Exploration of carrier-based time-varying networks: The power of waiting

David Ilcinkas, Ahmed Mouhamadou Wade

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].

[1] Partially 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).

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

Keywords: Exploration, Dynamic graphs, Mobile agent, C-graph

1. Introduction

1.1. The problem

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

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

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

1.2. Related work

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

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

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

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

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

1.3. Our results

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

Results from [9] Our results	Connected	Highly-connected
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.

2. Model and definitions

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

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

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

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

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

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

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

3. Solvability

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

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

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

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

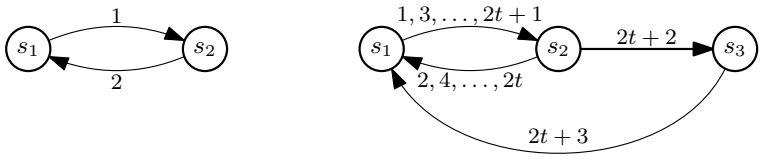

Figure 1: The C-graphs G_0 and G_t of the family $\mathcal{G}$.

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

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

4. General case

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

4.1. Lower bound on the number of moves

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

Lemma 1. *For any integers n, k, p such that $n \leq p + (k-1)(p-1)$ (necessary for connectedness), there exists a labeled homogeneous highly-connected C-graph $G_{n,k,p}$ with n sites, k carriers and period p such that any algorithm 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.*

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

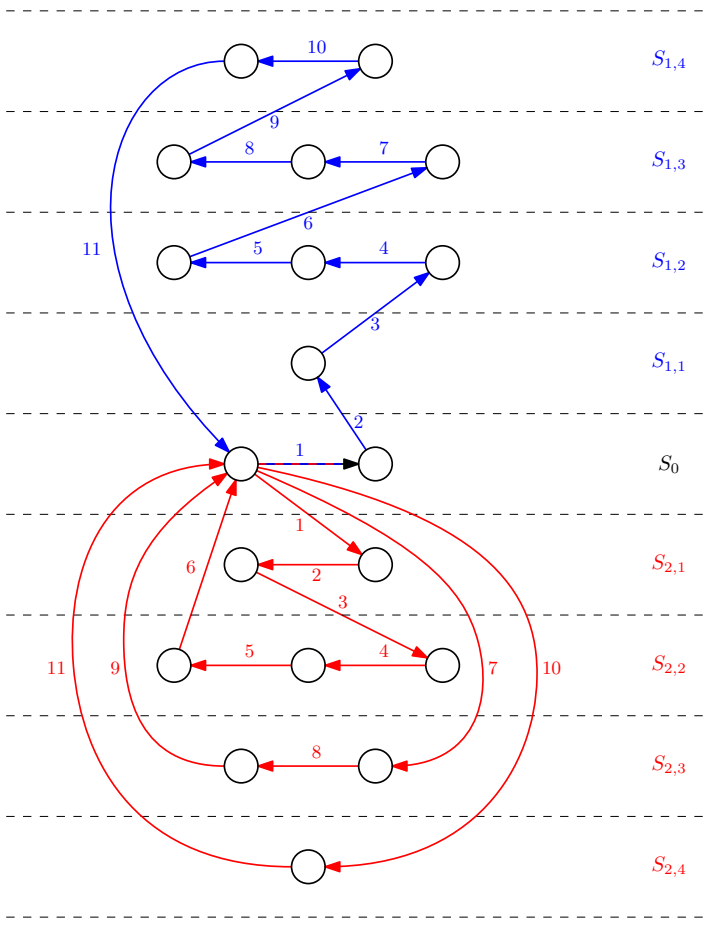

Figure 2: The C-graph $G_{n,k,p}$ used in Case 1 of the proof of Lemma 1, leading to the lower bound $\Omega(\min\{kp, np\})$. Here $n = 19$, $k = 2$, and $p = 11$.

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

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

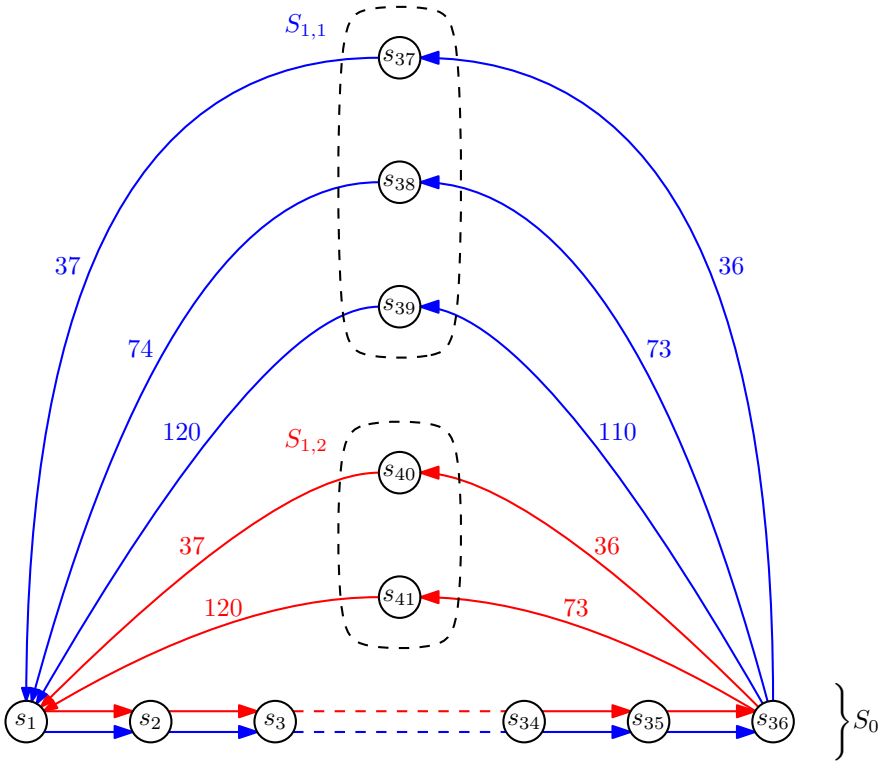

Figure 3: The C-graph $G_{n,k,p}$ used in Case 2 of the proof of Lemma 1, leading to the lower bound $\Omega(n^2)$. Here $n = 41$, $k = 2$, and $p = 120$.

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

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

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

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

4.2. Lower bound on time

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

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

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

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

Let $S = \{s_1, s_2, \ldots, s_n\}$ be the set of sites and let $C = \{c_1, c_2, \ldots, c_k\}$ be the set of carriers. Let us partition S into $k+1$ subsets $S_0, S_1, \ldots, S_{k-1}, S_k$ 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 $\lfloor \frac{n-1}{k} \rfloor$ or $\lceil \frac{n-1}{k} \rceil$.

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

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

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

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

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

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

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

Let $\mathcal{G}_q$ be the family $\mathcal{G}_{n,p,k}((u_1,t_2),(u_2,t_3),\ldots,(u_{q-1},t_q))$ whose existence is guaranteed by the induction hypothesis. Note that all C-graphs in $\mathcal{G}_q$ have exactly the same routes $R(c_i)$, for $1 \leq i \leq q$. We can thus define H_q to be the C-graph consisting only of the carriers c_1 to c_q of any C-graph in $\mathcal{G}_q$. Let us consider now the agent A starting at s_1 at time 0 in H_q. By induction hypothesis and by construction of H_q, the agent A sees c_q for the first time 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} 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 equal to t_{q+1} modulo p before time $t+p|S_q|$, and thus before time $p\sum_{i=1}^{q}|S_i|$.

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

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

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

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

4.3. Our algorithm

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

4.3.1. Principle

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

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

4.3.2. Description

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

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

- `currentNumber` : number of the currently studied site.

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

- `numberToID` : `numberToID`$[j]$ is the identifier of the site number j.

The agent maintains an ordered rooted tree whose different vertices correspond to the different encountered carriers.

- `tree` : the carrier tree.
- `currentCarrier` : identifier of the currently studied carrier.

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

- `route`$[i]$ is an array of length $3B$ (indexed from 0 to $3B-1$); it is used to memorize the sequence of sites visited by the carrier i.
- `position`$[i]$ is an integer between 0 and $3B-1$ (included); it indicates the current position of the carrier i regarding to `route`$[i]$.
- `period`$[i]$ is an integer between 1 and B (included); it indicates the minimum period of the carrier i given the current knowledge.

4.3.3. Correctness

Theorem 4. *Algorithm* Explore-With-Wait *correctly explores and maps in finite time any C-graph, even anonymous, but provided that an upper bound B on the maximum period is known.*

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

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 $(c_{i_1}, c_{i_2}, \ldots, c_{i_l})$ 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**

4.3.4. Move and time complexities

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

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

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

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

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

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

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

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

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

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

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

The obtained results from Lemma 3 and Lemma 4 give the following corollary, noticing that time is only spent in the two procedures studyCurrentSite() and `findAndReachNextSite`().

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

Combining the previous results, we obtain the following corollary.

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

5. Specific cases

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

5.1. The homogeneous case

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

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

Proof. The result directly follows from Theorem 2, Theorem 3 and Corollary 3. □

5.2. The highly-connected case

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

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

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

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

Let $S = \{s_1, s_2, \ldots, s_n\}$ be the set of sites and let $C = \{c_1, c_2, \ldots, c_k\}$ be the set of carriers. Let us partition S into $k+1$ subsets $S_0, S_1, \ldots, S_{k-1}, S_k$ 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 $\lfloor \frac{n-1}{k} \rfloor$ or $\lceil \frac{n-1}{k} \rceil$.

Fix any $u_1, u_2, \ldots, u_{k-1}$ and $t_2, t_3, \ldots, t_k$ such that, for every $1 \leq i \leq k-1$, 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. The C-graph $G((u_1, t_2), (u_2, t_3), \ldots, (u_{k-1}, t_k))$ is defined as follows.

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

The family $\mathcal{G}'_{n,p,k}$ is defined as the set of all C-graphs $G((u_1, t_2), (u_2, t_3), \ldots, (u_{k-1}, t_k))$ 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, $1 \leq t_{i+1} \leq p-1$, if i is even. All these C-graphs are labeled highly-connected C-graphs with n sites, k carriers and maximum period p. (Indeed, note that, 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 units.)

Let A be any exploring agent (i.e. executing any exploration algorithm). 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 which the agent A, starting at s_1 at time 0 in G, sees the carrier c_i. Given q, $1 \leq q \leq k$, and $u_1, u_2, \ldots, u_{q-1}$ and $t_2, t_3, \ldots, t_q$ in the usual ranges, we define

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

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

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

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

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

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

Again, summarizing the previous lemma, using Corollary 3, and considering the asymptotic behavior, we obtain the following theorem.

Theorem 6. *Given the a priori knowledge of an upper bound $B = O(p)$ on the maximum period p, Algorithm* Explore-With-Wait *is asymptotically optimal in the highly-connected case with respect to both the move and the time complexities. The optimal worst-case move complexity is in $\Theta(\min\{kp, np, n^2\})$ while the optimal worst-case time complexity is in $\Theta(np)$.*

[1] C. Avin, M. Koucky, and Z. Lotker. How to explore a fast-changing world (cover time of a simple random walk on evolving graphs). In *35th International Colloquium on Automata, Languages and Programming (ICALP)*, LNCS 5125, pages 121–132, 2008.

[2] B. Brejová, S. Dobrev, R. Královič, and T. Vinař. Efficient routing in Carrier-Based Mobile Networks. In *Theoretical Computer Science*, 509, pages 113–121, 2013.

[3] A. Casteigts, P. Flocchini, W. Quattrociocchi, and N. Santoro. Time-varying graphs and dynamic networks. In *International Journal of Parallel, Emergent and Distributed Systems*, 27(5), pages 387–408, 2012.

[4] C. Cooper and A. M. Frieze. Crawling on simple models of web graphs. In *Internet Mathematics*, 1(1), pages 57–90, 2003.

[5] A. Dessmark and A. Pelc. Optimal graph exploration without good maps. In *Theoretical Computer Science*, 326(1-3), pages 343–362, 2004.

[6] G. A. Di Luna, S. Dobrev, P. Flocchini, and N. Santoro. Distributed exploration of dynamic rings. In *Distributed Computing* https://doi.org/10.1007/s00446-018-0339-1, 2018.

[7] P. Flocchini, M. Kellett, P. C. Mason, and N. Santoro. Searching for black holes in subways. In *Theory of Computing Systems*, 50(1), pages 158–184, 2012.

[8] P. Flocchini, M. Kellett, P. C. Mason, and N. Santoro. Finding Good Coffee in Paris. In *6th International Conference on Fun with Algorithms (FUN)*, LNCS 7288, pages 154–165, 2012.

[9] P. Flocchini, B. Mans, and N. Santoro. On the exploration of time-varying networks. In *Theoretical Computer Science*, 469, pages 53–68, 2013.

[10] D. Ilcinkas and A. M. Wade. On the Power of Waiting when Exploring Public Transportation Systems. In *15th International Conference On Principles Of Distributed Systems (OPODIS)*, LNCS 7109, pages 451–464, 2011.

[11] D. Ilcinkas and A. M. Wade. Exploration of the T-Interval-Connected Dynamic Graphs: the Case of the Ring. In *Theory of Computing Systems*, volume 62, number 5, pages 1144–1160, 2018.

[12] F. Kuhn, N. A. Lynch, and R. Oshman. Distributed computation in dynamic networks. In *42nd ACM Symposium on Theory of Computing (STOC)*, pages 513–522, 2010.

[13] P. Panaite and A. Pelc. Exploring Unknown Undirected Graphs. In *Journal of Algorithms*, 33(2), pages 281–295, 1999.

[14] O. Reingold. Undirected connectivity in log-space In *Journal of the ACM*, 55(4),pages 17:1–17:24, 2008.