Collision-Free Network Exploration

Jurek Czyzowicz, Dariusz Dereniowski, Leszek Gasieniec, Ralf Klasing, Adrian Kosowski, Dominik Pajak

Collision-Free Network Exploration

Jurek Czyzowicz * Dariusz Dereniowski † Leszek Gąsieniec ‡ Ralf Klasing §
Adrian Kosowski ¶ Dominik Pająk §

September 27, 2012

Abstract

A set of mobile agents is placed at different nodes of a n-node network. The agents synchronously move along the network edges in a *collision-free* way, i.e., in no round may two agents occupy the same node. In each round, an agent may choose to stay at its currently occupied node or to move to one of its neighbors. An agent has no knowledge of the number and initial positions of other agents. We are looking for the shortest possible time required to complete the collision-free *network exploration*, i.e., to reach a configuration in which each agent is guaranteed to have visited all network nodes and has returned to its starting location.

We first consider the scenario when each mobile agent knows the map of the network, as well as its own initial position. We establish a connection between the number of rounds required for collision-free exploration and the degree of the minimum-degree spanning tree of the graph. We provide tight (up to a constant factor) lower and upper bounds on the collision-free exploration time in general graphs, and the exact value of this parameter for trees. For our second scenario, in which the network is unknown to the agents, we propose collision-free exploration strategies running in $O(n^2)$ rounds for tree networks and in $O(n^5 \log n)$ rounds for general networks.

1 Introduction

The graph searching problem is a task of central importance in many contexts, including network maintenance, terrain patrolling, and robotics. Its different aspects have been thoroughly investigated, cf. [7]. The *rendezvous search* problem has been often presented as a game with two mobile players walking within the *search space* and having the common goal of arriving at the same time at the same location (see [4]). On the other hand, the *exploration problem* consists in examining all elements of the search space by a mobile agent (e.g. visiting all graph nodes or traversing all its edges), e.g., in order to find a hidden target (see [1, 13]). The problem is often stated as the zero-sum game between the Searcher and the Hider. The Searcher attempts to minimize the time of its walk while the Hider does not want to be found (i.e. tries to maximize the time until its discovery). The vast literature on this problem covers diverse models, in particular the cases of a mobile Hider and a team of collaborating Searchers, often described as "cops and robbers" game (cf. [5, 7]).

*Université du Québec en Outaouais, Gatineau, Québec J8X 3X7, Canada
†Gdańsk University of Technology, 80-233 Gdańsk, Poland. E-mail: deren@eti.pg.gda.pl
‡University of Liverpool, Liverpool L69 3BX, UK. E-mail: l.a.gasieniec@liverpool.ac.uk
§LaBRI, CNRS — Université de Bordeaux — Inria, 33400 Talence, France. E-mail: ralf.klasing@labri.fr
¶Inria Bordeaux Sud-Ouest, 33400 Talence, France. E-mails: {adrian.kosowski,dominik.pajak}@inria.fr

In this paper we propose a new graph searching problem in which each of a set of mobile agents must explore a given undirected graph, in such a way that two agents may never visit the same node of the graph at the same time. This property of the model, which we call *collision avoidance*, is motivated by the fact that the processes executed by mobile agents (software agents or physical robots) sometimes require exclusive access to network resources. According to our knowledge, our problem has not been studied in the past, although a question related to its offline version has been given some attention in the context of routing (cf. [3]).

In our considerations, time is divided into synchronous rounds. Initially, each agent is placed at a different node and in each round it may choose to move to a neighboring node or to stay motionless. The agents are independent in the sense that they cannot communicate and none of them knows the number of other agents or their initial placement in the graph. The agents move independently, and each of them executes the same algorithm. The effectiveness of the algorithm is measured in terms of the *collision-free exploration time*, i.e., the number of rounds until all potentially existing agents are certain to have completed the exploration and returned to their initial location. Details of our model are discussed in Section 2.

1.1 Our results

We consider two scenarios, differing in the amount of global information about the network topology which is available to each agent. Our results are summarized in Table 1.

For the first scenario, considered in Section 3, we assume that a map of the network is *a priori* known to the agents. We show that a collision-free exploration strategy exists for any graph, and provide efficient solutions for trees and general graphs. For trees, we propose a strategy which involves the simultaneous activation of agents located at the endpoints forming a matching in some optimal edge-coloring of the tree. This strategy is shown to yield optimal exploration time. For general graphs, we show a surprisingly tight connection between our problem and the fractional relaxation of the LP formulation of the minimum-degree spanning tree problem. We make use of this to provide a lower bound on the collision-free exploration time in graphs, as well as a polynomially constructible exploration strategy which admits an exploration time within a constant factor of the optimum.

In the second scenario, discussed in Section 4, we deal with synchronous agents possessing only local knowledge about the graph to explore. In particular, no knowledge of the size of the graph is assumed. We suppose that each agent executes a local, distributed algorithm, in every round making a decision based on the information concerning the currently occupied node and the identifiers of the neighboring nodes. For this scenario, we show that a collision-free exploration is always feasible in finite time and we give algorithms for trees and general graphs. Our collision-free exploration strategies are of length $O(n^2)$ for trees and $O(n^5 \log n)$ for arbitrary graphs, and make use of the application of universal exploration sequences.

Throughout the paper, we assume that the strategies for collision-free exploration are required to return the agent to their initial location. This assumption allows us to see our strategies as an analogue of the classical Traveling Salesman Problem with mutually-exclusive salesmen on an unweighted graph, and also allows the agents to engage in perpetual (periodic) exploration of the graph. After minor modification of the proofs, all the results presented in Table 1 also hold up to constant factors for the variant of the problem in which agents may end exploration at an arbitrary node of the graph.

Table 1: The time of optimal collision-free graph exploration. $\Delta(G)$ denotes the maximum degree of a node in graph G, and $\Delta^*(G) = \Delta(T)$, where T is a minimum-degree spanning tree of G.

Scenario	*Tree*	*General graph*
With complete map:	$n\Delta(G)$ Thm. 3.2	$\Theta(n\Delta^*(G))$ Thm. 3.4
With local knowledge:	$O(n^2)$ Thm. 4.1	$O(n^5 \log n)$ Thm. 4.2

1.2 Related work

The offline setting of our question is related to the following problem (cf. [3]), which was studied in the context of routing. Each vertex of a given graph is initially occupied by a "pebble", which has to be moved to a destination, so that the destinations of different pebbles are different. In every synchronous round a set of edges is selected and the pebbles at each edge endpoints are interchanged. [3] attempts to minimize the number of rounds so that all pebbles reach their destination, giving lower and upper bounds for different classes of graphs. The routing model of [3] inherently implies the usage of matchings - the technique that we choose to apply in some results of our paper. The $3n$ upper bound for trees given in [3] was improved to $\frac{1}{2}n + O(\log n)$ in [16]. [12] and [14] independently extended this model to allow more than one pebble per origin and destination node. Although the matching model of routing was also considered in the online setting (e.g. [14]), this is unrelated to our paper. Indeed, in online routing the distributed decisions are made by the network nodes on the basis of local information concerning incoming packets, while in exploration, the mobile agents determine their subsequent moves while learning a piece of information about the network structure.

For the (classical) graph exploration problem with local knowledge, a lot of attention has been given to exploration in *anonymous networks*, in which the agent, when located at a node, has to decide on its next move based only on its own local memory state, the local port ordering at the node, and the port by which it entered the current node. It has been shown in [8] that an agent must be equipped with at least n states (i.e., $\Omega(\log n)$ bits of memory) to be able to explore all anonymous graphs with n nodes. On the positive side, unknown anonymous graphs can be deterministically explored by following so called universal traversal/exploration sequences. These exist for any number of nodes, and have polynomial length [2]. The exploration time obtained using such an approach is $O(n^5 \log n)$, i.e., a factor of about n^2 greater than the (expected) cover time of a corresponding random walk. It has only been shown recently [15] that such universal exploration sequences can also be constructed and followed using very small memory, and consequently, deterministic graph exploration can be performed by an agent with only $O(\log n)$ bits of memory. However, the exploration time is then expressed by a polynomial with a high exponent.

2 Model and definitions

We assume that the nodes of each n-node network have unique identifiers in $\{1, \ldots, n\}$. The identifier of a node v is denoted by $\sigma(v)$. Several agents are initially located at pairwise different nodes of the network. The initial position of each agent λ is denoted by $h(\lambda)$. Each agent is unaware of the

number and initial positions of the other agents, and all agents are given the same algorithm that determines their behavior in the subsequent rounds.

Each agent can perceive the identifier $\sigma(v)$ of the currently occupied node v and can perceive the identifiers of all neighbors of v. Moreover, the agent can distinguish the edges incident to v according to the identifiers of the nodes located at the endpoints of the edges. The latter assumption is necessary to properly perform the navigation in a node labeled network.

The agents are synchronous and hence the time is divided into rounds of equal duration. Each round is divided into two stages. In the first stage each agent λ makes a decision (by executing its algorithm) that determines its behavior in the second stage of the round. The decision can be three-fold:

(i) λ decides to stay, in this particular round, at the currently occupied node,

(ii) λ decides to move from the currently occupied node to one of its neighbors,

(iii) λ decides that the exploration is completed.

In the second stage of the round, all agents simultaneously perform one of the actions (i),(ii) or (iii). If, as a result, two agents located at some adjacent nodes u and v decide to move from u to v and from v to u, respectively, then they traverse the same edge in this round, but remain unaware of this event, i.e., the two agents do not communicate and do not perceive each other. We require that the algorithm given to the agents ensures the following:

- at the end of each round no two agents are present on the same node of the network,
- there exists an integer t such that by the end of round t all agents have decided that the exploration is completed,
- each agent visited each node of the network in one of the rounds $1, \ldots, t$,
- each agent λ is present at $h(\lambda)$ at the end of round t.

Note that, in this setting, the execution of the agent's algorithm (and thus the behavior of the agent) only depends on the input to the algorithm and on the identifiers of the nodes visited by the agent. Thus, in particular, an agent is unable to ever discover the initial or current position of any other agent, or the number of agents in the network, unless such an information is provided as an input.

With respect to the additional information available to the agents, we study two scenarios in this work: either the algorithm for designing the strategy has no input, or the complete map of the network is given. In the latter case the map consists of node identifiers, but provides no information on the locations of other agents. Note that if, together with a complete map of the network, all agents receive as an input information on the initial positions of all agents, then our exploration problem becomes similar to the off-line routing problems considered e.g. in [3, 14, 16].

Let $G = (V(G), E(G))$ be any network. For any node v of G let $N_G(v)$ be the set of neighbors of v in G. We use the symbol $\Delta(G)$ to denote the *degree* of G, defined as $\Delta(G) = \max\{|N_G(v)|\colon v \in V(G)\}$. ($|N_G(v)|$ is called the *degree* of v.) Given a set of edges $X \subseteq E(G)$, define $G[X]$ to be the network with nodes in $V(G)$ and edges in X, $G[X] = (V(G), X)$. Note that $G[X]$ is not necessarily connected. A connected network H such that $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$ is called a *connected component* of G if there exists no connected network H' such that $V(H) \subseteq V(H') \subseteq V(G)$ and $E(H) \subseteq E(H') \subseteq E(G)$ and $H \neq H'$.

Any sequence $\mathcal{R} = (v_0, v_1, \ldots, v_l)$ of nodes of a network G is called a *route in* G if $v_i = v_{i-1}$ or $\{v_i, v_{i-1}\}$ is an edge of G for each $i = 1, \ldots, l$. We say that l is the *length* of $\mathcal{R}$ and we write $\mathcal{R}_i = v_i$ for each $i = 0, \ldots, l$. The route $\mathcal{R}$ *covers* G if for each node v of G there exists $i \in \{0, \ldots, l\}$ such that $v = \mathcal{R}_i$. The route $\mathcal{R}$ is *closed* if $\mathcal{R}_0 = \mathcal{R}_l$, where l is the length of $\mathcal{R}$. Let λ be an agent. We say that the route $\mathcal{R}$ of length l is a *route of* λ if: (i) $\mathcal{R}_0 = h(\lambda)$ and λ is present at $\mathcal{R}_i$ at the end

of round i, $i = 1, \ldots, l$, and (ii) λ does not move in any round $r > l$.

We say that a route $\mathcal{R}$ of length l is an *exploration strategy for* λ if (i) $\mathcal{R}$ is a route of λ, (ii) $\mathcal{R}$ is closed, (iii) $\mathcal{R}$ covers G. Two routes $\mathcal{R}$ and $\mathcal{R}'$ of length l are *collision-free* if $\mathcal{R}_i \neq \mathcal{R}'_i$ for each $i = 0, \ldots, l$. Let $\mathcal{A} = \{\lambda_1, \ldots, \lambda_k\}$, $1 \leq k \leq n$, be the set of agents that are initially located at the nodes of G. Let $\mathcal{R}(\lambda)$ be the exploration strategy for each agent $\lambda \in \mathcal{A}$. We say that $\mathcal{R}(\lambda_1), \ldots, \mathcal{R}(\lambda_k)$ are *collision-free* if $\mathcal{R}(\lambda_i)$ and $\mathcal{R}(\lambda_j)$ are collision-free for each $i, j \in \{1, \ldots, k\}$, $i \neq j$. Let t be the minimum integer such that for each set of agents placed arbitrarily on the nodes of G there exist collision-free exploration strategies, each of length at most t, for the agents. Then, t is called the *collision-free exploration time of* G.

3 Network exploration with a map

In this section we consider the problem of collision-free exploration in the case when each agent is given a complete map of the network to be explored. In Subsection 3.1 we focus on tree networks, and in Subsection 3.2 we work with arbitrary networks.

3.1 Tree exploration with a map

We start with some notation and two preliminary lemmas that are the main tool in the analysis of an algorithm given in this section.

Given a tree network T, we say that a function $c\colon E(T) \to \{1, \ldots, d\}$ is a *d-edge-coloring* of T if $c(e) \neq c(e')$ for any two adjacent edges in T.

Let d be an integer, let c be a d-edge-coloring of G, and let v be any node of G. Define $\mathcal{T}(v, d, c) = (v_0, v_1, v_2, \ldots)$ to be an infinite route in G starting at v such that:
(i) if $c(\{v_{i-1}, u\}) \neq 1 + (i-1) \bmod d$ for each neighbor u of v_{i-1} in G, then $v_i = v_{i-1}$,
(ii) if $c(\{v_{i-1}, u\}) = 1 + (i-1) \bmod d$ for some neighbor u of v_{i-1}, then $v_i = u$.
Then, define $\mathcal{T}^l(v, d, c)$, $l \geq 0$, to be the prefix of $\mathcal{T}(v, d, c)$ of length l, and $\mathcal{T}^l_i(v, d, c)$ to be v_i for each $i = 0, \ldots, l$.

See Figure 1 for an example of $\mathcal{T}^{dn}(v, d, c)$ computed for a tree network T on $n = 15$ nodes. Figure 1(a) gives T and a 6-edge-coloring c of T (thus, $d = 6$ in this example).

We now give two preliminary lemmas in which we prove that if u and v are two distinct nodes of T, then the routes $\mathcal{T}^{dn}(u, d, c)$ and $\mathcal{T}^{dn}(v, d, c)$ are collision-free, and each of them is closed and covers the tree network.

Lemma 3.1. *Let T be a tree network. If c is a d-edge-coloring of T, then for any two distinct nodes u and v of T the routes $\mathcal{T}^l(u, d, c)$ and $\mathcal{T}^l(v, d, c)$ are collision-free for each $l \geq 0$.*

Proof. Let $l \geq 0$ be fixed arbitrarily. Denote $\mathcal{T}^l(u, d, c) = (u_0, u_1, \ldots, u_l)$ and $\mathcal{T}^l(v, d, c) = (v_0, v_1, \ldots, v_l)$.

We prove by induction on $i = 0, \ldots, l$ that $\mathcal{T}^i(u, d, c)$ and $\mathcal{T}^i(v, d, c)$ are collision-free. By assumption, $u \neq v$ and by definition, $\mathcal{T}^0(u, d, c) = (u)$ and $\mathcal{T}^0(v, d, c) = (v)$. Hence, the claim follows for $i = 0$.

Assume that the claim holds for some $i \in \{0, \ldots, l-1\}$ and we prove it for $i+1$. If $u_i = u_{i+1}$ and $v_i = v_{i+1}$, then the proof is completed. Thus, $u_i \neq u_{i+1}$ or $v_i \neq v_{i+1}$ and assume without loss of generality that the former occurs. By construction of $\mathcal{T}(u, d, c)$, $c(\{u_i, u_{i+1}\}) = 1 + (i \bmod d)$. If $u_{i+1} = v_i$, then from the construction of $\mathcal{T}(v, d, c)$ we obtain that $v_{i+1} = u_i$ and consequently

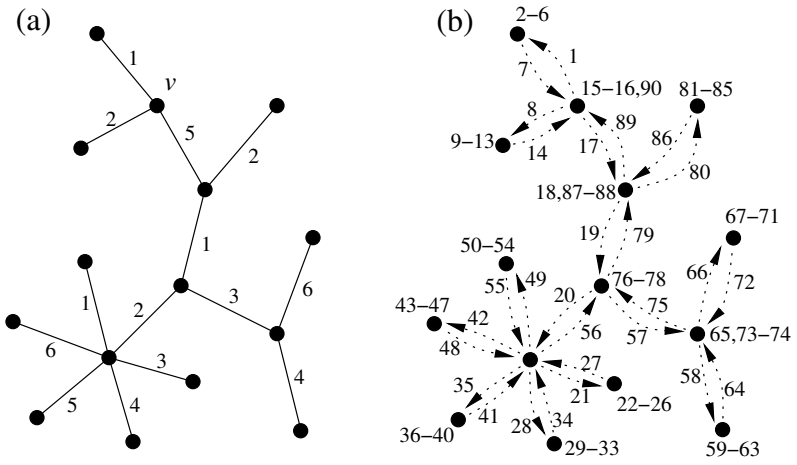

Figure 1: (a) a 15-node tree network T with a 6-edge-coloring c; (b) the route $\mathcal{T}^{dn}(v,d,c) = \mathcal{T}^{90}(v,6,c) = (v_0,\ldots,v_{90})$ encoded as follows: an integer i that is a label of an arc going from u to v indicates that $v_i = v$ and $v_{i-1} = u$; an integer label i of a node u of T means that $v_i = v_{i-1} = u$.

$u_{i+1} \neq v_{i+1}$ as required. If $u_{i+1} \neq v_i$, then $v_{i+1} \neq u_{i+1}$, because $u_i \neq v_i$ and c is an edge-coloring of T. □

Lemma 3.2. *Let T be a tree network and let d be an integer. If c is a d-edge-coloring of T and v is a node of T, then the route $\mathcal{T}^{dn}(v,d,c)$ is closed and covers T.*

Proof. Consider T to be rooted at v and let T_u to be a subtree of T induced by u and all its descendants for each $u \in V(T)$. Assume without loss of generality that T consists of at least two nodes. Denote $\mathcal{T}^{dn}(v,d,c) = (v_0, v_1, \ldots, v_{dn})$.

We prove by induction on the subtree size that for each $u \in V(T)$, if $i \in \{0,\ldots,dn\}$ is the minimum index such that $v_i = u$, then $(v_i,\ldots,v_{i+s})$ is closed and covers T_u, where $s = |V(T_u)|d - 1$.

We first consider the case of a single node tree T_u. Since $|V(T)| > 1$, u has a parent u'. By the definition of $\mathcal{T}(v,d,c)$, $c(\{u,u'\}) = 1 + (i-1) \bmod d$. Let $s \geq 0$ be the maximum index such that $v_i = v_{i+1} = \cdots = v_{i+s}$. Clearly, $v_{i+s+1} = u'$. We obtain that $s \geq d-1$, because otherwise $1 + (i+s) \bmod d = c(\{v_{i+s}, v_{i+s+1}\}) = c(\{u,u'\}) = 1 + (i-1) \bmod d$, and hence $(s+1) \bmod d = 0$, which is a contradiction. Moreover, $v_{i+d} = u'$, because $1 + (i+d-1) \bmod d = 1 + (i-1) \bmod d$. Hence, $s = d-1$ as required.

Consider any rooted subtree T_u, $|V(T_u)| > 1$, and suppose that the claim holds for any subtree with less than $|V(T_u)|$ nodes. We first consider the case when $u \neq v$.

Let $u_1,\ldots,u_p$, $p \geq 1$, be the children of u in T_u. By the definition of edge-coloring, $d \geq \Delta(T)$, and hence $p \leq d$. Let u' be the parent of u in T, and assume without loss of generality that for some $1 \leq q \leq p$

$$c(\{u,u'\}) < c(\{u,u_1\}) < \cdots < c(\{u,u_q\}) \text{ and } c(\{u,u_{q+1}\}) < \cdots < c(\{u,u_p\}) < c(\{u,u'\}). \quad (1)$$

Let i_j be the minimum index such that $v_{i_j} = u_j$, $j = 1,\ldots,p$. By the induction hypothesis, $\mathcal{R}(u_j) = (v_{i_j},\ldots,v_{i_j+s_j})$ is closed and covers T_{u_j}, where $s_j = |V(T_{u_j})|d - 1$, for each $j = 1,\ldots,p$.

First we prove that for each $j = 1, \ldots, p$ it holds $v_{i_j+s_j+1} = u$. Let $j \in \{1, \ldots, p\}$ be selected arbitrarily. By the choice of i_j, $v_{i_j} \neq v_{i_j-1}$. Since v_0 is the root of T, $u = v_{i_j-1}$. This implies that

$$c(\{u, u_j\}) = c(\{v_{i_j-1}, v_{i_j}\}) = 1 + (i_j - 1) \bmod d. \tag{2}$$

Note that

$$1 + (i_j + s_j) \bmod d = 1 + (i_j + |V(T_{u_j})|d - 1) \bmod d = 1 + (i_j - 1) \bmod d. \tag{3}$$

By the fact that $\mathcal{R}(u_j)$ is closed, $u_j = v_{i_j+s_j}$. Hence, by (2) and (3),

$$c(\{u_j, u\}) = c(\{v_{i_j+s_j}, v_{i_j+s_j+1}\}) = c(\{u_j, v_{i_j+s_j+1}\}).$$

Since, c is an edge-coloring of T, $v_{i_j+s_j+1} = u$ as required.

Let $\mathcal{C}(u_j)$ be the maximal subsequence of $\mathcal{T}^{dn}(v, d, c)$ starting with $v_{i_j+s_j+1}$ and with all elements equal u, $j = 1, \ldots, p$. By (1) and by construction of $\mathcal{T}(v, d, c)$,

$$\mathcal{C}(u_j) = (v_{i_j+s_j+1}, \ldots, v_{i_{j+1}-1}) \text{ for each } j = 1, \ldots, p-1. \tag{4}$$

Define $\mathcal{C}(u_0)$ and $\mathcal{C}(u_p)$ to be the maximal subsequences of $\mathcal{T}^{dn}(v, d, c)$ starting with v_i and v_{i_p} respectively, with all elements equal u. Note that the definition of $\mathcal{C}(u_0)$ is correct, because $v_i = u$. By (4),

$$(v_i, \ldots, v_{i+s}) = \mathcal{C}(u_0), \mathcal{R}(u_1), \mathcal{C}(u_1), \mathcal{R}(u_2), \mathcal{C}(u_2), \ldots, \mathcal{R}(u_p), \mathcal{C}(u_p),$$

where

$$s = \sum_{j=0}^{p} |\mathcal{C}(u_j)| + \sum_{j=1}^{p} |\mathcal{R}(u_j)| - 1 = 2p + \sum_{j=0}^{p} (|\mathcal{C}(u_j)| - 1) + \sum_{j=1}^{p} (|V(T_{u_j})|d - 1). \tag{5}$$

The sum $\sum_{j=0}^{p} (|\mathcal{C}(u_j)| - 1)$ equals, informally speaking, the number of two consecutive appearances of u in $(v_{i+1}, \ldots, v_{i+s})$. By (1), this sum equals $d - p - 1$, because c is a d-edge-coloring of T and u is not the root of T. Hence, by (5),

$$s = d - 1 + \sum_{j=1}^{p} |V(T_{u_j})|d = d(1 + \sum_{j=1}^{p} |V(T_{u_j})|) - 1 = |V(T_u)|d - 1.$$

Finally, if $u = v$, then the proof is analogous, and the fact that u has no parent implies that $\sum_{j=0}^{p} (|\mathcal{C}(u_j)| - 1) = d - p$. Hence, we obtain that $s = dn$ when u is the root, which completes the proof. □

It remains to observe that the considered routes can be implemented as exploration strategies. Indeed, each agent λ is able to construct some d-edge-coloring c of T (the same for all agents, e.g., lexicographically first with respect to some chosen ordering of all colorings) with $d = \Delta(T)$, and hence it is able to 'follow' $\mathcal{T}^{n\Delta(T)}(h(\lambda), \Delta(T), c)$. We formulate this strategy in the form of the algorithm below.

Algorithm `Tree-Exploration`(T)
Input: A node-labeled tree network T.
begin
Let v be the initial position of the executing agent.
Compute the lexicographically first $\Delta(T)$-edge-coloring c of T
for each round $r \leftarrow 1$ **to** $n\Delta(T)$ **do**
if there exists an edge $\{v, u\}$ such that $c(\{v, u\}) = 1 + (r-1) \bmod \Delta(T)$ **then**
move from v to u in round r, set $v \leftarrow u$.
else stay at v in round r.
end `Tree-Exploration`

For an agent λ following Algorithm `Tree-Exploration`, its route is of length $n\Delta(T)$, and given as $\mathcal{R}^{n\Delta(T)}(\lambda) = \mathcal{T}^{n\Delta(T)}(h(\lambda), \Delta(T), c)$, where c is the $\Delta(T)$-edge-coloring computed in the Algorithm. Consequently, taking into account Lemmas 3.1 and 3.2, we have the following.

Proposition 3.1. *Let T be a tree network and let $\lambda_1, \ldots, \lambda_k$, $1 \leq k \leq n$, be the agents initially located at pairwise different nodes of T. Suppose that the agent λ_i uses Algorithm* `Tree-Exploration` *to compute its route $\mathcal{R}^{n\Delta(T)}(\lambda_i)$, for each $i = 1, \ldots, k$. Then, $\mathcal{R}^{n\Delta(T)}(\lambda_1), \ldots, \mathcal{R}^{n\Delta(T)}(\lambda_k)$ are exploration strategies, and are collision-free.* □

It turns out that there exist no shorter collision-free exploration strategies than those constructed with Algorithm `Tree-Exploration`.

Theorem 3.2. *The collision-free exploration time of any n-node tree network T is precisely equal to $n\Delta(T)$.*

Proof. The upper bound follows from Proposition 3.1. Now, we prove the lower bound, i.e., that the collision-free exploration time of T is at least $n\Delta(T)$. Let u be a node of degree $\Delta(T)$ in T. First assume that there are n agents in T. We say that an agent λ is *active* in round r if λ goes from v to u in round r for some $v \in N_T(u)$. In each round at most one agent is active. For each agent λ there exist at least $\Delta(T)$ rounds in which λ is active, because the route of λ needs to be closed and T is a tree. Since there are n agents in total, we obtain that there are at least $n\Delta(T)$ rounds in which an agent is active. This proves that there exists an agent λ that is active in round $n\Delta(T)$, and hence its exploration strategy is of length at least $n\Delta(T)$. Finally, observe that λ constructs the same route regardless of the number of agents present in the network. This is due to the fact that T and $\sigma(h(\lambda))$ is the entire input to the algorithm that λ executes. □

We finish this section by remarking on the complexity of Algorithm `Tree-Exploration`. For any tree network T on n nodes, there exists a $\Delta(T)$-edge-coloring of T and it can be computed in $O(n)$-time. Consequently, the total time of an agent's local computations when running Algorithm `Tree-Exploration` is $O(n\Delta(T))$.

3.2 General network exploration with a map

We say that T is a *spanning tree* of G if T is a tree such that $V(T) = V(G)$ and $E(T) \subseteq E(G)$. Then, T is a *minimum degree spanning tree* of G if T is a spanning tree of G and the degree of T is minimum over the degrees of all spanning trees of G. Define $\Delta^*(G) = \Delta(T)$, where T is a minimum degree spanning tree of G. We propose the following solution to the collision-free exploration problem.

Algorithm `Network-Exploration`(G)
Input: A node-labeled network G.
begin
Compute the lexicographically first minimum-degree spanning tree T^* of G.
Call Algorithm `Tree-Exploration`(T^*).
end `Network-Exploration`

The next proposition follows from the formulation of Algorithm `Network-Exploration` and from Proposition 3.1.

Proposition 3.3. *Let G be a network and let $\lambda_1, \ldots, \lambda_k$, $1 \leq k \leq n$, be the agents initially located at pairwise different nodes of G. Suppose that the agent λ_i uses Algorithm* `Network-Exploration` *to compute its route $\mathcal{R}(\lambda)$, $i = 1, \ldots, k$. Then, $\mathcal{R}(\lambda_1), \ldots, \mathcal{R}(\lambda_k)$ are collision-free exploration strategies of length $n\Delta^*(G)$.* □

The following theorem implies that our result is asymptotically tight, i.e., it implies that Algorithm `Network-Exploration` constructs exploration strategies whose length is within a constant factor from the optimum.

Theorem 3.4. *The collision-free exploration time of any network G is $\Theta(n\Delta^*(G))$.*

Proof. The fact that the collision-free exploration time of G is $O(n\Delta^*(G))$ follows from Proposition 3.3.

Now, we prove the lower bound of $\Omega(n\Delta^*(G))$. Observe that if $\Delta^*(G) \leq 3$, then the theorem follows, because each exploration strategy must be of length $\Omega(n)$. To finish the proof, suppose that there exist exploration strategies for the agents, such that the length of each exploration strategy is at most $n(\Delta^*(G) - 3)/2$.

For each node v of G let $E_v = \{\{v, u\} \colon u \in N_G(v)\}$. Consider the following linear program (LP) with variables $(x_e \colon e \in E(G))$, which satisfies the following set of constraints [6, 11]:

$$\sum_{e \in E(G)} x_e = n - 1 \tag{6}$$

$$\sum_{e \in E(G[S])} x_e \leq |S| - 1, \quad \text{for each } S \subseteq V(G) \tag{7}$$

$$\sum_{e \in E_v} x_e \leq t, \quad \text{for each } v \in V(G) \tag{8}$$

$$0 \leq x_e \leq 1, \tag{9}$$

where t is an integer and n is the number of nodes of G.

Any solution to the above problem is called a *fractional spanning tree* of *degree* t of G. Informally speaking, if $(x_e \colon e \in E(G))$, is a solution to (6)-(9), then x_e is the 'fraction' of the edge e that is included in the resulting fractional spanning tree. Note that any integer solution, i.e. the one in which $x_e \in \{0, 1\}$ for each $e \in E(G)$, is a spanning tree of degree at most t of G.

Suppose that n agents $\lambda_1, \ldots, \lambda_n$ are present in the network G. Let $\mathcal{R}(\lambda_1), \ldots, \mathcal{R}(\lambda_n)$ be some collision-free exploration strategies for the agents. Suppose that the length of each exploration strategy is at most $nt/2$. Based on these exploration strategies, we now construct a solution to the LP in (6)-(9). For each $i = 1, \ldots, n$, let T_i be any spanning tree of G such that if $e \in E(T_i)$,

then there exists a round r such that $e = \{\mathcal{R}_{r-1}(\lambda_i), \mathcal{R}_r(\lambda_i)\}$ (in other words, λ_i traverses e in some round). Such a T_i exists, because $\mathcal{R}(\lambda_i)$ covers G, $i = 1, \ldots, n$. Define:

$$f_i(e) = \begin{cases} 1/n, & \text{if } e \in E(T_i) \\ 0, & \text{if } e \notin E(T_i) \end{cases} \qquad \text{and} \qquad x_e = \sum_{i=1}^{n} f_i(e) \text{ for each } e \in E(G). \tag{10}$$

Now, we prove that x_e's defined in (10) form a solution to the LP in (6)-(9).

First note that $\sum_{e \in E(G)} f_i(e) = (n-1)/n$ for each $i = 1, \ldots, n$, because f_i assigns $1/n$ to exactly $n-1$ edges of G, which follows from the fact that T_i is a spanning tree of G, $i = 1, \ldots, n$. Thus, (6) holds.

Now, let $S \subseteq V(G)$ be selected arbitrarily. For each $i = 1, \ldots, n$, $|E(T_i) \cap E(G[S])| = |E(T_i[S])| \leq |S| - 1$, because $T_i[S]$ is, by definition, a collection of node-disjoint trees on set S. Hence, (7) follows.

Let v be any node of G and let $X = E_v \cap (E(T_1) \cup \cdots \cup E(T_n))$. For each r there exist at most two edges in X traversed by an agent in round r. Hence,

$$\sum_{e \in X} \sum_{i=1}^{n} f_i(e) \leq \frac{nt}{2} \cdot \frac{2}{n} = t.$$

Note that if $e \in E_v \setminus X$, then $\sum_{i=1}^{n} x_e = 0$. This proves that (8) holds.

Finally, (9) follows directly from (10).

We have proved that the existence of exploration strategies of length $nt/2$ implies the existence of a solution to (6)-(9). Moreover, we have the following.

Claim ([11]). *If there exists a solution to* (6)-(9)*, then there exists an integer solution to (6),(7),(9) with additional constraint*

$$\sum_{e \in E_v} x_e \leq t + 2 \text{ for each } v \in V(G)$$

that replaces (8).

We remark that such an integer solution defines a spanning tree of G, given by the set of edges $\{e \in E(G) \colon x_e = 1\}$.

In view of the definition of x_e's in (10), it follows that if there exist exploration strategies of length at most $nt/2$ for the n agents, then there exists a spanning tree T^* of G, and the degree of T^* is at most $t + 2$. By assumption, there exist in G exploration strategies of length at most $n(\Delta^*(G) - 3)/2$, hence, putting $t = \Delta^*(G) - 3$, it follows that G has a spanning tree of degree at most $\Delta^*(G) - 1$, a contradiction with the definition of $\Delta^*(G)$. This completes the proof. □

We finish this section with a complexity remark. Finding a minimum-degree spanning tree is in general an NP-hard problem. We can, however, modify the approach to obtain an exploration strategy of length $n(\Delta^*(G)+1)$ that can be computed efficiently. We make use of a $O(mn\alpha(m,n)\log n)$-time algorithm that for a given G finds its spanning tree T of degree $\Delta(T) \leq \Delta^*(G) + 1$, where m and n are, respectively, the number of edges and nodes of G, and α is the inverse Ackermann function [9]. By using the tree T in Algorithm `Network-Exploration` instead of T^* we obtain an exploration strategy of length $n(\Delta^*(G) + 1)$ for agent λ, and this strategy is computed in time $O(mn\alpha(m,n)\log n)$.

Proposition 3.5. *The problem of deciding, for a given network G and integer l, whether the collision-free exploration time of G is at most l, is* NP*-complete.*

Proof. We prove that the problem is NP-complete already for the special case of $l = n$, where n is the number of nodes of G. The proof is by reduction from the Hamiltonian cycle problem [10]. We argue that there exists a Hamiltonian cycle of G if and only if the collision free exploration time of G is n.

If such a Hamiltonian cycle $C = v_1\text{-}v_2\text{-}\cdots\text{-}v_n$ exists, then one constructs an exploration strategy $\mathcal{R}(\lambda)$ for an agent λ with $h(\lambda) = v_i$ by taking $\mathcal{R}(\lambda) = (v_i, v_{i+1}, \ldots, v_n, v_1, \ldots, v_{i-1}, v_i)$. Clearly, $\mathcal{R}(\lambda)$ is a route of length n in G, because C is a cycle. Also, $\mathcal{R}(\lambda)$ is closed and covers G. Moreover, if for another agent λ' we have $h(\lambda') \neq v_i$, then $\mathcal{R}(\lambda)$ and $\mathcal{R}(\lambda')$ are collision-free.

Now suppose that the collision-free exploration time of G equals n. Let λ be any agent initially occupying any node of G and take an exploration strategy $\mathcal{R}(\lambda)$ of length n for λ. Since $\mathcal{R}(\lambda)$ covers G, $\mathcal{R}_i(\lambda) \neq \mathcal{R}_{i-1}(\lambda)$ for each $i = 1, \ldots, n$. By the fact that $\mathcal{R}(\lambda)$ is closed, $\mathcal{R}_0(\lambda) = \mathcal{R}_n(\lambda)$. Hence, $\mathcal{R}(\lambda)$ is a cycle of length n in G as required. □

4 Local network exploration

In this section we consider the problem of collision-free exploration in the setting when the agents do not receive any information about the network in which they operate. Recall that we assume, that each node $v \in V$ is equipped with a unique identifier $\sigma(v) \in \{1, 2, \ldots, n\}$, and each agent located at v is only aware of the identifier $\sigma(v)$ and the identifiers of the neighbors of v at the endpoints of respective edges incident to v. In Section 4.1 we consider tree networks, and in Section 4.2 we show how any network can be explored.

Let G be any network. For the purposes of this section we introduce an edge-labeling function σ' defined as

$$\sigma'(\{u, v\}) = \sigma(u) + \sigma(v) \text{ for each } \{u, v\} \in E(G). \tag{11}$$

We recall without proof the following essential property of function σ'.

Lemma 4.1. *Let G be any n-node network. Then, σ' is a $2n$-edge-coloring of G.* □

4.1 Local exploration of tree networks

In this section we provide an algorithm which defines collision-free routes of agents, and is guaranteed to perform exploration if the explored network is a tree. For any integer $b \geq 2$ define the following sequence of integers $U(b) = (1, \ldots, 2b, \quad \ldots, \quad 1, \ldots, 2b)$, where $1, \ldots, 2b$ is repeated b times, and let $U_i(b)$, $i \in \{1, \ldots, 2b^2\}$, be its i-th element.

Algorithm `Local-Tree-Exploration`
begin
Let v be the initial position of the executing agent.
$b \leftarrow 2$
$r \leftarrow 0$
while not all nodes have been visited so far **do** {start a new phase}
for $s \leftarrow 1$ **to** $|U(b)|$ in round $r+s$ **do**
if there exists an edge $\{v,u\}$ such that $\sigma(u) \leq b$
and $\sigma(v) \leq b$ and $\sigma'(\{v,u\}) = U_s(b)$
then move from v to u {in round $r+s$}; set $v \leftarrow u$.
else stay at v. {in round $r+s$}
end for
$r \leftarrow r + |U(b)|$
$b \leftarrow 2b$
end while
Backtrack all previous moves, i.e., λ moves from v to u in round $r+i$ if and only if λ moved from u to v in round $r-i+1$ for each $i = 1, \ldots, r$.
end `Local-Tree-Exploration`

Define *phase* p, $p \geq 1$, as the sequence of rounds $(1+\sum_{j=1}^{p-1} |U(2^j)|, \ldots, \sum_{j=1}^{p} |U(2^j)|)$ and denote by $\ell(p) = |U(2^p)|$ the number of rounds of phase p. Note that

$$\ell(p) = 2^{2p+1} \text{ for each } p \geq 1, \tag{12}$$

and that phase p consists of the rounds in which the behavior of any agent λ is determined in the p-th iteration of the 'while' loop of its execution of Algorithm `Local-Tree-Exploration`, whenever p does not exceed the total number of iterations executed. Denote by $\mathcal{R}(\lambda, p)$ the route of an agent λ restricted to its moves in phase p, $p \geq 1$.

We denote by T_p the subgraph of T induced by all edges e whose endpoints have identifiers at most 2^p, $T_p = T[\{\{u,v\} \in E(T) \colon \sigma(u) \leq 2^p \wedge \sigma(v) \leq 2^p\}]$.

Finally, define $\ell = 2\sum_{p=1}^{\lceil \log_2 n \rceil} \ell(p)$.

We now prove that each agent λ moves in phase p 'inside' the connected component T' of T_p that contains the vertex occupied by λ at the beginning of phase p.

Lemma 4.2. *Let $p \geq 1$ be an integer, let T be a tree network and let λ be an agent. Let v be the vertex occupied by λ at the beginning of phase p. Then, $\mathcal{R}(\lambda, p)$ is a route in the connected component of T_p that contains v, and $\mathcal{R}(\lambda, p) = \mathcal{T}^{\ell(p)}(v, 2^{p+1}, \sigma')$.*

Proof. First we argue that $\mathcal{R}(\lambda, p)$ is a route in the connected component T' of T_p that contains the node v. The agent λ performs its moves in phase p as a result of the execution of the p-th iteration of the 'while' loop of Algorithm `Local-Tree-Exploration`. The value of the variable b in this p-th iteration equals 2^p. Hence, if λ decides to move from a node v to a node u in some round of phase p, then $\sigma(u) \leq 2^p$ and $\sigma(v) \leq 2^p$. Thus, $\{u,v\}$ is an edge of T_p, and therefore $\{u,v\} \in E(T')$.

To conclude that $\mathcal{R}(\lambda, p) = \mathcal{T}^{\ell(p)}(v, 2^{p+1}, \sigma')$, note that, by Lemma 4.1, σ' restricted to T' is a 2^{p+1}-edge-coloring of T'. Thus, the lemma follows from the definition of $\mathcal{T}$ and from the formulation of Algorithm `Local-Tree-Exploration`. □

Note that the length of the route $\mathcal{R}(\lambda, p)$ of λ in phase p is bounded by $\ell(p)$, hence is, in general, 'unrelated' to the number of nodes of T'. For this reason, T' need not be completely explored. However, by the definition of T_p, we have that $T_p = T$ (and $T' = T$) if and only if $p \geq \lceil \log_2 n \rceil$. We use this observation to show that all agents perform backtracking and stop after exactly the same phase $p = \lceil \log_2 n \rceil$, and that in this phase each of them visits all nodes of T.

Lemma 4.3. *Let T be a n-node tree network. For each agent λ the number of iterations of the 'while' loop of Algorithm* `Local-Tree-Exploration` *executed by λ equals $\lceil \log_2 n \rceil$. Moreover, $\mathcal{R}(\lambda, \lceil \log_2 n \rceil)$ covers T.*

Proof. If $p \in \{1, \ldots, \lceil \log_2 n \rceil - 1\}$, then $T_p \neq T$. Hence, T_p is not connected and, by Lemma 4.2, $\mathcal{R}(\lambda, p)$ does not cover T. The agent λ determines this fact, e.g., by recording, in a set X, the identifiers of all nodes adjacent to the nodes of its route in phase p, $\mathcal{R}(\lambda, p)$. Then, due to the connectedness of T, X contains an identifier such that the corresponding node is not in $\mathcal{R}(\lambda, p)$ and consequently λ starts executing the $(p+1)$-st iteration of the 'while' loop of Algorithm `Local-Tree-Exploration`.

Now, let $p = \lceil \log_2 n \rceil$, and so $T_p = T$. Due to Lemma 4.2, $\mathcal{R}(\lambda, p) = \mathcal{T}^{\ell(p)}(u, 2^{p+1}, \sigma')$, where u is the node occupied by λ at the beginning of phase p. By the formulation of Algorithm `Local-Tree-Exploration` and by (12), $\ell(p) = 2^{2p+1} \geq 2n^2 \geq 2n\Delta(T)$. By Lemma 3.2, $\mathcal{R}(\lambda, p)$ covers T, because, due to Lemma 4.1, σ' is a $2n$-edge-coloring of T. □

We now argue that the agents will never meet while moving during any given phase p.

Lemma 4.4. *Let λ and λ' be any two agents, let T be a tree network, and let $p \geq 1$ be an integer. The routes $\mathcal{R}(\lambda, p)$ and $\mathcal{R}(\lambda', p)$ are collision-free.*

Proof. Let u and u' be the nodes occupied by λ and λ', respectively, at the beginning of phase p. By the formulation of Algorithm `Local-Tree-Exploration`, the moves of both agents in phase p are determined in the p-th iteration of the 'while' loop of their executions of Algorithm `Local-Tree-Exploration`. If u and u' belong to different connected components of T_p, then due to Lemma 4.2 the routes $\mathcal{R}(\lambda, p)$ and $\mathcal{R}(\lambda', p)$ are collision-free. Hence, assume that u and v are in the same connected component T' of T_p. By Lemma 4.2, the routes in T' are given as $\mathcal{R}(\lambda, p) = \mathcal{T}^{\ell(p)}(u, 2^{p+1}, \sigma')$ and $\mathcal{R}(\lambda', p) = \mathcal{T}^{\ell(p)}(u', 2^{p+1}, \sigma')$. Thus, the proof is complete in view of Lemma 3.1. □

Theorem 4.1. *Let T be a tree network and let $\lambda_1, \ldots, \lambda_k$, $1 \leq k \leq n$, be the agents initially located at pairwise different nodes of T. Suppose that the agent λ_i uses Algorithm* `Local-Tree-Exploration` *to compute its route $\mathcal{R}^\ell(\lambda_i)$, for each $i = 1, \ldots, k$. Then, $\mathcal{R}^\ell(\lambda_1), \ldots, \mathcal{R}^\ell(\lambda_k)$ are collision-free exploration strategies of length $O(n^2)$.*

Proof. By Lemma 4.3, each route $\mathcal{R}^\ell(\lambda_i)$, $i = 1, \ldots, l$ covers T in at least one phase. Since the route performed by each agent is closed due to the backtracking steps included in Algorithm `Local-Tree-Exploration`, $\mathcal{R}^\ell(\lambda_i)$ is an exploration strategy for λ_i. Taking into account that the phases of all agents are perfectly synchronized in each phase, and that the agents perform backtracking and stop after exactly the same phase $\lceil \log_2 n \rceil$, it follows from Lemma 4.4 that their exploration strategies are collision-free. Finally, from the definition of ℓ we have $\ell = O(n^2)$. □

4.2 Local exploration of general networks

For the purposes of analysis, we introduce some auxiliary notation concerning the so-called *anonymous graph model*. In this model nodes are anonymous, and each edge has two port numbers assigned, each to one of its endpoints, in such a way that the ports at edges incident to any node form a set of consecutive integers, starting from 1. An agent located at a node v can only perform its next move based on the local port numbers.

Before continuing, we provide several comments and informal intuitions concerning this model. First note that a collision-free exploration is, in general, impossible in arbitrary anonymous port-labeled networks. (This is the case, for example, for two agents located initially in symmetric, and thus indistinguishable, positions at the endpoints of a 3-node path.) However, we will overcome this difficulty by designing an auxiliary port-labeled network $A(G)$ based on the node-labeled network G, that has the property that each edge has identical port numbers at both of its endpoints, and in such a case the collision-free exploration will be guaranteed to exist. The behavior of an agent can be seen as navigating in our node-labeled network G by navigating in the underlying 'virtual' port-labeled network $A(G)$. In particular, the function σ' defined in (11) provides both port numbers for each edge. Hence, each agent, while present at any node v can compute the port number of the edges incident to v. Then, the agent 'simulates' its next move in the port-labeled network and, based on that, performs the move in the node-labeled network.

As a tool for our analysis we use the theory of *universal sequences* (formal definitions are provided below) that has been developed for regular port-labeled networks. Such a universal sequence, once computed by all agents, is then used to find a collision-free exploration strategy in the port-labeled network. In view of our earlier comment, the latter results in the collision-free exploration strategy in the node-labeled network.

We say that a network is *d-regular* if all nodes of the network have degrees equal to d. Given a port-labeled network A and a node v of A, we say that an agent λ initially located at v *follows* a sequence of integers $U = (x_1, \ldots, x_l)$, with $1 \leq x_i \leq d$ for $i = 1, \ldots, l$, if for each $i = 1, \ldots, l$, in round i the agent λ performs a move along the edge with port number x_i at its current node. By a slight extension of notation, we allow a port-labeled network to have self-loops (with exactly one port number assigned to the loop); a traversal of the self-loop is assumed not to change the location of the agent.

We say that a sequence U of integers is *(n, d)-universal* if for each node v of each regular n-node network A of degree d, an agent initially placed at v visits each node of A by following U. Aleliunas *et al.* [2] have shown non-constructively that for each $n > 0$ and $d > 0$, there exists a (n, d)-universal sequence of length $O(d^2 n^3 \log n)$ for networks with self-loops. Note that a (n, d)-universal sequence can be computed (rather inefficiently) by examining all sequences of the considered length and for each such candidate sequence one can generate all n-node port-labeled regular networks of degree d. Once a sequence U and a network A are selected, it can be tested if following U from each node of A results in visiting all nodes of A.

Given a node-labeled network G, we define the corresponding port-labeled network $A(G)$ so that there exists a bijection $\varphi\colon V(G) \to V(A(G))$ such that $\{u, v\} \in E(G)$ if and only if $\{\varphi(u), \varphi(v)\} \in E(A(G))$, and for each $\{u, v\} \in E(G)$ the port numbers at both endpoints of edge $\{\varphi(u), \varphi(v)\} \in E(A(G))$ are equal to $\sigma'(\{u, v\})$. Since, according to Lemma 4.1, σ' is an edge-coloring of G, no two edges of $A(G)$ sharing a node have the same port number at this node. Then, for each node $u \in V(G)$ we add $2n - |N_G(u)|$ loops at $\varphi(u)$ in $A(G)$. As a result, the degree of each node of $A(G)$ is $2n$, and the length of the universal sequences constructed following [2], which

we will use when exploring $A(G)$, will not exceed $O(n^5 \log n)$. In what follows, we will identify exploration of G with exploration of $A(G)$.

Theorem 4.2. *There exists an algorithm that allows any set of agents located initially at distinct nodes of any network G, and having no information about G, to compute collision-free exploration strategies of length $O(n^5 \log n)$.*

Proof. Consider an execution of Algorithm `Local-Tree-Exploration` such that the sequence $U(b)$ defined in Subsection 4.1 is replaced by a $(b, 2b)$-universal sequence. By [2], such a sequence exists and is of length $O(n^5 \log n)$. We argue that for this modified algorithm, the route $\mathcal{R}(\lambda)$ of each agent λ is a collision-free exploration strategy of G.

First, note that $\mathcal{R}(\lambda)$ is a well defined route, because, by the formulation of Algorithm `Local-Tree-Exploration`, the agent λ does check if an edge $\{v, u\}$ exists before moving from v to u in any round. Now we argue that $\mathcal{R}(\lambda)$ covers G. Consider phase $p = \lceil \log_2 n \rceil$ that consists of the rounds in which the moves of λ are determined in the p-th iteration of the 'while' loop of Algorithm `Local-Tree-Exploration`. We have that $\sigma(v) \leq b$ for each node v of G, because $b \geq n$ in this particular iteration. Let λ' be an agent that follows $U(b)$ in $A(G)$, starting at the node $\varphi(v')$ such that v' is the node occupied by λ at the beginning of phase p. By the formulation of Algorithm `Local-Tree-Exploration`, the agent λ moves from v to u in round s of phase p if and only if $\sigma'(\{u, v\}) = U_s(b)$. By the definition of $A(G)$, the port number of $\{\varphi(u), \varphi(v)\}$ at $\varphi(v)$ is $U_s(b)$. Hence, λ goes from v to u in round s of phase p if and only if λ' goes from $\varphi(v)$ to $\varphi(u)$ in $A(G)$ in round s. Since $U(b)$ is $(b, 2b)$-universal and $b \geq n$ in phase p, the route of λ in phase p covers G, regardless of the position of λ at the beginning of phase p. Finally, the fact that $\mathcal{R}(\lambda)$ is closed is due to the formulation of Algorithm `Local-Tree-Exploration` (λ backtracks its moves performed during the execution of the 'while' loop).

Let λ and λ' be two agents initially placed at distinct nodes of G. We prove that their routes $\mathcal{R}(\lambda)$ and $\mathcal{R}(\lambda')$ are collision-free. Similarly as in the proof of Lemma 4.3 one can argue that the number of phases for each agent equals $\lceil \log_2 n \rceil$. Hence, it is enough to analyze the moves of λ and λ' in an arbitrarily selected phase $p \in \{1, \ldots, \lceil \log_2 n \rceil\}$. Suppose that λ moves from v to u in round s of phase p. This implies that $\sigma'(\{u, v\}) = U_s(b)$. If λ' is located at u at the beginning of this round, then λ' moves from u to v in round s of phase p, because it also verifies that $\sigma'(\{u, v\}) = U_s(b)$. Also, two agents cannot simultaneously move from v to u and from v' to u for two different nodes v and v', because $\sigma(v) \neq \sigma(v')$ and therefore $\sigma'(\{v, u\}) = \sigma(v) + \sigma(u) \neq \sigma(v') + \sigma(u) = \sigma'(\{v', u\})$.

To complete the proof, observe that for each agent λ the length of its route is at most

$$2 \sum_{i=1}^{\lceil \log_2 n \rceil} |U(2^i)| = \sum_{i=1}^{\lceil \log_2 n \rceil} O(2^{5i} \log 2^i) = O(n^5 \log n).$$

□

5 Final remarks

We showed that, in our model, a solution to the collision-free exploration problem is always feasible, even when the agents only have local knowledge. This should be sharply contrasted with asynchronous variants of the problem (when agents do not have synchronized clocks or may perform an asynchronous meeting in the middle of an edge), in which a solution is not always feasible. This

is the case even for the fully symmetric scenario on the two-node line, where two agents starting from the two nodes cannot complete exploration without swapping, thus implying the possibility of asynchronous meeting.

References

[1] S. Albers and M. R. Henzinger. Exploring unknown environments. *SIAM J. Comput.*, 29(4):1164–1188, 2000.

[2] R. Aleliunas, R. M. Karp, R. J. Lipton, L. Lovasz, and C. Rackoff. Random walks, universal traversal sequences, and the complexity of maze problems. In *Proceedings of the 20th Annual Symposium on Foundations of Computer Science*, FOCS'79, pages 218–223, Washington, DC, USA, 1979. IEEE Computer Society.

[3] N. Alon, F. R. K. Chung, and R. L. Graham. Routing permutations on graphs via matchings. In *STOC*, pages 583–591, 1993; also SIAM J. Discrete Math. 7(3): 513-530, 1994.

[4] S. Alpern and S. Gal. *Theory of Search Games and Rendezvous*. Kluwer Acad. Publ., 2003.

[5] A. Bonato and R. Nowakowski. *The Game of Cops and Robbers on Graphs*. Amer. Math. Soc., Providence, RI, 2011.

[6] J. Edmonds. Matroids and the greedy algorithm. *Math. Programming*, 1:127–136, 1971.

[7] F. V. Fomin and D. M. Thilikos. An annotated bibliography on guaranteed graph searching. *Theor. Comput. Sci.*, 399(3):236–245, 2008.

[8] P. Fraigniaud, D. Ilcinkas, G. Peer, A. Pelc, and D. Peleg. Graph exploration by a finite automaton. *Theoretical Computer Science*, 345(2-3):331–344, 2005.

[9] M. Fürer and B. Raghavachari. Approximating the minimum-degree steiner tree to within one of optimal. *J. Algorithms*, 17(3):409–423, 1994.

[10] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.

[11] M. X. Goemans. Minimum bounded degree spanning trees. In *FOCS*, pages 273–282, 2006.

[12] D. Krizanc and L. Zhang. Many-to-one packed routing via matchings. In *COCOON*, pages 11–17, 1997.

[13] P. Panaite and A. Pelc. Exploring unknown undirected graphs. *J. Algorithms*, 33(2):281–295, 1999.

[14] G. E. Pantziou, A. Roberts, and A. Symvonis. Many-to-many routings on trees via matchings. *Theor. Comput. Sci.*, 185(2):347–377, 1997.

[15] O. Reingold. Undirected connectivity in log-space. *J. ACM*, 55(4), 2008.

[16] L. Zhang. Optimal bounds for matching routing on trees. In *SODA*, pages 445–453, 1997.