Deterministic network exploration by a single agent with Byzantine tokens

Yoann Dieudonné, Andrzej Pelc

Deterministic network exploration by a single agent with Byzantine tokens

Yoann Dieudonné [a,b,1], Andrzej Pelc [b,2]

[a] MIS, Université de Picardie Jules Verne Amiens, France
[b] Département d'informatique, Université du Québec en Outaouais, Gatineau, Québec J8X 3X7, Canada

A mobile agent has to explore an unknown network with unlabeled nodes: it must visit all nodes by walking along the links of the network, and eventually stop. If no upper bound on the size of the network is known and nodes of the network cannot be marked, then this exploration task cannot be accomplished for arbitrary networks by a deterministic terminating algorithm. On the other hand, it is feasible, if there is one unmovable token at the starting node of the agent. We investigate the exploration problem in arbitrary networks in the presence of identical unmovable tokens, some of which are Byzantine. A Byzantine token can be visible or invisible to the agent whenever the latter visits the node where the token is located, and visibility is decided by the adversary at each visit of the agent. If no upper bound on the number of tokens is known to the agent, deterministic exploration of all networks is impossible, even if all tokens are fault free. It is also impossible if all tokens are Byzantine, even if their number is known. Our main result is a deterministic exploration algorithm with cost polynomial in the (unknown) size of the network, which works in arbitrary networks, provided that the agent knows some upper bound on the total number of tokens, and that at least one token is fault free.

1. Introduction

The background. A mobile agent has to explore an unknown network with unlabeled nodes: it must visit all nodes by walking along the links of the network, and eventually stop. This task is known in the literature as node exploration with termination. It is fundamental for many network problems. The agent may need to collect data located at nodes of the network, or it may have to visit all nodes in order to check if they are functional. The task of visiting all nodes and that of traversing all edges are equivalent from the point of view of feasibility, and the latter can be accomplished using the former: after the visit of each node the agent may traverse all edges incident to it, go to visit the next node prescribed by node exploration, and so on. Thus any algorithm visiting all nodes that uses a number of edge traversals polynomial in the number of nodes can be transformed to a similar algorithm traversing all edges. The task of traversing all edges has applications, e.g., in network maintenance. Due to the above relation between the two tasks, we focus on the first. We will use the term *exploration* instead of *node exploration with termination*. The *cost* of an exploration algorithm for a given network is the worst-case number of edge traversals (called *steps*) until termination, taken over all starting nodes of the agent.

The model and the problem. The network is modeled as an undirected connected graph, referred to hereafter as a graph. The number of nodes of the graph is called its

E-mail addresses: yoann.dieudonne@u-picardie.fr (Y. Dieudonné), pelc@uqo.ca (A. Pelc).

[1] This research was done during this author's stay at the Research Chair in Distributed Computing of the Université du Québec en Outaouais as a postdoctoral fellow.

[2] Supported in part by NSERC discovery grant and by the Research Chair in Distributed Computing of the Université du Québec en Outaouais.

size. We seek exploration algorithms that do not rely on the knowledge of node labels, and can work in anonymous graphs as well (cf. [5,22]). The importance of designing such algorithms is motivated by the fact that, even when nodes are equipped with distinct labels, the agent may be unable to perceive them because of limited sensory capabilities, or nodes may refuse to reveal their labels, e.g., due to security or privacy reasons. On the other hand, we assume that edges incident to a node v have distinct labels in $\{0, \ldots, d-1\}$, where d is the degree of v. Thus every undirected edge $\{u, v\}$ has two labels, which are called its *port numbers* at u and at v. Port numbering is *local*, i.e., there is no relation between port numbers at u and at v. Note that in the absence of port numbers, edges incident to a node would be indistinguishable for the agent, and thus exploration would be often impossible, as the adversary could prevent the agent from taking some edge incident to the current node and thus block access to a part of the graph.

The agent starts at some node of the graph, chosen by the adversary. It does not know the topology of the graph or any upper bound on its size. We assume that the memory of the agent is unlimited: from the computational point of view it is modeled as a Turing machine. The agent leaves its starting node by a specified port. When the agent enters a node, it learns its port of entry and the degree of the node. Then, using its memory, it chooses the port by which it leaves the node, and repeats this process until termination.

In the absence of any additional information, exploration of an arbitrary network in this weak model is impossible. Indeed, it is impossible even in the class of oriented rings (i.e., rings in which ports 0, 1 at all nodes are situated clockwise). In any such ring, the memory of an agent executing a particular deterministic algorithm is the same after a given number of steps. Hence after some number t of steps it has to stop in every ring. This implies that exploration will not be accomplished in rings of size larger than $t+1$.

On the other hand, if the agent is equipped with a unique unmovable token situated at its starting node, which it can recognize at each visit, then exploration in arbitrary graphs is possible at a cost polynomial in the size of the graph [6]. In this paper we investigate the exploration problem in arbitrary graphs in the presence of identical unmovable tokens, each placed at a different node. Some of the tokens are Byzantine. Such a token can be visible or invisible to the agent whenever the latter visits the node where the token is located, and visibility is decided by the adversary at each visit of the agent. However, the adversary cannot change the location of the token. When the agent enters a node, it can recognize if the node contains a currently visible token, apart from learning the degree of the node and the port of entry. The presence of many identical tokens, even fault free, makes the task of the agent more difficult, as the agent may be unable to decide if it has previously seen a currently visited token. Actions of the adversary in the case of Byzantine tokens make the work of the agent even harder, due to the unpredictability of the visibility of previously seen tokens.

The scenario of many tokens has numerous applications. An agent can be a person walking in an unknown town with similarly looking streets (edges) and crossings (nodes). Some of the crossings can be distinguished from others by the presence of traffic lights (tokens). In a different application, a mobile robot exploring a labyrinth may see identical marks made at some of the corridor crossings, or a lamp can be located at each crossing but only some of the lamps may be on. Another scenario with multiple tokens occurs when many agents are present in the network, each equipped with an identical unmovable token located at its starting node, cf. [16], but agents are non-cooperative and each of them has to explore the network independently. Faulty behavior of the tokens may be due to someone maliciously erasing some of the marks in the labyrinth at unpredictable times, or to lamps switching on and off because of malfunctioning.

If no upper bound on the number of tokens is known to the agent, deterministic exploration of all networks is impossible, even if all tokens are fault free. Indeed, all nodes could be equipped with tokens, which is equivalent to no tokens at all. Observe that, e.g., knowing that some fraction $\alpha < 1$ of nodes contain tokens is of no help in solving the exploration problem. Consider all oriented rings of sizes ck, where c is a constant integer and $k = 1, 2, \ldots$, in which nodes containing tokens are evenly spaced, every c nodes. As in the case without tokens, in any such ring, the memory of an agent executing a particular deterministic algorithm is the same after a given number of steps. Hence exploration will fail in sufficiently large rings. On the other hand, if all tokens were Byzantine, then exploration of all networks would also be impossible (even if the number of tokens is known), as the adversary may decide to make all tokens invisible at all times, which is equivalent to no tokens at all. For this reason we make the following two assumptions: the agent knows some upper bound k on the total number of tokens, and at least one token is fault free, i.e., always visible. All other tokens may be Byzantine.

Our results. We present a deterministic exploration algorithm with cost polynomial in the (unknown) size of the graph, which works in arbitrary graphs, provided that the agent knows some upper bound on the total number of tokens and that at least one token is fault free. Moreover, the time of all computations of the agent is also polynomial in the size of the graph.

Related work. Algorithms for graph exploration by mobile agents (often called robots) have been intensely studied in recent literature. A lot of research is concerned with the case of a single robot exploring a graph with labeled nodes. In [1,4,5,10,14] the robot explores strongly connected directed graphs and it can move only in the direction from head to tail of an edge, not vice versa. In particular, [10] investigates the minimum time of exploration of directed graphs, and [1,14] give improved algorithms for this problem in terms of the deficiency of the graph (i.e., the minimum number of edges to be added to make the graph Eulerian). Many papers, e.g., [11–13,2] study the scenario where the explored graph is undirected

and the robot can traverse edges in both directions. In [11] the authors investigate the problem of how the availability of a map influences the efficiency of exploration. In some papers, additional restrictions on the moves of the robot are imposed. It is assumed that the robot has either a restricted tank [3], forcing it to periodically return to the base for refueling, or that it is tethered, i.e., attached to the base by a rope or cable of restricted length [13].

Exploration of anonymous graphs presents different difficulties. In this case it is impossible to explore arbitrary graphs by a single robot, if no marking of nodes is allowed. Hence the scenario adopted in [4,5] allows the use of *pebbles* which the robot can drop on nodes to recognize already visited ones, and then remove them and drop in other places. The authors concentrate attention on the minimum number of pebbles allowing efficient exploration and mapping of arbitrary directed n-node graphs. In [5] the authors compare the exploration power of one robot with a constant number of pebbles to that of two cooperating robots, and give an efficient randomized exploration algorithm for the latter scenario. In [4] it is shown that one pebble is enough, if the robot knows an upper bound on the size of the graph, and $\Theta(\log\log n)$ pebbles are necessary and sufficient otherwise. Exploration with one unmovable token marking the starting node of a single robot has been studied in [6].

In all the above papers, except [5] which deals with randomized algorithms, exploration is performed by a single robot. Deterministic exploration by many robots often assumed that moves of the robots are centrally coordinated. In [18], approximation algorithms are given for the collective exploration problem in arbitrary graphs. On the other hand, in [17] the authors study the problem of distributed collective exploration of trees of unknown topology. However, the robots performing exploration start in the same node and can directly communicate with each other. Exploration of arbitrary anonymous graphs by robots communicating through *whiteboards* has been studied in [8]. Other tasks involving many robots circulating in graphs or in terrains include rendezvous [16] and pattern formation [21].

Faulty unmovable tokens were considered in the context of the task of gathering robots at one node. In [7,15] the authors considered gathering in rings, and in [9] gathering was studied in arbitrary graphs, under the assumption that an unmovable token is located in the starting node of each agent. Tokens could disappear during the execution of the algorithm, but (as opposed to the Byzantine behavior considered in this paper), they could not reappear again.

2. Preliminaries

Let v be a node of graph G. By $succ(v,i)$ we denote the neighbor of v linked to it by the edge with port number i at v. We will use the notion of a *Universal Exploration Sequence* (UXS) defined in [19]. Let $(a_1, a_2, \ldots, a_r)$ be a sequence of integers. An *application* of this sequence to the graph G at node u is the sequence of nodes $(u_0, \ldots, u_{r+1})$ obtained as follows: $u_0 = u$, $u_1 = succ(u_0, 0)$; for any $1 \leqslant i \leqslant r$, $u_{i+1} = succ(u_i, (p + a_i) \bmod d(u_i))$, where p is the port number at u_i corresponding to the edge $\{u_i, u_{i-1}\}$. A sequence $(a_1, a_2, \ldots, a_r)$ whose application to a graph G at every node u contains all nodes of this graph is called a UXS for this graph. A UXS for the class $\mathcal{G}_n$ of graphs of size at most n is a UXS for all graphs in this class.

The following important result, based on a reduction from Koucký [19], follows from [20].

Proposition 2.1. *There exists a polynomial P, such that, for any positive integer n, there exists a UXS $R(n) = (a_1, a_2, \ldots, a_{P(n)})$ for the class $\mathcal{G}_n$ of all graphs of size at most n. This UXS can be computed in time polynomial in n.*

3. The algorithm and its analysis

We assume that the total number of tokens in the graph is at most k and that at least one token is fault free. The following notion will be crucial for our considerations. Let n be a positive integer. An application of $R(2n)$ to a graph G at some node u is called *clean*, if all nodes in this application are of degree at most $n-1$.

Algorithm Exploration-with-Byzantine-Tokens. The algorithm proceeds in phases $i = 1, 2, \ldots$. At the beginning of phase 1 the agent is at its starting node. In any phase i, the agent first applies $R(2i * P(i) * (k+1))$ at the node in which it started this phase. Let $(u_0, \ldots, u_{r+1})$ be this application. Call it the main application of this phase. If it is not clean, or if no token is seen, the agent aborts phase i and starts phase $i+1$. Otherwise, the agent backtracks to u_0, and applies $R(i)$ at each of the nodes u_j, interrupting a given application at u_j when it sees a token, every time recording the *code* of the path from u_j to this token. This code is defined as the sequence of ports encountered while walking along the path. (If, for some j, the token is at u_j, then this code is an empty sequence.) After seeing a token, the agent backtracks to u_j, goes to u_{j+1} and starts an application of $R(i)$ at this node. If either in the application of $R(i)$ at some node u_j no token is seen, or the agent has recorded more than $i * P(i)$ different codes in phase i, then the agent aborts phase i and starts phase $i+1$. Otherwise it stops upon completion of phase i.

The rest of the paper is devoted to the proof that Algorithm Exploration-with-Many-Tokens is correct and works at polynomial cost. We will use the following lemma.

Lemma 3.1. *Let $n \leqslant m$ be positive integers, and let G be a graph of size m. Let S be the sequence of nodes in G that is the application of $R(2n)$ to some node of G. If this application is clean, then S contains at least n different nodes.*

Proof. Let $S = (u_0, \ldots, u_{r+1})$. Suppose for contradiction that there are fewer than n different nodes in S, and let X be the set of these nodes. Consider any node $x \in X$. A port j at node x is called *occupied*, if for some index t, we have $x = u_t$ and either $succ(u_t, j) = u_{t-1}$ or $succ(u_t, j) = u_{t+1}$. Otherwise it is called *free*. Let d be the maximum number of free ports in any node of X. Construct the following graph H. The set of nodes of H is $X \cup \{y_1, \ldots, y_d\}$, where all y_s are distinct and do not belong to X. The set of edges of the graph H consists of all edges $\{u_t, u_{t+1}\}$ from G augmented by the following set of edges. Consider all nodes $x \in X$ in the order of their first appearance in the

sequence S. Let $c_1, \ldots, c_p$ be the free ports at x, listed in increasing order. We add edges $\{x, y_1\}, \ldots, \{x, y_p\}$ with the following ports: the port at x corresponding to the edge $\{x, y_q\}$ is c_q, and the port at y_q corresponding to the edge $\{x, y_q\}$ is the smallest port not yet used at this node. This completes the construction of graph H.

Since the application of $R(2n)$ is clean, we have $d < n$. Since the size of X is smaller than n, the graph H has fewer than $2n$ nodes. Since the size of X is smaller than m (in view of $n \leqslant m$), at least one port at some node of X is free, and consequently $d \geqslant 1$. It follows that some nodes y_s were added. Nodes y_s are not terms of the application of $R(2n)$ to the graph G at node u_0, which is exactly the sequence S. In view of Proposition 2.1, this is a contradiction with the fact that H has fewer than $2n$ nodes. □

We are now ready to prove our main result.

Theorem 3.1. *Algorithm Exploration-with-Byzantine-Tokens terminates in every graph G after a number of steps polynomial in the size of G. Upon its termination, all nodes of G are visited by the agent.*

Proof. Let m be the size of the graph G. First observe that the algorithm terminates at the latest after completion of phase m. Indeed, in this phase every application of $R(2m * P(m) * (k+1))$ must be clean and some token must be seen in this application, because at least one token is fault free. Moreover, the number of possible codes recorded by the agent cannot exceed $m * P(m)$ because the sequence $R(m)$ is applied at most m nodes and the first visible token may be at each of at most $P(m)$ nodes of an application.

Let us estimate the number of steps executed by the agent in phase i. The agent walks at most twice along the main application S of $R(2i * P(i) * (k+1))$, and at most twice along the application of $R(i)$ at each node of S. This gives a total of at most $2L(i) + 2L(i) * P(i)$ steps, where $L = P(2i * P(i) * (k+1))$ is the length of $R(2i * P(i) * (k+1))$. Hence the total number of steps made by the agent in the first m phases is polynomial in m.

It remains to show that if the agent stops upon completion of some phase $i \leqslant m$, then all nodes are visited. Consider this phase i. By the algorithm, the main application of $R(2i * P(i) * (k+1))$ made by the agent in this phase must be clean, some token must be seen in each application of $R(i)$ made in this phase, and the number of codes recorded by the agent cannot exceed $i * P(i)$. Suppose that $m \geqslant i * P(i) * (k+1)$. By Lemma 3.1, the set of nodes visited during the main application of $R(2i * P(i) * (k+1))$ in phase i is at least $i * P(i) * (k+1)$. On the other hand, by the algorithm, the agent has recorded at most $i * P(i)$ different codes. Hence, for at least $k+1$ distinct nodes visited during the main application of $R(2i * P(i) * (k+1))$, the code recorded by the agent is the same. This code cannot be empty, as this would imply that there are at least $k+1$ tokens in the graph. Hence it is non-empty, and consequently there are at least two different nodes u' and u'', for which the codes of the paths from each of these nodes to the same token are identical. This implies that there is a node w in the graph, for which edges to two of its neighbors correspond to the same port number at w, which is a contradiction. This implies that $m < i * P(i) * (k+1)$, and consequently all nodes of G are visited during any application of $R(2i * P(i) * (k+1))$, in view of Proposition 2.1. It follows that upon completion of phase i all nodes of G are visited. □

In view of Proposition 2.1, constructing the sequence $R(n)$ from [20] takes time polynomial in n. Also all other computations of the agent, such as recognizing that the application of a UXS is clean and recording and counting codes, take total time polynomial in the size of the graph. Hence not only the cost of the algorithm, i.e., the number of steps of the agent, but also the time of computations performed by the agent is polynomial in the size of the graph.

References

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

[2] C. Ambuehl, L. Gasieniec, A. Pelc, T. Radzik, X. Zhang, Tree exploration with logarithmic memory, ACM Trans. Algorithms 7 (2011), Article 17.

[3] B. Awerbuch, M. Betke, R. Rivest, M. Singh, Piecemeal graph learning by a mobile robot, in: Proc. 8th Conf. on Comput. Learning Theory, 1995, pp. 321–328.

[4] M.A. Bender, A. Fernandez, D. Ron, A. Sahai, S. Vadhan, The power of a pebble: exploring and mapping directed graphs, in: Proc. STOC 1998, pp. 269–278.

[5] M.A. Bender, D. Slonim, The power of team exploration: Two robots can learn unlabeled directed graphs, in: Proc. FOCS 1994, pp. 75–85.

[6] J. Chalopin, S. Das, A. Kosowski, Constructing a map of an anonymous graph: Applications of universal sequences, in: Proc. OPODIS 2010, pp. 119–134.

[7] S. Das, Mobile agent rendezvous in a ring using faulty tokens, in: Proc. ICDCN 2008, pp. 292–297.

[8] S. Das, P. Flocchini, S. Kutten, A. Nayak, N. Santoro, Map construction of unknown graphs by multiple agents, Theoret. Comput. Sci. 385 (2007) 34–48.

[9] S. Das, M. Mihalak, R. Sramek, E. Vicari, P. Widmayer, Rendezvous of mobile agents when tokens fail anytime, in: Proc. OPODIS 2008, pp. 463–480.

[10] X. Deng, C.H. Papadimitriou, Exploring an unknown graph, J. Graph Theory 32 (1999) 265–297.

[11] A. Dessmark, A. Pelc, Optimal graph exploration without good maps, Theoret. Comput. Sci. 326 (2004) 343–362.

[12] K. Diks, P. Fraigniaud, E. Kranakis, A. Pelc, Tree exploration with little memory, J. Algorithms 51 (2004) 38–63.

[13] C.A. Duncan, S.G. Kobourov, V.S.A. Kumar, Optimal constrained graph exploration, in: Proc. 12th Ann. ACM–SIAM Symp. on Discrete Algorithms, SODA 2001, pp. 807–814.

[14] R. Fleischer, G. Trippen, Exploring an unknown graph efficiently, in: Proc. 13th European Symp. on Algorithms, ESA 2005, pp. 11–22.

[15] P. Flocchini, E. Kranakis, D. Krizanc, F.L. Luccio, N. Santoro, C. Sawchuk, Mobile agents rendezvous when tokens fail, in: Proc. SIROCCO 2004, pp. 161–172.

[16] P. Flocchini, E. Kranakis, D. Krizanc, N. Santoro, C. Sawchuk, Multiple mobile agent rendezvous in a ring, in: Proc. LATIN 2004, pp. 599–608.

[17] P. Fraigniaud, L. Gasieniec, D. Kowalski, A. Pelc, Collective tree exploration, Networks 48 (2006) 166–177.

[18] G.N. Frederickson, M.S. Hecht, C.E. Kim, Approximation algorithms for some routing problems, SIAM J. Comput. 7 (1978) 178–193.

[19] M. Koucký, Universal traversal sequences with backtracking, J. Comput. System Sci. 65 (2002) 717–726.

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

[21] I. Suzuki, M. Yamashita, Distributed anonymous mobile robots: formation of geometric patterns, SIAM J. Comput. 28 (1999) 1347–1363.

[22] A. Ta-Shma, U. Zwick, Deterministic rendezvous, treasure hunts and strongly universal exploration sequences, in: Proc. SODA 2007, pp. 599–608.