

Exploration of carrier-based time-varying networks: the power of waiting[☆]

David Ilcinkas^{a,1}, Ahmed M. Wade^{b,2}

^a*LaBRI, CNRS, Univ. Bordeaux, France*

^b*LTISI, École Polytechnique de Thiès, Senegal*

Abstract

We study the problem of exploration by a mobile entity (agent) of a class of highly dynamic networks, namely the carrier graphs (the C-graphs, modeling public transportation systems, among others). These are defined by a set of carriers following infinitely their prescribed route along the stations of the network. Flocchini, Mans, and Santoro [9] studied this problem in the case when the agent must always travel on the carriers and thus cannot wait on a station. They described the necessary and sufficient conditions for the problem to be solvable and proved that the optimal worst-case number of time units (and thus of moves) to explore a n -node C-graph of k carriers and maximal period p is in $\Theta(kp^2)$ in the general case.

In this paper, we study the impact of the ability to wait at the stations. We exhibit the necessary and sufficient conditions for the problem to be solvable in this context, and we prove that waiting at the stations allows the agent to reduce the optimal worst-case number of moves by a multiplicative factor of at least $\Theta(p)$, while the worst-case time complexity is reduced to $\Theta(np)$. (In any connected carrier graph, we have $n \leq kp$.) We also show some complementary optimal results in specific cases (same period for all carriers, highly connected C-graphs). Finally this new ability allows the agent to completely map the C-graph, in addition to just exploring it.

[☆]A preliminary version of this paper appeared in the Proceedings of the 15th International Conference On Principles Of Distributed Systems (OPODIS 2011) [10].

¹Partially supported by the ANR project DESCARTES (ANR-16-CE40-0023), and the “Investments for the future” Programme IdEx Bordeaux – CPU (ANR-10-IDEX-03-02).

²Partially supported by the African Center of Excellence in Mathematics, Computer Science and ICT (CEA-MITIC)

1 **1. Introduction**

2 *1.1. The problem*

3 The problem of graph exploration consists, for a mobile entity, in explor-
4 ing all nodes (or edges) of an a priori unknown graph. This problem being
5 one of the most classical in the mobile agent computing framework, it has re-
6 ceived a lot of attention so far. Time complexity, space complexity, or impact
7 of a priori knowledge have extensively been studied in the last 40 years (see,
8 e.g., [5, 13, 14]). However, the large majority of these works concern static
9 graphs. Considering networks nowadays, it is now common to deal with dy-
10 namic networks. In this paper, we study the graph exploration problem in
11 a particular class of time-varying graphs namely the carrier graph (C-graph)
12 model (the C-graphs were called PV-graphs in the first papers concerning
13 them).

14 Roughly speaking, a C-graph consists of a set of carriers, each following
15 periodically its respective route among the sites of the system. This models
16 in particular various types of public transportation systems like bus systems
17 or subway systems for example. It also models low earth orbiting satellite
18 systems, or security systems composed of security guards making tours in the
19 place to be secured. Performing exploration in such systems may be useful
20 for maintenance operations for example. Indeed, an agent can check that
21 everything is in order during the exploration. This agent may be a piece of
22 software, or a human being.

23 The exploration problem in the C-graph model was already considered
24 by Flocchini, Mans, and Santoro in [9]. They considered that the agent
25 cannot leave the carrier to stay on a site. Not being able to stay on a site
26 is particularly legitimate in low earth orbiting satellite systems for example,
27 where the sites do not correspond to any physical station. However, in most
28 public transportation systems, it is possible for the agent (human or not)
29 to stay on a site in order to wait for a (possibly different) carrier. In this
30 paper, we consider the same problem but in the case when the agent can
31 leave carriers to wait on a site. We study the impact of this new ability
32 on the worst-case complexity (time and number of moves) of the C-graph
33 exploration problem.

34 *1.2. Related work*

35 Motivated by the automatic exploration of the Web, Cooper and Frieze [4]
36 studied the question of the minimum cover time of a graph that evolves over
37 time. They considered a particular model of so-called web graphs and showed
38 that if after every constant number of steps of the walk a new node appears
39 and is connected to the graph, a random walk does not visit a constant
40 fraction of nodes. Avin, Koucky and Lotker [1] showed that a random walk
41 may have an exponential cover time in some dynamic graphs. They also
42 show that a variant, the lazy random walk, has however a polynomial cover
43 time in any dynamic graph.

44 To investigate distributed computations in dynamic networks, Kuhn,
45 Lynch and Oshman [12] introduced a new stability property called T -interval-
46 connectivity, for a given positive integer T . This property ensures that for
47 any T consecutive rounds, there is a stable and connected common subgraph.
48 Considering this stability property, Ilcinkas and Wade studied the complex-
49 ity of the exploration of dynamic rings by a mobile agent [11]. The same
50 stability property, with $T = 1$, is also considered by Di Luna, Dobrev, Floc-
51 chini, and Santoro in [6] to study the decentralized (or live) exploration of a
52 dynamic ring by a team of agents.

53 Casteigts, Flocchini, Santoro and Quattrociocchi [3] integrated a large
54 collection of concepts, formalisms and results in the literature about dynamic
55 graphs in an unified space called time-varying graphs. Flocchini, Mans and
56 Santoro [9] introduced a specific class of time-varying graphs, the C-graph
57 model. They first show that if the nodes of the C-graph are labeled, the
58 knowledge of an upper bound on the longest period or the exact knowledge
59 of the number n of nodes is necessary and sufficient for an agent to explore
60 the C-graph. If the nodes of the C-graph are anonymous, then the knowledge
61 of an upper bound on the longest period is necessary and sufficient. In both
62 settings, the worst-case time and move complexity of the agent is proved to
63 be in $\Theta(kp^2)$, where k is the number of carriers and p the maximum period
64 of the carriers. In the particular case of homogeneous C-graphs (C-graphs
65 for which all carriers have the same period), the worst-case time and move
66 complexity drops to $\Theta(kp)$.

67 Using a C-graph to model an urban subway system with black holes (sites
68 destroying agents), Flocchini, Kellett, Mason, and Santoro [7, 8] examined
69 the problem of constructing a map of such a subway. They considered that
70 several agents are operating in the C-graph, and that they can leave messages
71 on the sites. The goal of the agents is to construct the map of the C-graph

72 without losing too many agents. The class of C-graphs is also used in [2],
 73 where the authors consider oblivious carriers and investigate the routing
 74 problem.

75 *1.3. Our results*

76 In this article, we extend the study of Flocchini, Mans and Santoro [9]
 77 to the case when the agent can leave a carrier to stay at a site. This new
 78 ability allows the agent to explore C-graphs that are less connected over
 79 time (formal definitions are given in Section 2). We prove that in the general
 80 case (so, even considering non highly-connected C-graphs) the worst-case
 81 move complexity is reduced to $\Theta(\min\{kp, np, n^2\})$, while the worst-case time
 82 complexity decreases to $\Theta(np)$. (Note that in any connected C-graph, we
 83 have $n \leq kp$.) If the C-graphs are restricted to be both homogeneous and
 84 highly-connected, then Flocchini, Mans and Santoro proved that the worst-
 85 case time complexity is in $O(kp)$. In this paper, we prove that if the C-graphs
 86 satisfy only one of these restrictions, then the worst-case time complexity
 87 remains in $\Theta(np)$. Besides, it turns out that our algorithm not only performs
 88 exploration but also performs mapping, i.e., it can output an isomorphic
 89 copy of the C-graph. Finally, note that our algorithm does not use possible
 90 identifiers of the nodes, while all our lower bounds still hold when the agent
 91 has access to unique node identifiers.

Results from [9]	Connected	Highly-connected
Our results		
Not necessarily homogeneous	Impossible $\Theta(\min\{kp, np, n^2\})$ moves $\Theta(np)$ time units	$\Theta(kp^2)$ moves & time units $\Theta(\min\{kp, np, n^2\})$ moves $\Theta(np)$ time units
Homogeneous	Impossible $\Theta(\min\{kp, np, n^2\})$ moves $\Theta(np)$ time units	$\Theta(kp)$ moves & time units $\Theta(\min\{kp, np, n^2\})$ moves $O(np)$ time units

Table 1: Comparison of our results (bottom of each cell, in red color) with the results obtained in [9] (top of each cell, in blue color). All mentioned complexities are asymptotic worst-case complexities.

92 **2. Model and definitions**

93 We consider a system $S = \{s_1, \dots, s_n\}$ of n sites among which k carriers
 94 are moving. Each carrier c has an identifier $\text{Id}(c)$ and follows a finite sequence

95 $R(c) = (s_{i_1}, \dots, s_{i_{p(c)}})$ of sites, called its *route*, in a periodic manner. The
 96 positive integer $p(c)$ is called the *period* of the carrier c . More precisely,
 97 the carrier c starts at node s_{i_1} at time 0 and then proceeds along its route,
 98 moving to the next site at each time unit, in a cyclic manner (that is, when
 99 c is at node $s_{i_{p(c)}}$, it goes back to s_{i_1} and follows the route again and again).

100 A C-graph (for carrier graph) is a pair (S, C) , where S is a set of sites,
 101 and C is a set of carriers operating among these sites. We will usually denote
 102 by n , k and p , respectively, the number of sites, the number of carriers and
 103 the maximum over the periods of the carriers. A C-graph is said to be
 104 *homogeneous* if and only if all its carriers have the same period.

105 For any C-graph G , we define two (classical) graphs $H_1(G)$ and $H_2(G)$ as
 106 follows. Both graphs have the set of carriers as the set of nodes. There is an
 107 edge in $H_1(G)$ between two carriers c and c' if and only if there exists a site
 108 appearing in both the routes of c and c' . There is an edge in $H_2(G)$ between
 109 two carriers c and c' if and only if there exists a site s and a time $t \geq 0$ such
 110 that c and c' are both in s at time t . A C-graph is said to be *connected* if
 111 and only if $H_1(G)$ is connected. A C-graph is said to be *highly-connected*
 112 if and only if $H_2(G)$ is connected. In this paper, we will always consider
 113 C-graphs that are at least connected. (Non-connected C-graphs cannot be
 114 explored by a single agent.) Furthermore note that, for any connected C-
 115 graph, its parameters n (number of sites), k (number of carriers), and p
 116 (maximal period) satisfy the inequality $n \leq p + (k - 1)(p - 1)$. Indeed, if one
 117 adds the carriers one by one to the C-graph in such a way that the growing
 118 C-graph is always connected, the first carrier has at most p sites, and any
 119 subsequent carrier introduces at most $p - 1$ new sites (because one of its sites
 120 must be common with the C-graph constructed so far). This leads to the
 121 claimed upper bound on the number n of sites.

122 An entity, called *agent*, is operating on these C-graphs. It can see the
 123 carriers and their identifiers. It can ride on a carrier to go from a site to
 124 another. Contrary to the model in [9], the agent is allowed to leave a carrier,
 125 stay at the current site, and get back on a carrier (the same or another).
 126 We do not assume any restriction on the memory size of the agent or on its
 127 computational capabilities. We consider two models concerning the nodes'
 128 identities. In an *anonymous* C-graph, the nodes do not have any identities,
 129 or the agent is not able to see them. In a *labeled* C-graph, the nodes have
 130 distinct identities and the agent can see and memorize them.

131 We say that an agent *explores* a C-graph if and only if, starting at time 0
 132 on the starting site of the first carrier (this can be assumed without loss of

133 generality), the agent eventually visits all sites of the C-graph and switches
 134 afterwards to a terminal state. This terminal state expresses the fact that
 135 the agent knows that exploration has been completed.

136 3. Solvability

137 Similarly as in the case when the agent cannot wait, an agent without
 138 information on the C-graphs it has to explore cannot explore all C-graphs
 139 (even if restricted to the labeled homogeneous highly-connected ones).

140 **Theorem 1.** *There exists a family of labeled homogeneous highly-connected*
 141 *C-graphs such that no agent can explore all the graphs of this family if it has*
 142 *no information on the C-graphs it has to explore.*

143 *Sketch of proof.* Intuitively, the family consists of a small C-graph G_0 and
 144 an infinity of C-graphs “looking like” G_0 for an arbitrarily large time. The
 145 agent must enter to a terminal state in a finite time t after completing the
 146 exploration of G_0 . It is possible to prove that there is a C-graph of the family
 147 that the agent will not be able to differentiate from G_0 until time $t + 1$ and
 148 that has one more site, which will never be explored by the agent.

149 **PROOF.** Let $S = \{s_1, s_2, s_3\}$ be a set of three sites with distinct IDs. For
 150 $t > 0$, we define the C-graph G_t over the set S of sites composed of a single
 151 carrier. Its route is $(s_1, s_2, \dots, s_1, s_2, s_1, s_2, s_3)$, where (s_1, s_2) is repeated
 152 exactly t times. Moreover, let G_0 be the C-graph over the set of sites $\{s_1, s_2\}$
 153 composed of a single carrier, whose route is (s_1, s_2) . The family $\{G_0, G_1, \dots\}$
 154 is denoted \mathcal{G} , see Figure 1.

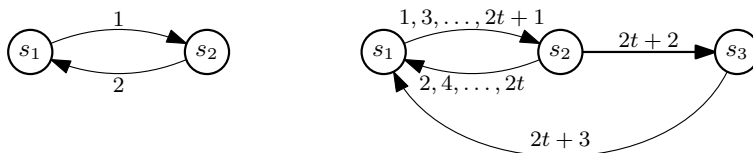


Figure 1: The C-graphs G_0 and G_t of the family \mathcal{G} .

155 Assume, for the purpose of contradiction, that there exists an algorithm
 156 solving the exploration problem in all the C-graphs in \mathcal{G} , provided that the

157 agent A running this algorithm does not receive any additional information.
 158 In particular, A explores G_0 . Let t be the time at which A switches to the
 159 terminal state. Assume now that A is placed in G_t . For the first t time
 160 units, A cannot tell the difference between G_0 and G_t , because A has no
 161 information about the C-graph it has to explore and in particular it does not
 162 know the number of sites or an upper bound on the system period. It will
 163 therefore act exactly the same in G_t as in G_0 . In particular, it will switch to
 164 the terminal state at time t although the site s_3 has not yet been explored.
 165 This contradiction concludes the proof. \square

166 4. General case

167 In this section, we make no assumption on the C-graphs (except the
 168 connectedness assumption of course). We basically show that the ability to
 169 wait allows the agent to explore, and even map, all connected C-graphs (not
 170 only the highly-connected ones), provided that the agent knows for each of
 171 them an upper bound on its maximal period. This can be done in only
 172 $\Theta(\min\{kp, np, n^2\})$ moves in the worst case, that is, at least p times less
 173 than when the agent cannot wait. Besides, the worst-case time complexity
 174 is reduced from $\Theta(kp^2)$ to $\Theta(np)$.

175 4.1. Lower bound on the number of moves

176 Flocchini, Mans and Santoro [9] proved a lower bound $\Omega(kp)$ on the num-
 177 ber of moves to explore the C-graphs with k carriers and maximum period p
 178 (even if restricted to the labeled homogeneous highly-connected ones). This
 179 lower bound does not apply directly in our setting because the agent, having
 180 the possibility to wait, could potentially be able to explore in significantly
 181 less moves. We will prove later that this is actually the case: the move com-
 182 plexity of our algorithm is bounded by $O(\min\{kp, np, n^2\})$. We prove here
 183 that this complexity is optimal.

184 **Lemma 1.** *For any integers n, k, p such that $n \leq p + (k - 1)(p - 1)$ (neces-
 185 sary for connectedness), there exists a labeled homogeneous highly-connected
 186 C-graph $G_{n,k,p}$ with n sites, k carriers and period p such that any algorithm
 187 needs at least $\min\{kp - 1, \lfloor \frac{n}{8} \rfloor p - 1, \frac{7n}{8} (\lfloor \frac{n}{8} \rfloor - 1)\}$ moves to explore it.*

188 *Sketch of proof.* For any feasible choice of the parameters n , k , and p , we
 189 construct a C-graph in a way that forces some sites to be visited many times
 190 in order to visit the other sites. Different constructions are used according
 191 to the relative values of the different parameters, yielding the different terms
 192 of the minimum.

193 **PROOF.** Fix any integers $n \geq 16$, k , and p such that $n \leq p + (k - 1)(p - 1)$.
 194 (If $n < 16$, then the third term of the minimum is obviously a lower bound.)
 195 We consider two cases.

196 Case 1: $p \leq \frac{3n^2}{16k}$.

197 Let us first assume that $k \leq n/8$ and let $q = \lfloor \frac{n}{2k} \rfloor$. Note that $p/2 \geq q \geq 4$.
 198 We denote by r the non-negative integer $\lceil p/q \rceil q - p$. Let $S = \{s_1, s_2, \dots, s_n\}$
 199 be a set of n sites. We partition S into the sets S_0 and $S_{i,j}$, with $1 \leq i \leq k$
 200 and $1 \leq j \leq q$, such that:

- 201 • $S_0 = \{s_1, s_2, \dots, s_{\lceil p/q \rceil - 1}\}$ and $S_{1,1} = \{s_{\lceil p/q \rceil}\}$;
- 202 • For all $1 \leq i \leq k$ and $1 \leq j \leq q$, we have $S_{i,j} \neq \emptyset$;
- 203 • For all $2 \leq i \leq k$, we have $|S_{i,1}| \leq \lceil p/q \rceil - 1$;
- 204 • For all $1 \leq i \leq k$ and $2 \leq j \leq q - r$, we have $|S_{i,j}| \leq \lceil p/q \rceil$;
- 205 • For all $1 \leq i \leq k$ and $q - r < j \leq q$, we have $|S_{i,j}| \leq \lceil p/q \rceil - 1$;
- 206 • While respecting the previous bounds on the size of the sets $S_{i,j}$,
 207 we have that if some $S_{i,j}$ has not its maximum, resp. minimum,
 208 allowed size, then all sets $S_{i',j'}$, with (i', j') lexicographically larger, resp.
 209 smaller, than (i, j) , have their minimum, resp. maximum, allowed size.

210 Such a partition is always possible, for the following reasons. First, p and
 211 k being fixed, the maximum number of sites permitting the construction is
 212 obtained when all sets $S_{i,j}$ have their maximum allowed size. This leads to
 213 the inequality

$$n \leq (\lceil p/q \rceil - 1) + 1 + (k - 1)(\lceil p/q \rceil - 1) + k(q - 1 - r)\lceil p/q \rceil + kr(\lceil p/q \rceil - 1)$$

214 which is equivalent to the connectivity condition $n \leq p + (k - 1)(p - 1)$, by
 215 definition of r . Second, still with p and k fixed, the minimum number of sites
 216 permitting the construction is obtained when all sets $S_{i,j}$ have size 1. This
 217 leads to the inequality

$$n \geq (\lceil p/q \rceil - 1) + kq$$

218 which is implied by the condition $p \leq \frac{3n^2}{16k}$. Indeed, this condition implies that
 219 $p \leq \frac{n}{4k}(n-2k) = \frac{n}{2}(\frac{n}{2k} - 1)$, because $k \leq n/8$. We thus have $p \leq \frac{n}{2} \lfloor \frac{n}{2k} \rfloor = \frac{n}{2}q$,
 220 by definition of q . Rearranging the inequality, we obtain $\frac{n}{2} + 1 \geq \frac{p}{q} + 1 \geq \lceil \frac{p}{q} \rceil$.
 221 Finally, we obtain $n \geq (\lceil \frac{p}{q} \rceil - 1) + k\frac{n}{2k} \geq (\lceil \frac{p}{q} \rceil - 1) + kq$, as desired.

222 The C-graph $G_{n,k,p}$ is now defined as follows, see Fig. 2. Let S be its
 223 set of sites and $C = \{c_1, c_2, \dots, c_k\}$ be the set of its carriers. For every
 224 $1 \leq i \leq k$, the route $R(c_i)$ is defined as follows. The route starts at s_1 at
 225 time 0 and then visits s_2, s_3, \dots, s_l , with $l = \lceil p/q \rceil - |S_{i,1}|$, followed by each
 226 site of the set $S_{i,1}$. (When $l = 1$, the route goes directly from s_1 to the sites
 227 of $S_{i,1}$.) The route continues by visiting, for successive values of j from 2
 228 to q , the sites s_1, s_2, \dots, s_l , with $l = \lceil p/q \rceil - |S_{i,j}|$ (or $l = \lceil p/q \rceil - 1 - |S_{i,j}|$
 229 if $j > q - r$), followed by each site of the set $S_{i,j}$. (When $l = 0$, the route
 230 directly continues to $S_{i,j}$, without going through any site in S_0 .) Note that
 231 $G_{n,k,p}$ is both homogeneous (of period p) and highly-connected (because s_1
 232 is the starting site of all routes).

233 The C-graph $G_{n,k,p}$ is constructed in such a way that the agent basically
 234 has to follow each carrier's route entirely to visit all sites. More precisely, to
 235 visit the sites of any set $S_{i,j}$ and to come back to s_1 , the agent has to pay
 236 $\lceil p/q \rceil$ moves ($\lceil p/q \rceil - 1$ if $j > q - r$). Hence the minimum number of moves
 237 an exploring agent has to perform in $G_{n,k,p}$ is $kp - 1$.

238 Now assume that $k > n/8$. In this case, we simply use the above con-
 239 struction for $\lfloor n/8 \rfloor$ carriers. All carriers c_i , with $i > \lfloor n/8 \rfloor$ are given the
 240 same route as c_1 . This gives the lower bound $\lfloor n/8 \rfloor p - 1$.

241 Case 2: $p > \frac{3n^2}{16k}$.

242 First assume that $k \leq n/16$. The C-graph $G_{n,k,p}$ is defined in this case
 243 as follows, see Fig. 3. Let $C = \{c_1, c_2, \dots, c_k\}$ be the set of its carriers
 244 and let $S = \{s_1, s_2, \dots, s_n\}$ be the set of its sites, partitioned in $S_0 =$
 245 $\{s_1, s_2, \dots, s_{n-\lfloor n/8 \rfloor}\}$ and $S_1 = S \setminus S_0$. The set S_1 is further partitioned
 246 into the sets $S_{1,1}$ to $S_{1,k}$ of weakly increasing size such that the sizes of any
 247 two sets differs by at most one (differently speaking, the size of a set $S_{1,i}$
 248 is either $\lfloor |S_1|/k \rfloor$ or $\lceil |S_1|/k \rceil$). If $S_{1,i} = \{s_{i_1}, s_{i_2}, \dots, s_{i_l}\}$ for some l , then
 249 the route $R(c_i)$ is the route starting at s_1 at time 0, visiting the sequences
 250 $(s_1, s_2, \dots, s_{n-\lfloor n/8 \rfloor}, s_{i_1}), (s_1, s_2, \dots, s_{n-\lfloor n/8 \rfloor}, s_{i_2}),$ up to $(s_1, s_2, \dots, s_{n-\lfloor n/8 \rfloor}, s_{i_l})$.
 251 It finally stays in s_{i_l} so that its period is exactly p . In order for the construc-
 252 tion to be possible, the maximum period p must satisfy

$$p \geq (n - \lfloor n/8 \rfloor + 1) \lceil \lfloor n/8 \rfloor / k \rceil$$

253 which is implied by the condition $p > \frac{3n^2}{16k}$ when $k \leq n/16$. Note that $G_{n,k,p}$

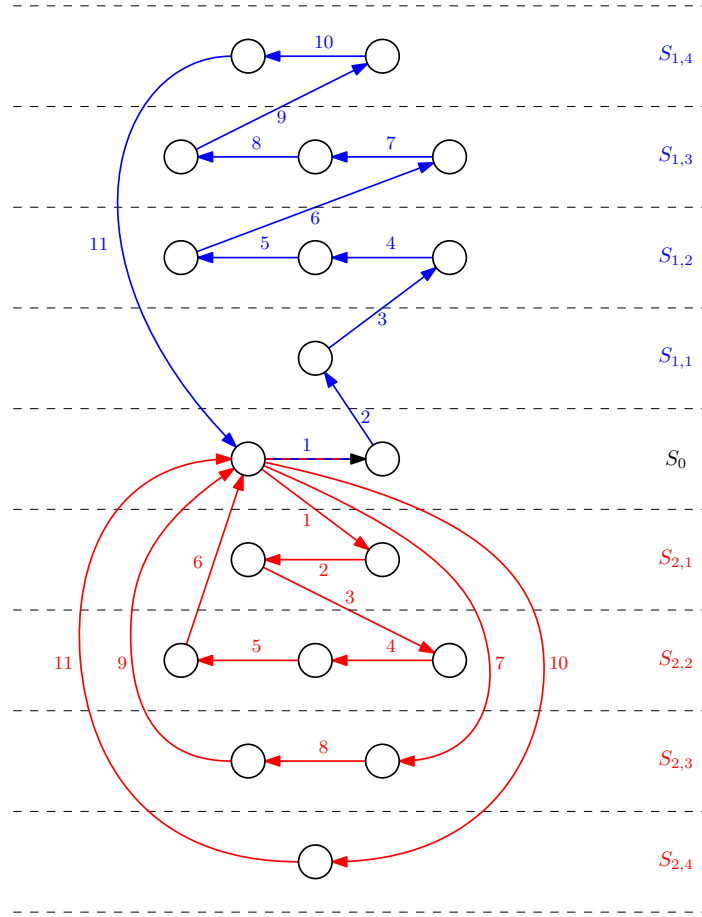


Figure 2: The C-graph $G_{n,k,p}$ used in Case 1 of the proof of Lemma 1, leading to the lower bound $\Omega(\min\{kp, np\})$. Here $n = 19$, $k = 2$, and $p = 11$.

254 is both homogeneous (of period p) and highly-connected (because s_1 is the
 255 starting site of all routes).

256 By construction, all sites in S_1 are only accessible through $s_{n-\lfloor n/8 \rfloor}$ and
 257 the agent can only leave them by going to s_1 with some carrier. Again by
 258 construction, any agent willing to go from s_1 to $s_{n-\lfloor n/8 \rfloor}$ has to go through
 259 all the sites $s_1, s_2, \dots, s_{n-\lfloor n/8 \rfloor}$. Therefore, for any i, j such that $1 \leq i \neq j \leq$
 260 $\lfloor n/8 \rfloor$, going from s_{n-i+1} to s_{n-j+1} requires any agent to perform at least
 261 $n - \lfloor n/8 \rfloor + 1$ moves. Since any agent performing exploration of the C-graph
 262 must visit all its sites, any agent requires at least $(n - \lfloor n/8 \rfloor + 1)(\lfloor n/8 \rfloor - 1)$

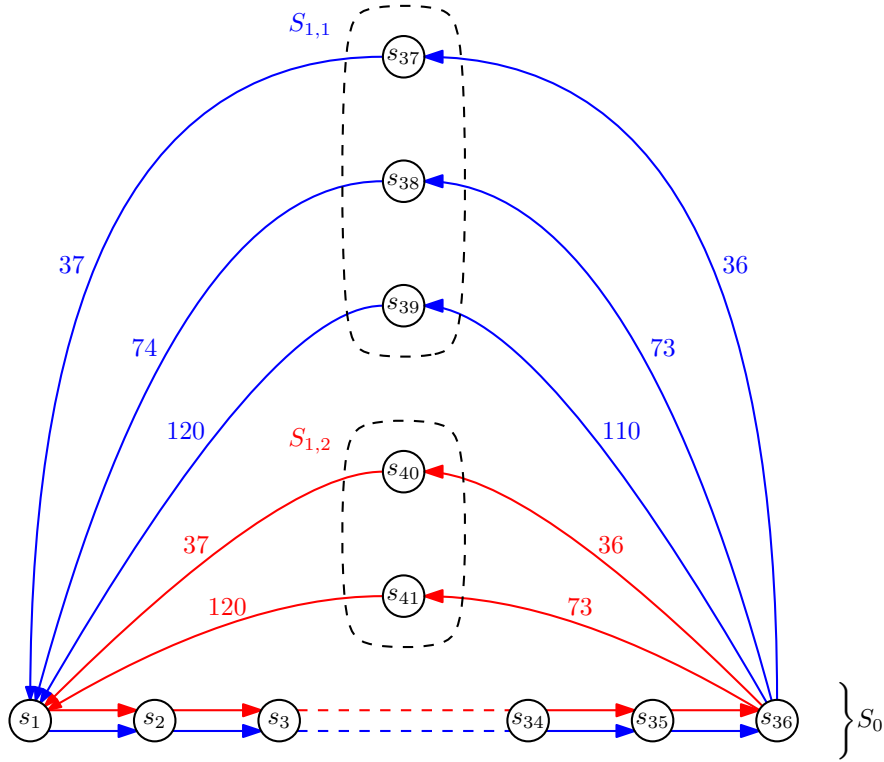


Figure 3: The C-graph $G_{n,k,p}$ used in Case 2 of the proof of Lemma 1, leading to the lower bound $\Omega(n^2)$. Here $n = 41$, $k = 2$, and $p = 120$.

263 moves to explore $G_{n,k,p}$.

264 Now assume that $k > n/16$. In this case, we simply use the above con-
 265 struction for $\lfloor n/16 \rfloor$ carriers. All carriers c_i , with $i > \lfloor n/16 \rfloor$ are given the
 266 same route as c_1 . This gives the same lower bound $7n/8(\lfloor n/8 \rfloor - 1)$. \square

267 Summarizing the previous lemma by considering the asymptotic behavior,
 268 we directly obtain the following theorem.

269 **Theorem 2.** *The worst-case move complexity of the C-graph exploration*
 270 *problem is in $\Omega(\min\{kp, np, n^2\})$, where n , k , and p denote respectively the*
 271 *number of sites, the number of carriers, and the maximal period. This result*
 272 *holds even if the agent knows completely the C-graph, has unlimited memory,*
 273 *and even in the labeled homogeneous highly-connected case.*

274 4.2. Lower bound on time

275 We prove a larger lower bound for the worst-case time complexity than
 276 for the worst-case move complexity in the general case. More precisely, we
 277 have the following lemma.

278 **Lemma 2.** Consider any $n \geq 2$, k , and p such that $n \leq p + (k - 1)(p - 1)$
 279 (necessary for connectedness). There exists a family $\mathcal{G}_{n,p,k}$ of labeled homoge-
 280 neous (connected) C-graphs with n sites, k carriers and period p such that, for
 281 any algorithm, there exists a C-graph in this family which cannot be explored
 282 by the algorithm using less than $p(n - 1 - \lfloor \frac{n-1}{\min(n-1,k)} \rfloor)$ time units.

283 *Sketch of proof.* The C-graphs used to prove this theorem are constructed as
 284 follows. Carriers are numbered from 1 to k . The carriers all have period p . A
 285 carrier i has only common sites with carriers $i - 1$ and $i + 1$. More precisely,
 286 carrier i shares exactly one site with carrier $i - 1$ and visits it exactly once
 287 per period. The proof is then based on the fact that the agent does not know
 288 precisely in which C-graph it is. In particular, the agent does not know when
 289 and on which site of carrier $i - 1$ the next carrier (number i) will pass. It is
 290 possible to prove, roughly, that the agent must wait at least p time units on
 291 each site to be sure to find the next carrier, leading to the claimed bound.

292 **PROOF.** Fix any $n \geq 2$, k , and p such that $n \leq p + (k - 1)(p - 1)$. We further
 293 assume that $k \leq n - 1$ (otherwise the exceeding carriers are fixed to have the
 294 same route as the first carrier).

295 Let $S = \{s_1, s_2, \dots, s_n\}$ be the set of sites and let $C = \{c_1, c_2, \dots, c_k\}$ be
 296 the set of carriers. Let us partition S into $k + 1$ subsets $S_0, S_1, \dots, S_{k-1}, S_k$
 297 such that $S_0 = \{s_1\}$, $|S_k| = \lfloor \frac{n-1}{k} \rfloor$, and for $1 \leq i \leq k - 1$, S_i has size $\lfloor \frac{n-1}{k} \rfloor$
 298 or $\lceil \frac{n-1}{k} \rceil$.

299 Fix any u_1, \dots, u_{k-1} and t_2, \dots, t_k such that, for every $1 \leq i \leq k - 1$, we
 300 have $u_i \in S_i$ and $1 \leq t_{i+1} \leq p$. The C-graph $G((u_1, t_2), (u_2, t_3), \dots, (u_{k-1}, t_k))$
 301 is defined as follows.

302 Let $u_0 = s_1$ and $t_1 = 0$. Consider any i such that $1 \leq i \leq k$. The route
 303 $R(c_i)$ is any route of period p going through (and only through) all the sites
 304 in $S_i \cup \{u_{i-1}\}$ satisfying the following two conditions. First, c_i visits u_{i-1} only
 305 once per period, at all times equal to t_i modulo p . Second, the route $R(c_i)$
 306 does not depend on the values u_l and t_{l+1} , for $l \neq i - 1$. Such a construction
 307 is possible thanks to the connectivity condition $n \leq p + (k - 1)(p - 1)$.

308 We denote $\mathcal{G}_{n,p,k}$ the family of all C-graphs $G((u_1, t_2), \dots, (u_{k-1}, t_k))$ with,
 309 for every $1 \leq i \leq k - 1$, $u_i \in S_i$ and $1 \leq t_{i+1} \leq p$. All these C-graphs

310 are labeled homogeneous connected C-graphs with n sites, k carriers and
 311 period p .

312 Let A be any exploring agent (i.e. executing any exploration algorithm).
 313 Given $1 \leq i \leq k$ and a C-graph G of $\mathcal{G}_{n,p,k}$, let $\mathcal{T}_i(G)$ be the first time at
 314 which the agent A , starting at s_1 at time 0 in G , sees the carrier c_i . Given q ,
 315 $1 \leq q \leq k$, and u_1, u_2, \dots, u_{q-1} and t_2, t_3, \dots, t_q in the usual ranges, we define
 316 $\mathcal{G}_{n,p,k}((u_1, t_2), (u_2, t_3), \dots, (u_{q-1}, t_q))$ as the set of all the C-graphs $G((u_1, t_2),$
 317 $(u_2, t_3), \dots, (u_{q-1}, t_q))$ with, for every $q \leq i \leq k-1$, $u_i \in S_i$ and $1 \leq t_{i+1} \leq p$.

318 **Claim 1.** *For every q , $1 \leq q \leq k$, there exist u_i and t_{i+1} satisfying $u_i \in S_i$
 319 and $1 \leq t_{i+1} \leq p$ for every i , $1 \leq i \leq q-1$, such that for every graph
 320 $G \in \mathcal{G}_{n,p,k}((u_1, t_2), (u_2, t_3), \dots, (u_{q-1}, t_q))$ we have $\mathcal{T}_q(G) \geq p \sum_{i=1}^{q-1} |S_i|$.*

321 *Proof of the Claim:* We prove the claim by induction on q . The base case
 322 $q = 1$ is trivially true. Fix any q such that $1 \leq q \leq k-1$, and assume, by
 323 induction hypothesis, that the claim holds for the value q .

324 Let \mathcal{G}_q be the family $\mathcal{G}_{n,p,k}((u_1, t_2), (u_2, t_3), \dots, (u_{q-1}, t_q))$ whose existence
 325 is guaranteed by the induction hypothesis. Note that all C-graphs in \mathcal{G}_q have
 326 exactly the same routes $R(c_i)$, for $1 \leq i \leq q$. We can thus define H_q to be
 327 the C-graph consisting only of the carriers c_1 to c_q of any C-graph in \mathcal{G}_q . Let
 328 us consider now the agent A starting at s_1 at time 0 in H_q . By induction
 329 hypothesis and by construction of H_q , the agent A sees c_q for the first time
 330 at time t with $t \geq p \sum_{i=1}^{q-1} |S_i|$ time units. Thus there exists u_q and t_{q+1}
 331 satisfying $u_q \in S_q$ and $1 \leq t_{q+1} \leq p$ such that A is never at u_q at a time
 332 equal to t_{q+1} modulo p before time $t + p|S_q|$, and thus before time $p \sum_{i=1}^q |S_i|$.

333 Consider now the agent A starting at s_1 at time 0 in any C-graph G in
 334 $\mathcal{G}_{n,p,k}((u_1, t_2), (u_2, t_3), \dots, (u_{q-1}, t_q), (u_q, t_{q+1}))$. Before time $p \sum_{i=1}^q |S_i|$, the
 335 agent will behave exactly the same as in H_q and will not see the carrier c_{q+1} .
 336 This concludes the proof of the claim. \diamond

337 The lemma follows by considering the claim for the last value $q = k$, and
 338 removing the assumption $k \leq n-1$. \square

339 Again, summarizing the previous lemma by considering the asymptotic
 340 behavior, we directly obtain the following theorem.

341 **Theorem 3.** *The worst-case time complexity of the C-graph exploration prob-
 342 lem is in $\Omega(np)$ in the general case (for $k \geq 2$). This result holds even if
 343 the agent knows n , k , and p , has unlimited memory, and even in the labeled
 344 homogeneous (connected) case.*

345 *4.3. Our algorithm*

346 In the above part of the paper, we exhibited some necessary conditions on
347 the existence of a solution. We then provided lower bounds on the worst-case
348 move and time complexities. We now essentially prove that all these results
349 are optimal by describing and proving a C-graph exploration algorithm with
350 matching upper bounds on the move and time complexities, provided that
351 the agent knows a linear upper bound B on the maximum period p . As a
352 consequence, we show that the ability to wait allows to decrease both the
353 worst-case move and time complexities, the former by a multiplicative factor
354 at least $\Theta(p)$.

355 *4.3.1. Principle*

356
357 As previously specified, our algorithm uses an upper bound B on the
358 largest period p of the C-graph (cf. Theorem 3). The main idea of the
359 algorithm consists of getting off on each site and, during $\Theta(B)$ time units, to
360 note each visit of the carriers at the site. Properly managed, this information
361 allows to map the C-graph (i.e., to list the routes and times of passage of all
362 carriers).

363 Several precautions must be taken into account in order not to miss any
364 site and to optimize the number of moves. For example, after each study of a
365 site (during $\Theta(B)$ time units), the algorithm computes the smallest possible
366 period of each seen carrier by using the already collected information. This
367 allows to know all the future passing times of the carriers on the studied
368 sites. In order to avoid unnecessary moves, the algorithm uses the concept
369 of current carrier. The agent studies all the sites of the current carrier before
370 moving on to the next. The algorithm also maintains a tree of carriers,
371 where a carrier c is a child of a carrier c' if c was discovered for the first
372 time while visiting c' . Carriers are treated in a depth-first-search manner for
373 performance reasons.

374 *4.3.2. Description*

375
376 In addition to the upper bound B on the largest period p of the C-graph,
377 our algorithm uses the variables described below.

378 The algorithm uses its own numbering to identify the sites. This way, it
379 will work even if the C-graph is anonymous.

380 • `currentNumber` : number of the currently studied site.

381 If the C-graph is labeled, the agent maintains a correspondence table
382 between the numbers given by the algorithm and the real identifiers of the
383 sites.

384 • `numberToID` : `numberToID[j]` is the identifier of the site number j .

385 The agent maintains an ordered rooted tree whose different vertices cor-
386 respond to the different encountered carriers.

387 • `tree` : the carrier tree.

388 • `currentCarrier` : identifier of the currently studied carrier.

389 For each carrier i present in `tree`, we have the following variables:

390 • `route[i]` is an array of length $3B$ (indexed from 0 to $3B - 1$); it is used
391 to memorize the sequence of sites visited by the carrier i .

392 • `position[i]` is an integer between 0 and $3B - 1$ (included); it indicates
393 the current position of the carrier i regarding to `route[i]`.

394 • `period[i]` is an integer between 1 and B (included); it indicates the
395 minimum period of the carrier i given the current knowledge.

396 4.3.3. Correctness

397 **Theorem 4.** *Algorithm EXPLORE-WITH-WAIT correctly explores and maps*
398 *in finite time any C-graph, even anonymous, but provided that an upper*
399 *bound B on the maximum period is known.*

400 **PROOF.** First observe that when an agent stays at a site for $2B$ time units,
401 where B is the known upper bound on the maximum period, it sees all the
402 carriers visiting that site. Moreover, after filling in the matrix with that
403 information, it is able to predict at any point in the future which carrier
404 will be at that site. Since the C-graph is connected, the agent will miss no
405 carriers and thus no sites either. At the end of the algorithm, the matrix will
406 be completely filled in and it will be equivalent to a map of the C-graph. \square

Algorithm EXPLORE-WITH-WAIT – Our C-graph exploration algorithm

```
1: bFinite ← false
2: currentNumber ← 1
3: currentCarrier ← 1
4: tree ← tree reduced to a single vertex (the root) corresponding to the
   carrier 1
5: while (bFinite = false) do
6:   studyCurrentSite()
7:   cleaning()
8:   if no value 0 in any array route[.] then
9:     bFinite ← true
10:  else
11:    findAndReachNextSite()
12:  end if
13: end while
14: Finish by providing the C-graph map (variables route[.], position[.] and
   eventually numberToID if the C-graph is labeled)
```

Procedure studyCurrentSite() – collect all possible information on the current site

```
1: Stay on the current site for  $2B$  time units
2: for each time unit do
3:   for each carrier  $i$  present on the current site at the current time do
4:     if  $i$  absent in tree then
5:       Add  $i$  as the last child of currentCarrier in tree
6:       route[ $i$ ] ← array of length  $3B$  filled with 0
7:       position[ $i$ ] ← 0
8:     end if
9:     route[ $i$ ][position[ $i$ ]] ← currentNumber
10:  end for
11: end for
```

After each time unit:

```
1: for each carrier  $i$  present in tree do
2:   position[ $i$ ] ← position[ $i$ ] + 1
3: end for
```

Procedure `cleaning()` – uses the acquired knowledge to update the variables

- 1: **for** each carrier i present in `tree` **do**
 - 2: `period[i]` \leftarrow minimum period of `route[i]` between `position[i] - 2B + 1` and `position[i]`
 - 3: Make the whole array `route[i]` periodic of period `period[i]`, using the values of `route[i]` between `position[i] - 2B + 1` and `position[i]`
 - 4: `position[i]` \leftarrow `position[i]` mod `period[i]`
 - 5: **end for**
-

Procedure `findAndReachNextSite()` – find the next site to study and go there

- 1: **if** `route[currentCarrier]` still contains the value 0 **then**
- 2: Compute from the array `route[currentCarrier]` the foremost journey using only `currentCarrier` that goes to the next site marked 0 on `currentCarrier`'s route
- 3: **else**
- 4: `oldCarrier` \leftarrow `currentCarrier`
- 5: `currentCarrier` \leftarrow identifier i of the first carrier following the DFS order in `tree` such that the value 0 appears in `route[i]`
- 6: Let $(c_{i_1}, c_{i_2}, \dots, c_{i_l})$ be the path of carriers in `tree` from `oldCarrier = c_{i_1}` to `currentCarrier = c_{i_l}`.
- 7: Compute from the arrays `route[.]` a foremost journey using only those carriers and such that, if the journey uses the carrier c_{i_j} and later the carrier $c_{i_{j'}}$, then $j \leq j'$.
- 8: **end if**
- 9: Transform the journey so that it leaves each site at most once (by waiting on the site)
- 10: Follow this journey
- 11: `currentNumber` \leftarrow `currentNumber` + 1
- 12: `numberToID[currentNumber]` \leftarrow identifier of the current site (if applicable)

After each time step along the journey:

- 1: **for** each carrier i present in `tree` **do**
 - 2: `position[i]` \leftarrow `position[i]` + 1 mod `period[i]`
 - 3: **end for**
-

407 *4.3.4. Move and time complexities*

408 **Lemma 3.** *When executing algorithm EXPLORE-WITH-WAIT, the agent*
409 *makes at most $O(\min\{kp, np, n^2\})$ moves and uses at most $O(\min\{kp, np\})$*
410 *time units in total in the different calls to procedure `findAndReachNextSite()`.*

411 **PROOF.** The procedure `findAndReachNextSite()` is executed between two
412 executions of Procedure `studyCurrentSite()`. This is done at most n times
413 because a site is studied only once.

414 Let us first prove that the agent makes at most $O(n^2)$ moves in total
415 in the different calls to the procedure `findAndReachNextSite()`. From the
416 previous remark, it is sufficient to prove that the agent performs at most
417 n moves for each execution of Procedure `findAndReachNextSite()`. This is
418 obviously the case as the path from the current site to the next unvisited site
419 is such that it leaves each site at most once.

420 We now prove that the number of moves and time units is in $O(kp)$.
421 Note that an agent finishes studying the sites of the current carrier's route
422 before going on another carrier's route. Therefore, given a carrier c , the calls
423 to procedure `findAndReachNextSite()` that stay on carrier c (line 2) are
424 consecutive. Since the agent follows the foremost journey computed from the
425 array `route[currentCarrier]` that only uses `currentCarrier` to go from the
426 current site to the next unvisited site, the different journeys corresponding to
427 these calls are consecutive and thus disjoint portions of c 's route. Therefore
428 the total number of moves and time units performed by the agent during
429 these calls concerning carrier c are bounded by p .

430 Let us now focus on the calls to procedure `findAndReachNextSite()` that
431 do not stay on the same carrier (lines 4–7). Note that the concatenation of the
432 paths computed line 6 in these calls consists of at most a DFS traversal of the
433 tree of carriers. Since a given carrier is at most once the end of such a path,
434 a carrier is at most its degree plus one times in a path. Each time, at most
435 p moves and time units are used. Hence these calls use at most $3kp$ moves
436 and time units. In total, all the calls to procedure `findAndReachNextSite()`
437 (whether using line 2 or lines 4–7) use at most $4kp$ moves and time units.

438 We finally prove that the number of moves and time units is in $O(np)$.
439 This is done by refining the previous argument. A carrier is always added
440 as a leaf to the tree of carriers. Moreover, a carrier is used only if the agent
441 goes to visit an unvisited site of the carrier. Since the agent has to visit at
442 most n sites, it means that at most n carriers of the tree are used. Hence
443 the number of moves and time units is bounded by $4np$. \square

444 With the algorithm EXPLORE-WITH-WAIT, the agent actually moves
445 only when executing Procedure `findAndReachNextSite()`. This gives the
446 following corollary.

447 **Corollary 1.** *With the algorithm EXPLORE-WITH-WAIT, the agent makes*
448 *at most $O(\min\{kp, np, n^2\})$ moves to explore any n -site k -carrier C -graph of*
449 *maximum period p .*

450 On the other hand, time is also spent when studying a site, in the calls
451 to procedure `studyCurrentSite()`.

452 **Lemma 4.** *When executing algorithm EXPLORE-WITH-WAIT, the agent*
453 *uses at most $O(nB)$ time units in total in the different calls to the proce-*
454 *cedure `studyCurrentSite()`, where B is a known upper bound on p .*

455 **PROOF.** The procedure `studyCurrentSite()` is executed when the agent studies
456 a site. This is done at most n times because a site is studied only once. During
457 the study of a site, the agent stays $O(B)$ time units on the site to note all
458 passing carriers. This gives the bound claimed in the lemma. \square

459 The obtained results from Lemma 3 and Lemma 4 give the following corol-
460 lary, noticing that time is only spent in the two procedures `studyCurrentSite()`
461 and `findAndReachNextSite()`.

462 **Corollary 2.** *The algorithm EXPLORE-WITH-WAIT allows to explore any*
463 *n -node C -graph in $O(nB)$ time units, where B is a known upper bound on p .*

464 Combining the previous results, we obtain the following corollary.

465 **Corollary 3.** *Given the a priori knowledge of an upper bound $B = O(p)$ on*
466 *the maximum period p , Algorithm EXPLORE-WITH-WAIT is asymptotically*
467 *optimal in the general case with respect to both the move and the time com-*
468 *plexities. The optimal worst-case move complexity is in $\Theta(\min\{kp, np, n^2\})$*
469 *while the optimal worst-case time complexity is in $\Theta(np)$.*

470 **5. Specific cases**

471 We showed in the previous section the optimal worst-case move and time
 472 complexities for the C-graph exploration problem in the general case. This
 473 section is devoted to the specific cases of homogeneous or highly-connected C-
 474 graphs. In both cases, we prove that the worst-case move and time complex-
 475 ities remain the same as in the general case. Note, however, that when con-
 476 sidering C-graphs being both homogeneous and highly-connected, we know
 477 from [9] that the optimal worst-case time complexity is at most $O(kp)$, even
 478 when n is large.

479 *5.1. The homogeneous case*

480 If we consider the homogeneous C-graphs (but not necessarily highly-
 481 connected), the worst-case time and move complexities remain the same as
 482 in the general case.

483 **Theorem 5.** *Given the a priori knowledge of an upper bound $B = O(p)$ on
 484 the maximum period p , Algorithm EXPLORE-WITH-WAIT is asymptotically
 485 optimal in the homogeneous case with respect to both the move and the time
 486 complexities. The optimal worst-case move complexity is in $\Theta(\min\{kp, np, n^2\})$
 487 while the optimal worst-case time complexity is in $\Theta(np)$.*

488 **PROOF.** The result directly follows from Theorem 2, Theorem 3 and Corol-
 489 lary 3. □

490 *5.2. The highly-connected case*

491 If we consider the highly-connected C-graphs (but possibly not homoge-
 492 neous), the worst-case time and move complexities remain the same as in the
 493 general case.

494 **Lemma 5.** *Consider any $n \geq 2$, k , and p such that $n \leq p + (k - 1)(p - 1)$
 495 (necessary for connectedness). There exists a family $\mathcal{G}'_{n,p,k}$ of labeled highly-
 496 connected C-graphs with n sites, k carriers and maximum period p such that,
 497 for any algorithm, there exists a C-graph in this family which cannot be ex-
 498 plored by the algorithm using less than $(p - 1)(n - 1 - \lfloor \frac{n-1}{\min(n-1,k)} \rfloor)$ time
 499 units.*

500 *Sketch of proof.* The C-graphs used to prove this theorem are constructed as
501 follows. Carriers are numbered from 1 to k . The carriers of odd identifier,
502 respectively even, are of period p , respectively $p - 1$. A carrier i has only
503 common sites with carriers $i - 1$ and $i + 1$. (The alternation of periods thus
504 ensures high connectivity.) More precisely, carrier i shares exactly one site
505 with carrier $i - 1$ and visits it exactly once per period. The proof is then
506 based on the fact that the agent does not know precisely in which C-graph it
507 is. In particular, the agent does not know when and on which site of carrier
508 $i - 1$ the next carrier (number i) will pass. It is possible to prove, roughly,
509 that the agent must wait at least p time units on each site to be sure to find
510 the next carrier, leading to the claimed bound.

511 **PROOF.** Fix any $n \geq 2$, k , and p such that $n \leq p + (k - 1)(p - 1)$. We further
512 assume that $k \leq n - 1$ (otherwise the exceeding carriers are fixed to have the
513 same route as the first carrier).

514 Let $S = \{s_1, s_2, \dots, s_n\}$ be the set of sites and let $C = \{c_1, c_2, \dots, c_k\}$ be
515 the set of carriers. Let us partition S into $k + 1$ subsets $S_0, S_1, \dots, S_{k-1}, S_k$
516 such that $S_0 = \{s_1\}$, $|S_k| = \lfloor \frac{n-1}{k} \rfloor$, and for $1 \leq i \leq k - 1$, S_i has size $\lfloor \frac{n-1}{k} \rfloor$
517 or $\lceil \frac{n-1}{k} \rceil$.

518 Fix any u_1, u_2, \dots, u_{k-1} and t_2, t_3, \dots, t_k such that, for every $1 \leq i \leq k - 1$,
519 we have $u_i \in S_i$ and $1 \leq t_{i+1} \leq p$ if i is odd, $1 \leq t_{i+1} \leq p - 1$, if i is even.
520 The C-graph $G((u_1, t_2), (u_2, t_3), \dots, (u_{k-1}, t_k))$ is defined as follows.

521 Let $u_0 = s_1$ and $t_1 = 0$. Consider any i such that $1 \leq i \leq k$. The route
522 $R(c_i)$ is any route going through (and only through) all the sites in $S_i \cup \{u_{i-1}\}$
523 satisfying the following three conditions. First, c_i is of period p if i is odd,
524 and of period $p - 1$ if i is even. Second, c_i visits u_{i-1} only once per period,
525 at all times equal to t_i modulo its period. Third, the route $R(c_i)$ does not
526 depend on the values u_l and t_{l+1} , for $l \neq i - 1$.

527 The family $\mathcal{G}'_{n,p,k}$ is defined as the set of all C-graphs $G((u_1, t_2), (u_2, t_3), \dots,$
528 $(u_{k-1}, t_k))$ with, for every $1 \leq i \leq k - 1$, $u_i \in S_i$ and $1 \leq t_{i+1} \leq p$, if i is odd,
529 $1 \leq t_{i+1} \leq p - 1$, if i is even. All these C-graphs are labeled highly-connected
530 C-graphs with n sites, k carriers and maximum period p . (Indeed, note that,
531 for every $1 \leq i \leq k - 1$, c_i and c_{i+1} meet at u_i at least every $p(p - 1)$ time
532 units.)

533 Let A be any exploring agent (i.e. executing any exploration algorithm).
534 Given $1 \leq i \leq k$ and G a C-graph of $\mathcal{G}'_{n,p,k}$, let $\mathcal{T}_i(G)$ be the first time at
535 which the agent A , starting at s_1 at time 0 in G , sees the carrier c_i . Given q ,
536 $1 \leq q \leq k$, and u_1, u_2, \dots, u_{q-1} and t_2, t_3, \dots, t_q in the usual ranges, we define

537 $\mathcal{G}'_{n,p,k}((u_1, t_2), (u_2, t_3), \dots, (u_{q-1}, t_q))$ as the set of all the C-graphs $G((u_1, t_2),$
538 $(u_2, t_3), \dots, (u_{k-1}, t_k))$ with, for every $q \leq i \leq k-1$, $u_i \in S_i$ and $1 \leq t_{i+1} \leq p$,
539 if i is odd, $1 \leq t_{i+1} \leq p-1$, if i is even.

540 **Claim 2.** *For every q , $1 \leq q \leq k$, there exist u_i and t_{i+1} satisfying $u_i \in S_i$
541 and $1 \leq t_{i+1} \leq p$ ($t_{i+1} \leq p-1$ when i is even) for every i , $1 \leq i \leq q-1$,
542 such that for every graph $G \in \mathcal{G}'_{n,p,k}((u_1, t_2), (u_2, t_3), \dots, (u_{q-1}, t_q))$ we have
543 $\mathcal{T}_q(G) \geq (p-1) \sum_{i=1}^{\lfloor \frac{q-1}{2} \rfloor} |S_{2i}| + p \sum_{i=1}^{\lceil \frac{q-1}{2} \rceil} |S_{2i-1}|$.*

544 *Proof of the Claim:* We prove the claim by induction on q . The base case
545 $q = 1$ is trivially true. Fix any q such that $1 \leq q \leq k-1$, and assume, by
546 induction hypothesis, that the claim holds for the value q .

547 Let \mathcal{G}'_q be the family $\mathcal{G}'_{n,p,k}((u_1, t_2), (u_2, t_3), \dots, (u_{q-1}, t_q))$ whose existence
548 is guaranteed by the induction hypothesis. Note that all C-graphs in \mathcal{G}'_q have
549 exactly the same routes $R(c_i)$, for $1 \leq i \leq q$. We can thus define H'_q to be
550 the C-graph consisting only of the carriers c_1 to c_q of any C-graph in \mathcal{G}'_q . Let
551 us consider now the agent A starting at s_1 at time 0 in H'_q . By induction
552 hypothesis and by construction of H'_q , the agent A sees c_q for the first time
553 at time t with $t \geq (p-1) \sum_{i=1}^{\lfloor \frac{q-1}{2} \rfloor} |S_{2i}| + p \sum_{i=1}^{\lceil \frac{q-1}{2} \rceil} |S_{2i-1}|$ time units. Thus
554 there exists u_q and t_{q+1} satisfying $u_q \in S_q$ and $1 \leq t_{q+1} \leq p$, if q is even,
555 $1 \leq t_{q+1} \leq p-1$, if q is odd, such that A is never at u_q at a time equal to
556 t_{q+1} modulo the period p' of c_{q+1} before time $t + p'|S_q|$, and thus before time
557 $(p-1) \sum_{i=1}^{\lfloor \frac{q}{2} \rfloor} |S_{2i}| + p \sum_{i=1}^{\lceil \frac{q}{2} \rceil} |S_{2i-1}|$.

558 Consider now the agent A starting at s_1 at time 0 in any C-graph G in
559 $\mathcal{G}'_{n,p,k}((u_1, t_2), (u_2, t_3), \dots, (u_{q-1}, t_q), (u_q, t_{q+1}))$. Before time $(p-1) \sum_{i=1}^{\lfloor \frac{q}{2} \rfloor} |S_{2i}| +$
560 $p \sum_{i=1}^{\lceil \frac{q}{2} \rceil} |S_{2i-1}|$, the agent will behave exactly the same as in H'_q and will not
561 see the carrier c_{q+1} . This concludes the proof of the claim. \diamond

562 The lemma follows by considering the claim for the last value $q = k$, and
563 removing the assumption $k \leq n-1$. \square

564 Again, summarizing the previous lemma, using Corollary 3, and consid-
565 ering the asymptotic behavior, we obtain the following theorem.

566 **Theorem 6.** *Given the a priori knowledge of an upper bound $B = O(p)$
567 on the maximum period p , Algorithm EXPLORE-WITH-WAIT is asymptot-
568 ically optimal in the highly-connected case with respect to both the move
569 and the time complexities. The optimal worst-case move complexity is in
570 $\Theta(\min\{kp, np, n^2\})$ while the optimal worst-case time complexity is in $\Theta(np)$.*

- 571 [1] C. Avin, M. Koucky, and Z. Lotker. How to explore a fast-changing
572 world (cover time of a simple random walk on evolving graphs). In *35th*
573 *International Colloquium on Automata, Languages and Programming*
574 *(ICALP)*, LNCS 5125, pages 121–132, 2008.
- 575 [2] B. Brejová, S. Dobrev, R. Královič, and T. Vinař. Efficient routing in
576 Carrier-Based Mobile Networks. In *Theoretical Computer Science*, 509,
577 pages 113–121, 2013.
- 578 [3] A. Casteigts, P. Flocchini, W. Quattrociocchi, and N. Santoro. Time-
579 varying graphs and dynamic networks. In *International Journal of Par-*
580 *allel, Emergent and Distributed Systems*, 27(5), pages 387–408, 2012.
- 581 [4] C. Cooper and A. M. Frieze. Crawling on simple models of web graphs.
582 In *Internet Mathematics*, 1(1), pages 57–90, 2003.
- 583 [5] A. Dessmark and A. Pelc. Optimal graph exploration without good
584 maps. In *Theoretical Computer Science*, 326(1-3), pages 343–362, 2004.
- 585 [6] G. A. Di Luna, S. Dobrev, P. Flocchini, and N. Santoro. Dis-
586 tributed exploration of dynamic rings. In *Distributed Computing*
587 <https://doi.org/10.1007/s00446-018-0339-1>, 2018.
- 588 [7] P. Flocchini, M. Kellett, P. C. Mason, and N. Santoro. Searching for
589 black holes in subways. In *Theory of Computing Systems*, 50(1), pages
590 158–184, 2012.
- 591 [8] P. Flocchini, M. Kellett, P. C. Mason, and N. Santoro. Finding Good
592 Coffee in Paris. In *6th International Conference on Fun with Algorithms*
593 *(FUN)*, LNCS 7288, pages 154–165, 2012.
- 594 [9] P. Flocchini, B. Mans, and N. Santoro. On the exploration of time-
595 varying networks. In *Theoretical Computer Science*, 469, pages 53–68,
596 2013.
- 597 [10] D. Ilcinkas and A. M. Wade. On the Power of Waiting when Exploring
598 Public Transportation Systems. In *15th International Conference On*
599 *Principles Of Distributed Systems (OPODIS)*, LNCS 7109, pages 451–
600 464, 2011.

- 601 [11] D. Ilcinkas and A. M. Wade. Exploration of the T-Interval-Connected
602 Dynamic Graphs: the Case of the Ring. In *Theory of Computing Sys-*
603 *tems*, volume 62, number 5, pages 1144–1160, 2018.
- 604 [12] F. Kuhn, N. A. Lynch, and R. Oshman. Distributed computation in
605 dynamic networks. In *42nd ACM Symposium on Theory of Computing*
606 *(STOC)*, pages 513–522, 2010.
- 607 [13] P. Panaite and A. Pelc. Exploring Unknown Undirected Graphs. In
608 *Journal of Algorithms*, 33(2), pages 281–295, 1999.
- 609 [14] O. Reingold. Undirected connectivity in log-space In *Journal of the*
610 *ACM*, 55(4),pages 17:1–17:24, 2008.