METHOD
 322 is $f_M(C) = 3(n_r - 1) + \sum_{i=1}^{\ell} f_M(C_i) + \min_{1 \leq k \leq \ell} (k \cdot (n_r - 1) + f_M(C_{k+1}))$.

323 5.1 Upper bound for the algorithm MIXED-METHOD

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

329 **Theorem 6** *An agent executing the algorithm MIXED-METHOD requires at most*
 330 $O\left(n \frac{\log n}{\log \log n}\right)$ *time units to explore any constantly connected dynamic graph based*
 331 *on a plump cactus of n vertices.*

332 *Proof* Fix an arbitrary constantly connected dynamic graph based on a plump cac-
 333 tus C of n vertices. In order to study the exploration cost of the algorithm MIXED-
 334 METHOD, we will discuss another algorithm, denoted EXPLORE-CACTUS, which is
 335 less efficient but easier to analyze. The upper bound obtained for this less efficient
 336 algorithm will also give us a valid upper bound for the MIXED-METHOD. The algo-
 337 rithm EXPLORE-CACTUS is defined as the algorithm MIXED-METHOD except that
 338 the number k of sub-cactuses on which it uses the algorithm CHAIN-METHOD is al-
 339 ways $\min\{\ell, 2c\}$, with $c = \frac{\log n}{\log \log n}$.

340 Therefore, the exploration cost $f_E(C)$ of the algorithm EXPLORE-CACTUS in a
 341 plump cactus C having a root cycle of size n_r and sub-cactuses C_1, \dots, C_{ℓ} is at most
 342 $3(n_r - 1) + \sum_{i=1}^{\ell} f_E(C_i) + 2c(n_r - 1) + f_E(C_{2c+1})$. Among the first $2c$ sub-cactuses,
 343 there are at least c sub-cactuses whose number of vertices is at most n/c , where n is
 344 the total number of vertices in C . Also, all these sub-cactuses have a cost larger than
 345 $f_E(C_{2c+1})$. It is therefore possible to charge the additional cost $f_E(C_{2c+1})$ to these
 346 sub-cactuses. We obtain the upper bound $f_E(C) \leq 3(n_r - 1) + \sum_{i=1}^{\ell} f_E(C_i) + 2c(n_r -$
 347 $1) + \frac{1}{c} \sum_{|C_i| \leq \frac{n}{c}} f_E(C_i)$.

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

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

354 5.2 Lower bound for the algorithm MIXED-METHOD

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

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

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

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

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

372 First, let $C_{d,0}$ be a ring of $m_{d,0} = r_{d,0} = (d + 3)/3$ edges (which is an integer by
 373 definition of h). The exploration cost of this cactus is $t_{d,0} = f_M(C_{d,0}) = 3(r_{d,0} - 1) =$
 374 d . For $i \geq 1$, we define inductively $C_{d,i}$ as a cactus with a root cycle of size $r_{d,i} =$
 375 $t_{d,i-1} + 1$ on which are attached on d different vertices the cactuses $\text{sub}_1(C_{d,i-1})$,
 376 $\text{sub}_2(C_{d,i-1})$, up to $\text{sub}_d(C_{d,i-1})$ (see Figure 6). Finally, C_d is defined as $C_{d,h}$.

377 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 al-
 378 gorithm MIXED-METHOD chooses the number k of sub-cactuses to explore with the
 379 algorithm CHAIN-METHOD such as to minimize the additional cost $k \cdot (r_{d,i} - 1) +$
 380 $f_M(\text{sub}_{d-k}(C_{d,i-1}))$. On one hand, we have $r_{d,i} - 1 = t_{d,i-1}$ by definition. On the
 381 other hand, the recursive exploration cost of each sub-cactus $\text{sub}_j(C_{d,i-1})$ is $j \cdot t_{d,i-1}$.
 382 Therefore, the additional cost does not depend on k and is always equal to $d \cdot t_{d,i-1}$. In
 383 other words, the cactus is constructed in such a way that all the algorithms CHAIN-
 384 METHOD, STAR-METHOD, and MIXED-METHOD have the same exploration cost.
 385 Therefore, the exploration cost $f_M(C_{d,i})$ of $C_{d,i}$ is $t_{d,i} = 3(r_{d,i} - 1) + \sum_{j=1}^d (j \cdot t_{d,i-1}) +$
 386 $d \cdot t_{d,i-1} = (3 + d(d + 1)/2 + d) \cdot t_{d,i-1} = ((d^2 + 3d + 6)/2) \cdot t_{d,i-1}$.

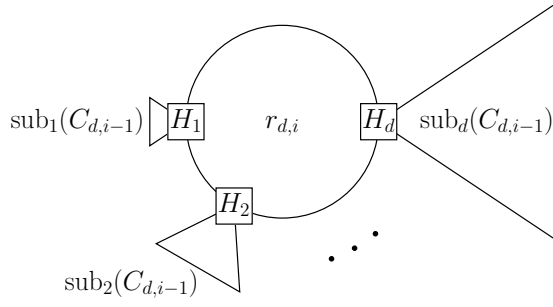


Fig. 6 Inductive construction of the cactus $C_{d,i}$.

387 To simplify the notation, we now remove the first index d . Also, let $\alpha = (d^2 +$
 388 $3d + 6)/2$ and $\beta = d(d + 1)/2$. To summarize, we have

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

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

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

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

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

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

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

392 Besides, we have

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

393 Plugging the last equation into the previous one, we obtain

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

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

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

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

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

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

399 The inequality $2^d \leq n$ implies that $\log d \leq \log \log n$, while $d^{6h} = d^{3d \log d} \geq n$ allows
400 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 $t_h \geq$

401 $\frac{1}{18} n \frac{\log n}{(\log \log n)^2}$, which concludes the proof of the theorem. \square

402 6 Conclusion

403 In this paper, we studied the time complexity for exploring constantly connected dy-
404 namic graphs based on cactuses, under the assumption that the agent knows the dy-
405 namics of the graph. We gave an exploration algorithm for dynamic graphs that we
406 called MIXED-METHOD, and we have shown that for exploring the whole class of
407 constantly connected dynamic graphs based on cactuses of n vertices, with this algo-
408 rithm, $\Omega\left(n \frac{\log n}{(\log \log n)^2}\right)$ time units are necessary (for infinitely many n), and $O\left(n \frac{\log n}{\log \log n}\right)$
409 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. E. Aaron, D. Krizanc, and E. Meyerson. DMVP: Foremost Waypoint Coverage of Time-Varying Graphs. In *40th International Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, LNCS 8147, pages 29–41, 2014.
2. E. Aaron, D. Krizanc, and E. Meyerson. 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, pages 22–38, 2014.
3. A. Agarwalla, J. Augustine, W. K. Moses Jr., S. Madhav K. and A. K. Sridhar. Deterministic Dispersion of Mobile Robots in Dynamic Rings. In *19th International Conference on Distributed Computing and Networking, ICDCN 2018*, pages 19:1–19:4, 2018.
4. H. L. Bodlaender and T. C. van der Zanden. On exploring always-connected temporal graphs of small pathwidth. In *Information Processing Letters*, volume 142, pages 68–71, 2019.
5. M. Bournat, S. Dubois and F. Petit. Computability of Perpetual Exploration in Highly Dynamic Rings. In *37th IEEE International Conference on Distributed Computing Systems (ICDCS)*, IEEE Computer Society, pages 794–804, 2017.
6. R. Burkard and J. Krarup. A Linear Algorithm for the Pos/Neg-Weighted 1-Median Problem on a Cactus. In *Computing*, volume 60(3), pages 193–216, 1998.
7. A. Casteigts, P. Flocchini, W. Quattrociocchi, and N. Santoro. Time-varying graphs and dynamic networks. In *International Journal of Parallel, Emergent and Distributed Systems*, volume 27(5), pages 387–408, 2012.
8. S. Das, G. A. Di Luna and L. A. Gasieniec. Patrolling on Dynamic Ring Networks. In *45th International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM)*, LNCS 11376, pages 150–163, 2019.
9. G. A. Di Luna, S. Dobrev, P. Flocchini, and N. Santoro. Distributed exploration of dynamic rings. In *Distributed Computing.*, volume 33(1), pages 41–67, 2020.
10. G. A. Di Luna, P. Flocchini, L. Pagli, G. Prencipe, N. Santoro and G. Viglietta. Gathering in dynamic rings. In *Theoretical Computer Science*, volume 811, pages 79–98, 2020.
11. C. Dutta, G. Pandurangan, R. Rajaraman, Z. Sun and E. Viola. On the Complexity of Information Spreading in Dynamic Networks. *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 717–736, 2013.
12. T. Erlebach, M. Hoffmann, and F. Kammer. On Temporal Graph Exploration. In *42nd International Colloquium on Automata, Languages, and Programming (ICALP)*, LNCS 9134, pages 444–455, 2015.
13. T. Erlebach, J. T. Spooner. Faster Exploration of Degree-Bounded Temporal Graphs. In *43rd International Symposium on Mathematical Foundations of Computer Science (MFCS)*, page 36:1–36:13, 2018.

- 460 14. A. Ferreira. Building a Reference Combinatorial Model for Dynamic Networks: Initial Results in
461 Evolving Graphs. *INRIA*, RR-5041 (2003)
- 462 15. T. Gotoh, P. Flocchini, T. Masuzawa and Nicola Santoro Tight Bounds on Distributed Exploration of
463 Temporal Graphs In *23rd International Conference on Principles of Distributed Systems (OPODIS)*,
464 volume 153 pages 22:1–22:16, 2019.
- 465 16. T. Gotoh, Y. Sudo, F. Ooshita, H. Kakugawa and T. Masuzawa. Group Exploration of Dynamic Tori.
466 In *38th IEEE International Conference on Distributed Computing Systems (ICDCS)*, IEEE Computer
467 Society, pages 775–785, 2018.
- 468 17. T. Gotoh, Y. Sudo, F. Ooshita and T. Masuzawa Dynamic Ring Exploration with (H,S) View In
469 *Algorithms*, volume 13(6), pages 141, 2020.
- 470 18. D. Ilcinkas, R. Klasing, and A. M. Wade. Exploration of Constantly Connected Dynamic Graphs
471 Based on Cactuses. In *21st International Colloquium on Structural Information and Communication
472 Complexity (SIROCCO)*, LNCS 8576, pages 250–262, 2014.
- 473 19. D. Ilcinkas and A.M. Wade. Exploration of the T -Interval-Connected Dynamic Graphs: the Case of
474 the Ring. In *Theory of Computing Systems*, volume 62, pages 1144–1160, 2018.
- 475 20. F. Kuhn, N.A. Lynch, and R. Oshman: Distributed computation in dynamic networks. In *42nd ACM
476 Symposium on Theory of Computing (STOC)*, pages 513–522, 2010.
- 477 21. F. Kuhn and R. Oshman. Dynamic networks: models and algorithms. In *ACM SIGACT News*, volume
478 42(1), pages 82–96, 2011.
- 479 22. O. Michail. An Introduction to Temporal Graphs: An Algorithmic Perspective. In *Internet Mathe-
480 matics*, volume 12(4), pages 239–280, 2016.
- 481 23. O. Michail and P. G. Spirakis. Traveling salesman problems in temporal graphs. In *Theoretical
482 Computer science*, volume 634, pages 1–23, 2016.
- 483 24. T. Mömke and O. Svensson. Removing and Adding Edges for the Traveling Salesman Problem. J. In
484 *Journal of the ACM*, volume 63(1), pages 2:1–2:28, 2016.
- 485 25. M. Mucha. 13/9-Approximation for Graphic TSP. In *Theory of Computing Systems*, volume 55(4),
486 pages 640–657, 2014.
- 487 26. R. O’Dell and R. Wattenhofer. Information dissemination in highly dynamic graphs. In *DIALM-
488 POMC*, pages 104–110, 2005.
- 489 27. A. Sebö, J. Vögen. Shorter tours by nicer ears: $7/5$ -Approximation for the graph-TSP, $3/2$ for the path
490 version, and $4/3$ for two-edge-connected subgraphs. In *Combinatorica*, volume 34(5) pages 597–629,
491 2014.
- 492 28. C.E. Shannon. Presentation of a maze-solving machine. In *8th Conf. of the Josiah Macy Jr. Found.
493 (Cybernetics)*, pages 173–180, 1951.