Impact of Topographic Information on Graph Exploration Efficiency

Petrişor Panaite, Andrzej Pelc
Département d'Informatique, Université du Québec à Hull, Hull, Québec J8X 3X7, Canada

A robot has to explore an undirected connected graph by visiting all its nodes and traversing all edges. It may either have a complete *a priori* knowledge of the graph or only have an unoriented map of it, or, finally, lack any knowledge of the graph. We study the impact of this varying amount of knowledge on exploration performance. It is shown that the best exploration algorithm lacking any knowledge of the graph uses twice as many edge traversals in the worst case as does the best algorithm which has an unoriented map of the graph. On the other hand, the latter uses twice as many edge traversals in the worst case as does the best algorithm having a complete knowledge of the graph. Similar results for the restricted case of exploration algorithms working only for trees are also established. © 2000 John Wiley & Sons, Inc.

Keywords: algorithm; exploration; graph; knowledge; traversal

1. INTRODUCTION

A robot has to explore an undirected connected graph by visiting all its nodes and traversing all edges. To do so, it may have different amounts of topographic information available. The case of the most complete information occurs when the robot has a labeled map of the explored graph given as input and is supplied with a *sense of direction* (cf. [16]). By this, we mean that, arriving at any node v, the robot knows this label and knows the label of the other end of every edge incident to v.

The case of incomplete knowledge occurs when the robot has only an unlabeled map of the graph, that is, knows an isomorphic copy of it but cannot locate its position on the map and, arriving at a node, does not have any *a priori* knowledge concerning the other ends of yet unexplored edges incident to it. This knowledge is acquired during the exploration. The robot draws a partial map consisting of all nodes and edges that it has already explored. It assigns labels to these components and can recognize them when encountered again. In particular, after coming to an already explored node v incident to an explored edge e, the robot knows the location of v and of the other end of e on the partial map. At any stage of exploration, the robot also knows the number of unexplored edges incident to an explored node but does not know the other ends of these edges. It has two options for leaving an explored node: Either use a specific explored edge or an arbitrary unexplored edge (chosen by the adversary). The latter assumption corresponds to the lack of orientation in the unexplored part of the graph. It should be noted that, if the graph is highly symmetric, the robot may be unable to identify its position on the map during a large part of the exploration, or even until its very end.

Finally, the most extreme case of lack of knowledge of the explored environment occurs when the robot has no *a priori* information concerning the graph, whatsoever, not even an unlabeled map. Apart from this lack of input, the robot has the same capabilities of acquiring information on-line as under the previous scenario.

A run of an exploration algorithm is a *covering path* of the graph, that is, a path $v_0, v_1, \ldots, v_n$, where v_0 is the starting node, and every edge of the graph belongs to the path. It is natural to measure the performance of an exploration algorithm for a given graph by its longest run on this graph, that is, the number of edge traversals it uses in the worst case. Under the complete knowledge scenario, the robot can decide unambiguously which edge it chooses at every stage, and, hence, the *cost* of an exploration algorithm in this case is simply the maximum length of its run taken over all starting nodes. However, under the two other scenarios, if the robot decides to take a yet unexplored edge, the choice of the particular edge it will take is made by the adversary. Hence, under these scenarios, the cost of an exploration algorithm for a given graph is defined as the worst-case number of edge traversals it uses, taken over all starting nodes *and all adversary decisions*.

Received October 1997; accepted March 2000
Correspondence to: A. Pelc; e-mail: pelc@uqah.uquebec.ca
Contract grant sponsor: NSERC; contract grant number: OGP 0008136

The aim of the present paper was to study to what extent adding topographic information can increase the performance of graph exploration by a robot. To measure this impact, we adopt an approach similar to competitive analysis of on-line algorithms, first used in [17]. To isolate the influence of the *a priori* knowledge of the map of the graph on the performance of the robot, we introduce, for any exploration algorithm A, working without the knowledge of the map, and any graph G, an index called *map ignorance sensitivity* [$mis(A,G)$]. This is the ratio between the cost of A for G and the minimum cost for G of an exploration algorithm working with a map but without sense of direction. The supremum of $mis(A,G)$, taken over all graphs (or all graphs of some family, e.g., trees), shows how vulnerable is this algorithm to the lack of this particular knowledge (in general or with respect to a given family of graphs). Similarly, to isolate what impact the sense of direction has on exploration performance, we introduce an analogous index called *direction ignorance sensitivity* [$dis(A,G)$], for an algorithm A working with a map but without a sense of direction. $dis(A,G)$ is the ratio between the cost of A for G and the minimum cost for G of an exploration algorithm working with a map and with a sense of direction. Finding an algorithm A with the lowest possible supremum of indices *mis* or *dis*, and establishing this lowest supremum, quantifies the intrinsic value of the knowledge of the map and/or of the sense of direction in graph exploration. The analogy to competitive analysis is clear: In both cases, the performance of an algorithm lacking some crucial information is compared to the performance of the best algorithm to which this information is available. In competitive analysis, an on-line algorithm not knowing the future is compared to the best off-line algorithm that would know it.

It is not difficult to compute the cost of the following natural algorithm, namely, *depth-first search* (*DFS*). According to this strategy, the robot uses unexplored edges as long as possible, and when stuck at a node v, it relocates to the most recently visited node incident to yet unexplored edges. This relocation is done along the shortest path in the explored part of the graph. *DFS* does not use the map of the graph as input and its cost does not exceed twice the number of edges. Since the cost of any exploration algorithm is at least the number of edges in the graph, it follows that both indices *mis* and *dis* are at most 2 for *DFS*. The main result of this paper is the corresponding lower bounds: We show that there are no exploration algorithms with smaller map or direction ignorance sensitivity. We also establish an analogous result for trees. For this restricted class of graphs, the best indices *mis* and *dis* turn out to be $4/3$ rather than 2.

Previous work on exploration and navigation problems for robots concentrated on the case of an unknown environment (cf. the survey [15]). There are two basic approaches to modeling these problems. In one of them, a particular geometric setting is assumed, for example, unknown terrain with convex obstacles [6] or a room with polygonal [7] or rectangular [3] obstacles. In the second approach, the environment is modeled as a graph and the robot may only move along its edges. This latter setting was further specified in two different ways: In [1, 8], 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 [2, 5], the explored graph is undirected but an additional requirement is imposed that the robot has to come back to the starting point every so often, say for refueling. In all these papers, the aim was to explore the environment as efficiently as possible. Indeed, in [1, 2, 5, 8], such exploration algorithms were constructed and analyzed, and (nearly) matching lower bounds were also given. It turns out that, due to the requirement of traversing edges only from tail to head in one case and due to the necessity of periodic returns in the other case, the smallest possible cost is relatively high in both situations.

The above work leaves out the natural scenario of exploring an unknown undirected connected graph without the extra requirement of periodic returns to the starting point. This is the model used in this paper. It was briefly observed in [5] that the depth-first search algorithm has cost at most $2|E(G)|$ in this case. Our result shows that this simple algorithm cannot be beaten in exploring unknown graphs, under a natural measure of performance. (Cf. [13], where a different approach to evaluating exploration performance was adopted.)

The paper is organized as follows: In Section 2, we establish terminology and state some preliminary facts. Section 3 is devoted to establishing lower bounds on ignorance sensitivity indices for general graphs. In Section 4, we consider the same problem restricted to trees. Section 5 contains conclusions.

2. TERMINOLOGY AND PRELIMINARIES

Throughout the paper, *graph* means undirected connected graph. The *order* of a graph is its number of nodes. A *path of length n* in a graph is denoted by the sequence $[v_0, v_1, \ldots v_n]$ of its consecutive adjacent nodes. The length of a path P is denoted by $|P|$. A *covering path* is a path that uses all edges of the graph. A *line of length n* is any graph of order $n+1$, with two nodes of degree 1, called the *extremities* of the line, and $n-1$ nodes of degree 2. A line can be specified by its unique elementary path of length n.

A run of an exploration algorithm is a sequence of *edge traversals*, also called *moves*, where consecutive edges form a covering path in the graph. At any stage of the algorithm execution, already traversed edges are called *explored* and other edges are called *free*. A node is *saturated* if all its incident edges are explored. Otherwise, it is *free*. The robot is *stuck* after coming to node v if v becomes saturated after this move of the robot.

If an exploration algorithm A is supposed to know the *map* of the environment to be explored, that is, the graph G is an input for A, then A is said to be an *exploration algorithm with map*. Otherwise, A is said to be an *exploration algorithm without map*. Let $\mathcal{M}$ and $\mathcal{M}'$ be the set of all exploration algorithms with map and without map, respectively.

We study two classes of exploration algorithms working with incomplete topographic information: One is the class $\mathcal{M}'$, that is, exploration algorithms without map. The other is the subclass $\mathcal{M}^*$ of $\mathcal{M}$, consisting of those exploration algorithms with map that lack the *sense of direction*. By this we mean that the robot currently visiting a node v does not know other ends of free edges incident to v. Graph exploration under this assumption has been studied, for example, in [9–11].

For both of the above classes of algorithms, the robot, currently visiting a node v, has two options: It may either use a specific explored edge incident to v or decide to use a free edge incident to v. In the latter case, the choice of the free edge belongs to the *adversary*, as all these edges look alike to the robot and we are interested in worst-case performance.

Let $G = (V(G), E(G))$ be a graph, $v \in V(G)$, and let A be an exploration algorithm from $\mathcal{M}'$ or $\mathcal{M}^*$. $c_A(G, v)$ denotes the worst-case number of edge traversals, taken over all possibilities to choose the free edges (i.e., taken over all adversaries), when the robot starts at v and moves according to algorithm A. The *exploration cost of A for G* is $c_A(G) = \max\{c_A(G, v) : v \in V(G)\}$. By definition, $c_A(G) \geq |E(G)|$.

If $A \in \mathcal{M}'$, then the *map ignorance sensitivity of A for G*, denoted by *mis*(A, G), is the ratio between $c_A(G)$ and $\min_{A^* \in \mathcal{M}^*} c_{A^*}(G)$. For a family of graphs $\mathcal{U}$ (a *universe*), $mis_{\mathcal{U}}(A) = \sup_{G \in \mathcal{U}} mis(A, G)$. If $A \in \mathcal{M}^*$, then the *direction ignorance sensitivity of A for G*, denoted by *dis*(A, G), is the ratio between $c_A(G)$ and

$$\max_{v \in V(G)} \min\{|P| : P \text{ is a covering path in } G \text{ with } v \text{ as one of its ends}\}.$$

For a universe $\mathcal{U}$, $dis_{\mathcal{U}}(A) = \sup_{G \in \mathcal{U}} dis(A, G)$. When the index $\mathcal{U}$ is missing, $\mathcal{U}$ is understood to be the set of all graphs.

A natural way to explore a graph is given by the *depth-first search* (*DFS*) strategy. The *DFS* exploration algorithm is defined as follows: Every time the current node is free, the robot takes a free edge incident to the node. When stuck at a node, the robot relocates to the most recently visited free node, following a shortest path in the explored part of the graph. Clearly, $DFS \in \mathcal{M}'$.

Notice that an exploration algorithm A without map can be regarded as an algorithm with map that does not use its map. Of course, it lacks a sense of direction. Hence, the class $\mathcal{M}'$ can be considered as a subclass of $\mathcal{M}^*$. Therefore, it makes sense to consider the direction ignorance sensitivity for A. By definition, we obtain the following lemma:

Lemma 2.1. *If $A \in \mathcal{M}'$, then $mis(A, G) \leq dis(A, G)$ for every graph G.* ■

Since, for all graphs G, $c_{DFS}(G) \leq 2|E(G)|$ [5], we can state the following result:

Lemma 2.2. *$dis(DFS) \leq 2$.* ■

The following estimate will be used in the sequel.

Lemma 2.3. *If L is a line, then $c_A(L) \geq 2|L| - 1$ for every exploration algorithm $A \in \mathcal{M}^*$.*

Proof. By definition, we have to prove that there exists $v \in V(L)$ such that $c_A(L, v) \geq 2|L| - 1$. To this end, we study the behavior of the exploration algorithm A in a *frame graph*. For $n = |L|$, the frame graph is defined as the line $F = [v_{-(n-1)}, \ldots, v_{-1}, v_0, v_1, \ldots, v_{n-1}]$. For $k \in \{1, \ldots, n-1\}$, let $L_k = [v_{-k}, \ldots, v_0, \ldots, v_{n-k}]$, that is, a line of length n. Notice that the existence of $v \in V(L)$ such that $c_A(L, v) \geq 2|L| - 1$ is equivalent to the existence of $k \in \{1, \ldots, n-1\}$ such that $c_A(L_k, v_0) \geq 2n - 1$.

Place the robot at node v_0 and run A in F. After any move, let l and r denote, respectively, the smallest and the largest indices, among all indices i for which v_i is a visited node. Stop the execution of A at the first moment t_0 when $r - l = n - 1$. If, at this moment, the robot is at v_l, let $k = l$. Otherwise, that is, if the robot is at v_r, let $k = l - 1$. Notice that, in both cases, if the robot starts at v_0 and moves according to A in L_k, after t_0 moves, it is at one of the extremities of L_k and there exists exactly one free edge, incident to the other extremity. It follows that $c_A(L_k, v_0) \geq t_0 + n$. Since $t_0 \geq n - 1$, we conclude that $c_A(L_k, v_0) \geq 2n - 1$. ■

3. GENERAL CASE

In this section, we prove that the best map (respectively, direction) ignorance sensitivity that an exploration algorithm in the class $\mathcal{M}'$ (respectively, in the class $\mathcal{M}^*$) can have is 2. To this end, we need to define some families of graphs.

An *intersection with center v* is a tree of height 2 with root v, where v has degree 4 and it has a child with i children, for each $i \in \{1, 2, 3, 4\}$. See Figure 1(a). Let $\alpha'(v)$ [respectively, $\beta'(v), \gamma'(v), \delta'(v)$] be the child of v with 2 (respectively, 3, 4, 1) children. Let $\alpha(v)$ denote one of the children of $\alpha'(v)$. $\beta(v), \gamma(v)$, and $\delta(v)$ are defined analogously.

Let $n \geq 3, k = \lfloor \log_2 n \rfloor + 1, i \in \{0, \ldots, n-1\}$, and let $b = b_1 \ldots b_k$ be the binary representation of i with k digits. An *(n, i)-code of root v* is a tree obtained from

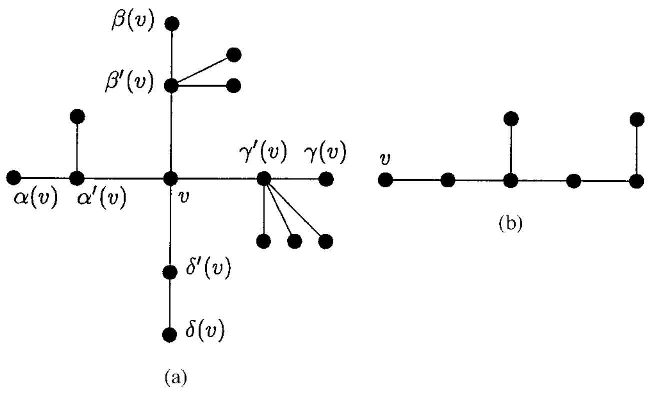

FIG. 1. (a) Intersection with center v; (b) (11, 5)-code of root v.

a line $[v, x_1, \ldots, x_k]$ by adding an edge $[x_h, y_h]$ for every $h \in \{1, \ldots, k\}$ with $b_h = 1$. See Figure 1(b).

For every $n \geq 3$, we introduce the family $\mathcal{S}_n$ of *n-ray suns*. For $i \in \{0, \ldots, n-1\}$, let X_i be an intersection with center v_i, let H_i be a (n, i)-code of root u_i, let P_i be a line of length n with its extremities denoted by l_i and r_i, and let R_i be a line of any length with one of its extremities denoted by w_i. These $4n$ graphs are considered to be pairwise disjoint. An n-ray sun is constructed as follows (see Fig. 2):

For every $i \in \{0, \ldots, n-1\}$,

- X_i and H_i are connected by $[\delta(v_i), u_i]$;
- X_i and R_i are connected by $[up, w_i]$, where $up \in \{\alpha(v_i), \beta(v_i), \gamma(v_i)\}$;
- X_i and P_i are connected by $[right, l_i]$, where

$$right = \begin{cases} \gamma(v_i) & \text{if } up \neq \gamma(v_i) \\ \beta(v_i) & \text{if } up = \gamma(v_i); \end{cases}$$

- X_i and P_{i-1} are connected by $[left, r_{i-1}]$, where

$$left = \begin{cases} \alpha(v_i) & \text{if } up \neq \alpha(v_i) \\ \beta(v_i) & \text{if } up = \alpha(v_i). \end{cases}$$

The index arithmetic is modulo n. Notice that every n-ray sun S is a connected graph with exactly one cycle. The paths P_i and R_i are called the i^{th} *segment* and the i^{th} *ray* of S, respectively. Notice that the length of a ray does not depend on n.

Theorem 3.1. *For every exploration algorithm $A \in \mathcal{M}', mis(A) \geq 2$.*

Proof. We prove that, for every exploration algorithm $A \in \mathcal{M}'$, there exists a sequence $(S_n)_{n \geq 3}$ of n-ray suns such that $\lim_{n \to \infty} mis(A, S_n) \geq 2$.

First, we show that there exists an exploration algorithm $A^* \in \mathcal{M}^*$, for which $c_{A^*}(S) \leq 2m'(S) + n^2 + a'n \log n$ for every n-ray sun S, where $m'(S)$ denotes the sum of ray lengths in S and a' is a constant independent of S. We define A^* as a sequence of $n+1$ phases. In Phase 0, the robot looks for an intersection center by following the DFS strategy. By construction, this intersection center can be defined as the first visited node of degree 4 that has no neighbor of degree 1. The search for it takes at most $n + O(1)$ moves if the starting node does not belong to a ray. Suppose that the starting node belongs to a ray R. If the first move is toward the closest intersection center, then the robot needs at most $|R| + O(1)$ moves to reach that node; otherwise, it needs at most $2|R| + O(1)$ moves. Notice that, in the latter situation, all edges of the ray are explored.

Upon arriving at the first intersection center v_{i_0}, the robot explores all edges of X_{i_0} and those of H_{i_0}, spending $O(\log n)$ moves. This is possible since the robot knows that H_{i_0} can be accessed through the node $\delta(v_{i_0})$ [which is connected to $\delta'(v_{i_0})$, the unique neighbor of v_{i_0} of degree 2]. By learning H_{i_0}, the robot locates v_{i_0} on the input map. Using the map, it can identify through which nodes the intersection X_{i_0} is connected with its ray (i.e., R_{i_0}) and with its left and right segments (i.e., P_{i_0-1} and P_{i_0}). After that, Phase 0 is ended by exploring all edges of P_{i_0} and reaching v_{i_0+1}.

Suppose that the robot starts a new phase at the intersection $v_i, i \neq i_0$. The robot explores all edges of X_i, H_i, R_i, and P_i in this order. The current phase ends when v_{i+1} is reached. This can be done in $O(\log n) + 2|R_i| + n$ moves. When v_{i_0} is reached again, the robot starts Phase n. The exploration of S is already accomplished if, in Phase 0, all edges of R_{i_0} were explored. If this is not the case, then a trip along R_{i_0} will end the execution of A^*. In both cases, the total number of moves along R_{i_0}, performed in Phase 0 and Phase n, is at most $2|R_{i_0}|$. Therefore, we can conclude that the total number of moves performed during the execution of A^* in S is at most $2m'(S) + n^2 + a'n \log n$, where a' is a constant independent of S.

Consider an exploration algorithm $A \in \mathcal{M}'$. To prove the theorem, it is sufficient to show that, for every $n \geq 3$, there exists an n-ray sun S such that $c_A(S) \geq 4m'(S) + 2n^2 - a''n$, where a'' is a constant independent of S. We construct S dynamically, according to the decisions taken by the robot in intermediate graphs (or *frame graphs*) denoted by $F_0, F_1, \ldots, F_{n-2}$ and defined as follows:

For each $i \in \{0, \ldots, n-1\}$, let X_i be an intersection with center v_i and let H_i be an (n, i)-code of root u_i. Connect X_i with H_i by $[\delta(v_i), u_i]$. Now, let $Q_j, j \in \{1, \ldots, 5\}$

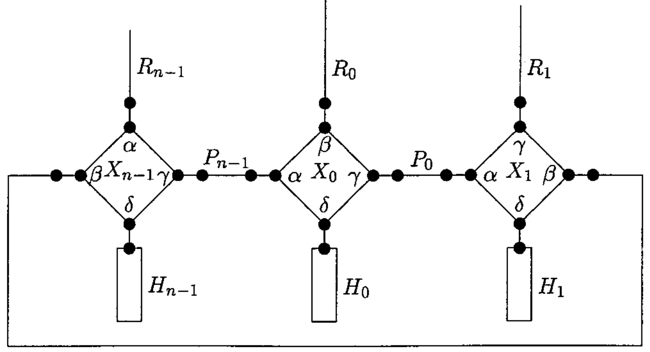

FIG. 2. An n-ray sun.

be disjoint lines of length n; let p_j and q_j denote the extremities of Q_j. The frame graph F_0 is obtained by connecting X_0 to Q_1 by $[\alpha(v_0), p_1]$, to Q_2 by $[\beta(v_0), p_2]$, and to Q_3 by $[\gamma(v_0), p_3]$. Place the robot at node v_0 and start the exploration of the graph F_0 by following the algorithm A. End the execution of A at the moment t_0 when the robot reaches, for the first time, a node $q \in \{q_1, q_2, q_3\}$. At this point, we extend F_0 to F_1 as follows:

If $q = q_3$, then Q_3 is further connected to X_1 by $[q_3, \alpha(v_1)]$. X_1 is connected to Q_4 by $[\beta(v_1), p_4]$ and to Q_5 by $[\gamma(v_1), p_5]$. Let $Q'_1 = Q_1, Q'_2 = Q_2, Q'_3 = Q_4$, and $Q'_4 = Q_5$. If $q = q_2$, we do similar connections by interchanging Q_2 and Q_3. If $q = q_1$, then Q_1 is further connected to X_{n-1} by $[q_1, \gamma(v_{n-1})]$. X_{n-1} is connected to Q_4 by $[\alpha(v_{n-1}), p_4]$ and to Q_5 by $[\beta(v_{n-1}), p_5]$. Let $Q'_1 = Q_4, Q'_2 = Q_5, Q'_3 = Q_2$, and $Q'_4 = Q_3$. The resulting graph is denoted by F_1. The two intersections, X_0 and X_1 (or X_0 and X_{n-1}), are called *outposts*. The four lines Q'_1, Q'_2, Q'_3, and Q'_4, connected to outposts, are called *external lines*. We will show later how two outposts and four external lines are defined for each frame graph $F_i, i \in \{2, \ldots, n-2\}$.

Since F_0 is a subgraph of F_1, if the robot starts at v_0 and explores F_1 according to A, then the sequence of edges traversed during the first t_0 steps is the same as the one traversed in F_0.

Let $i \in \{2, \ldots, n-2\}$ and suppose that F_{i-1} is already constructed. Let X_l and X_r be the outposts of F_{i-1}, where $l, r \in \{0, \ldots, n-1\}$ and $l \neq r$. Let $Q_j, j \in \{1,2,3,4\}$, be the external lines of F_{i-1}, with p_j and q_j as corresponding extremities. Q_1 and Q_2 are supposed to be connected to X_l by $[p_1, \alpha(v_l)]$ and $[p_2, \beta(v_l)]$, respectively. Q_3 and Q_4 are supposed to be connected to X_r by $[p_3, \beta(v_r)]$ and $[p_4, \gamma(v_r)]$, respectively. Place the robot at node v_0 and start the exploration of the graph F_{i-1} according to algorithm A. End the execution of A at the moment t_{i-1} when the robot reaches, for the first time, a node $q \in \{q_1, q_2, q_3, q_4\}$. At this point, we extend F_{i-1} to F_i as follows:

Suppose that $q = q_4$. The other three cases are similar. Connect Q_4 to X_{r+1} by $[q_4, \alpha(v_{r+1})]$. The line Q_4 represents the r^{th} segment in the n-ray sun S that we want to construct. Let $Q'_3 = [p_3, \ldots, q'_3, q''_3]$ be the prefix of Q_3 for which $[p_3, \ldots, q'_3]$ is the explored part of Q_3. Hence, the edge $[q'_3, q''_3]$ is still free. Such an edge exists since q_3 was not yet visited. Delete all the vertices in Q_3 between q''_3 and q_3, q_3 included if $q_3 \neq q''_3$. The line Q'_3 represents the r^{th} ray in the n-ray sun S under construction. Let Q_5 and Q_6 be two disjoint lines of length n such that $V(Q_j) \cap V(F_{i-1}) = \emptyset, j \in \{5,6\}$; let p_j and q_j denote the extremities of Q_j. Connect X_{r+1} to Q_5 by $[\beta(v_{r+1}), p_5]$ and to Q_6 by $[\gamma(v_{r+1}), p_6]$. The resulting graph is the next frame graph F_i. X_l and X_{r+1} are the outposts of F_i. Q_1, Q_2, Q_5, and Q_6 are the external lines of F_i.

Since F_{i-1} is a subgraph of F_i, if the robot starts at v_0 and explores F_i according to A, the sequence of edges traversed during the first t_{i-1} steps is the same as the one traversed in F_{i-1}.

Let X_l and X_r be the outposts of F_{n-2}, and let $Q_j, j \in \{1,2,3,4\}$, be the external lines of F_{n-2}; the extremities of Q_j are p_j and q_j. Notice that, in the construction of F_{n-2}, we have used $n-1$ intersections. Therefore, $r+2 \equiv l \pmod{n}$. Connect Q_4 to X_{r+1} by $[q_4, \alpha(v_{r+1})]$ and X_{r+1} to Q_1 by $[\gamma(v_{r+1}), q_1]$. Add a new node and connect it to $\beta(v_{r+1})$ [i.e., the $(r+1)^{th}$ ray is of length 0]. Notice that the resulting graph is an n-ray sun, each ray being of length at most n. Let S be this graph.

By construction, if the robot starts at v_0 and explores S following algorithm A, the sequence of edges traversed during the first t_i steps is the same as the one traversed in F_i, where $i \in \{1, \ldots, n-3\}$. Therefore, at each moment t_i, we can count $n + 2(|R| - 1)$ new moves, where R is the ray defined when F_i was extended to F_{i+1}. Notice that, at the moment t_i, R has an unexplored edge and the robot is at least at distance n from the center of intersection connected to R. Consequently, to explore the last edge in R and then to leave R (if R is not the last explored ray), $n + 2|R|$ new moves are necessary. These moves can be counted independently of the other rays. Hence, the exploration of each ray R involves at least $2n - 2 + 4|R|$ moves. There are at most four rays in S for which the above reasoning may not apply: the last explored ray and the l^{th}, the r^{th}, and the $(r+1)^{th}$ ray, where l and r are the indices of the outposts of F_{n-2}. The length of any such ray does not exceed n. We conclude that, when the starting point is v_0, the number of moves necessary to explore S by algorithm A is at least $(n-4)(2n-2) + 4(m'(S) - (n+n+n+0)) = 2n^2 + 4m'(S) - 22n + 8$. ∎

We now turn our attention to the lower bound on direction ignorance sensitivity. To this end, we define the following family of graphs. A *thick line L of length n* is a graph defined by $V(L) = \{v_0, v_1, \ldots, v_n, x_1, \ldots, x_n, y_1, \ldots, y_n\}$ and $E(L) = \{[x_i, v_{i-1}], [x_i, v_i], [y_i, v_{i-1}], [y_i, v_i] : 1 \leq i \leq n\}$. See Figure 3. The nodes v_0 and v_n are called the *extremities* of L. For $i \in \{0, \ldots, n-1\}$, the cycle $[v_i, x_{i+1}, v_{i+1}, y_{i+1}, v_i]$ is called the *cycle connecting v_i and v_{i+1}*. We denote L by $v_0 \diamond v_1 \diamond \cdots \diamond v_n$.

Notice that a thick line of length n is an Eulerian graph with $4n$ edges.

Theorem 3.2. *For every exploration algorithm $A \in \mathcal{M}^*, dis(A) \geq 2$.*

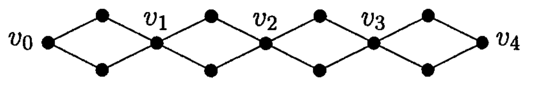

FIG. 3. A thick line of length 4.

Proof. By definition, it is sufficient to show that, for every $n \geq 1$, there exists an Eulerian graph G_n with $f(n)$ edges such that $c_A(G_n) \geq 2f(n) - a$, where f is an unbounded function and a is a constant independent of n.

All graphs G_n will be thick lines. As in the case of (normal) lines (see the proof of Lemma 2.3), G_n are defined according to the moves of the robot in a frame graph F. Let $n \geq 1$ and let $F = v_{-(n-1)} \diamond \cdots \diamond v_{-1} \diamond v_0 \diamond v_1 \diamond \cdots \diamond v_{n-1}$. For $k \in \{1, \ldots, n-1\}$, let L_k denote the thick line $v_{-k} \diamond \cdots \diamond v_0 \diamond \cdots \diamond v_{n-k}$ of length n. Since every L_k is an Eulerian graph with $4n$ edges, it is sufficient to prove that there exists $k \in \{1, \ldots, n-1\}$ such that $c_A(L_k, v_0) \geq 8n - a$, where a is a constant independent of n.

Place the robot at the node v_0 and run A in F.

Claim. There exists an adversary $\mathscr{A}$ such that, when the robot reaches v_k (respectively, v_{-k}) for the first time, $k \in \{2, \ldots, n-1\}$, at least six moves have already been performed along the edges of the cycle connecting v_{k-2} and v_{k-1} [respectively, $v_{-(k-2)}$ and $v_{-(k-1)}$].

An adversary is defined by the free edges that it chooses at the moments when the robot is at a free node and the running algorithm decides to use a free edge. We define the adversary $\mathscr{A}$ as the last one in a sequence $\mathscr{A}_0, \ldots, \mathscr{A}_{2n-1}$ of adversaries constructed iteratively as follows:

Let $\mathscr{A}_0$ be any adversary for algorithm A running in F, with v_0 as the starting node. Suppose that $\mathscr{A}_i, i \in \{0, \ldots, 2n-2\}$, is already defined. We modify $\mathscr{A}_i$ to $\mathscr{A}_{i+1}$ using the behavior of algorithm A. Let $t_0^i < t_1^i < \cdots < t_{2n-2}^i$ be the moments at which the robot reaches a node in $\{v_{-(n-1)}, \ldots, v_0, \ldots, v_{n-1}\}$ for the first time. These moments depend on A and $\mathscr{A}_i$. Notice that $t_0^i = 0$; at the moment t_1^i, the robot reaches for the first time either v_{-1} or v_1; at the moment t_2^i, the robot reaches for the first time a node in $\{v_{-2}, v_{-1}, v_1, v_2\}$, and so on. Let v_k be the node reached at the moment t_{i+1}^i. Suppose that $k > 0$. The other case, $k < 0$, is similar. If $k = 1$, then let $\mathscr{A}_{i+1} = \mathscr{A}_i$. Consider now the case $k \geq 2$. Let $t' = t_i^i$. Notice that, at the moment t', the robot reaches v_{k-1} for the first time. Let $t'' = \max\{t \leq t_{i+1}^i :$ the robot reaches v_{k-1} at the moment $t\}$. In other words, t'' is the last moment, before reaching v_k, when the robot is at v_{k-1}. Let C be the cycle connecting v_{k-2} and v_{k-1}. If the robot reaches again v_{k-2} at a moment t, where $t' < t < t''$, then let $\mathscr{A}_{i+1} = \mathscr{A}_i$. Notice that, in this case, at least six moves are already performed along the edges of C when the robot reaches v_k. Hence, it remains to consider the case when the robot does not reach v_{k-2} between the moments t' and t''. Let P and Q be the two paths of length 2 connecting v_{k-2} and v_{k-1}. Without loss of generality, suppose that the robot arrives for the first time at v_{k-1} by traversing the path P. Let y be the node between v_{k-2} and v_{k-1} on the path Q. Notice that, at the moment t', the edge $[v_{k-1}, y]$ is free. We consider two cases:

Case 1. *The edge* $[v_{k-2}, y]$ *is free*. See Figure 4(a). Since the robot reaches v_k at the moment $t_{i+1}^i > t'$, the running of algorithm A between the moments t' and t_{i+1}^i involves the exploration of a path of length 2, starting at v_{k-1}. We can change the adversary $\mathscr{A}_i$ so that this path is the path Q. Notice that we can do it by changing only the behavior of $\mathscr{A}_i$ after the moment t'. The new adversary obtained in this way is denoted by $\mathscr{A}_{i+1}$. To reach v_k for the first time, under the adversary $\mathscr{A}_{i+1}$, the robot performs at least six moves along the edges of C.

Case 2. *The edge* $[v_{k-2}, y]$ *is already explored*. See Figure 4(b). Notice that, at the moment t', at least four moves are already performed along the edges of C. Two supplementary moves are obtained by changing $\mathscr{A}_i$ to $\mathscr{A}_{i+1}$, so that the first exploration of a free edge incident to v_{k-1}, after the moment t', is the edge $[v_{k-1}, y]$.

By induction on $i \in \{0, \ldots, 2n-2\}$, it follows that, for every $j \leq i$, if the robot moves according to A and $\mathscr{A}_i$ and, at the moment t_j^i, it reaches v_k (respectively, v_{-k}) for the first time, $k \in \{2, \ldots, n-1\}$, then at least six moves are performed along the edges of the cycle connecting v_{k-2} and v_{k-1} [respectively, $v_{-(k-2)}$ and $v_{-(k-1)}$] during the first t_j^i steps.

We conclude that if the robot moves according to algorithm A and the adversary $\mathscr{A} = \mathscr{A}_{2n-2}$, whenever it reaches a node v_k (respectively, v_{-k}) for the first time, $k \in \{2, \ldots, n-1\}$, at least six moves are performed along the edges of the cycle connecting v_{k-2} and v_{k-1} [respectively, $v_{-(k-2)}$ and $v_{-(k-1)}$]. This concludes the proof of the claim.

During the run of A in the frame graph F, under the adversary $\mathscr{A}$, let l and r denote, respectively, the smallest and the largest indices, among all indices i for which v_i is a visited node. Stop the execution of A at the first moment t when $r - l = n - 1$. If, at this moment, the robot is at v_l, let $k = l$. Otherwise, that is, if the robot is at v_r, let $k = l - 1$. Notice that, in both cases, if the robot starts at v_0 and moves according to A in L_k, after t moves, it is at one of the extremities of L_k and there exists an unexplored cycle of length 4 containing the other extremity. It follows that $c_A(L_k, v_0) \geq t + 2n + 2$. At the moment t, there are n visited nodes v_i in L_k. By the claim, we get $t \geq 6(n-3) + 2 \cdot 2$. We conclude that $c_A(L_k, v_0) \geq 8n - 12$. ∎

Theorems 3.1 and 3.2, together with Lemmas 2.1 and 2.2, imply that *DFS* has the best possible map ignorance sensitivity among all exploration algorithms in class $\mathscr{M}'$, and—considered as an algorithm in $\mathscr{M}^*$—it has the best

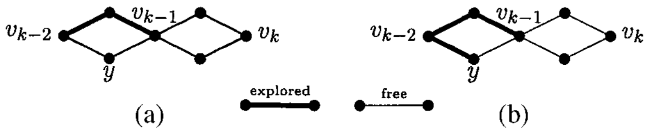

FIG. 4. The explored edges in C at the moment t'.

possible direction ignorance sensitivity among all exploration algorithms in class $\mathscr{M}^*$.

4. TREES

In this section, we restrict our attention to algorithms exploring trees. We show that the best map (respectively, direction) ignorance sensitivity that an exploration algorithm in class $\mathscr{M}'$ (respectively, in class $\mathscr{M}^*$) can have for the universe of trees is $\frac{4}{3}$. Moreover, *DFS* turns out to achieve this bound in both cases.

Let $\mathscr{T}$ denote the set of all trees.

Lemma 4.1. *$mis_{\mathscr{T}}(DFS) \le \frac{4}{3}$ and $dis_{\mathscr{T}}(DFS) \le \frac{4}{3}$.*

Proof. By Lemma 2.1, it is sufficient to show that $dis_{\mathscr{T}}(DFS) \le \frac{4}{3}$. Let T be a tree of order n, let v be a vertex in T, and let P_v be a longest elementary path in T, with v as one of its extremities. A shortest covering path in T starting at v has the length $2|E(T)| - |P_v|$. Let $P = [v_0, v_1, \ldots, v_k]$ be a longest elementary path in T. Then, for $k' = \lceil k/2 \rceil, P_{v_{k'}} = [v_0, v_1, \ldots, v_{k'}]$ is a longest elementary path in T with $v_{k'}$ as one of its extremities. It follows that a shortest covering path in T starting at $v_{k'}$ has the length $2|E(T)| - |P_{v_{k'}}| \ge 2|E(T)| - \lceil \frac{|E(T)|}{2} \rceil \ge 2(n-1) - \frac{n}{2} = \frac{3}{2}n - 2$. Since $c_{DFS}(T) \le 2|E(T)| = 2n - 2$, it follows that $dis(DFS, T) \le (2n-2)/(\frac{3}{2}n - 2)$. Consequently, $dis_{\mathscr{T}}(DFS) \le \frac{4}{3}$. ■

To establish our lower bounds, we introduce the family of *encoded lines* as follows: Let $n \ge 2$ and $1 \le k \le n-1$. Let P be a line of length $n^2 + 2n$ with w as one of its extremities. For $i \in \{-k, \ldots, -1, 0, 1 \ldots, n-k\}$, let v_i denote the node of P at distance $(i+k+1)n$ from w. Let H_i denote a $(2n, i')$-code of center u_i (see the definition given in Section 3), where

$$i' = \begin{cases} 2|i| & \text{if } i \le 0 \\ 2i - 1 & \text{if } i > 0. \end{cases}$$

Let L be the graph obtained from P and H_i, $i \in \{-k, \ldots, 0, \ldots, n-k\}$, by adding the edges $[u_i, v_i]$. The graph L is an *encoded line of length n* and will be denoted by $v_{-k} \perp \cdots \perp v_0 \perp \cdots \perp v_{n-k}$. L is said to be *k-starting*. The nodes v_{-k} and v_{n-k} are called the *extremities* of L. See the example given in Figure 5.

Theorem 4.1. *For every exploration algorithm $A \in \mathscr{M}', mis_{\mathscr{T}}(A) \ge \frac{4}{3}$.*

Proof. We prove that, for every $A \in \mathscr{M}'$, there exists a sequence $(G_n)_{n \ge 2}$ of encoded lines of length n such that $\lim_{n \to \infty} mis(A, G_n) \ge \frac{4}{3}$.

First, we show that there exists an exploration algorithm $A^* \in \mathscr{M}^*$, for which $c_{A^*}(L) \le \frac{3}{2}n^2 + a'n \log n$ for every encoded line L of length n, where a' is a constant

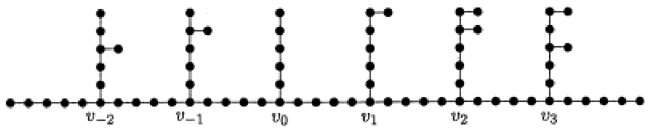

FIG. 5. A 2-starting encoded line of length 5.

independent of L. Let $L = v_{-k} \perp \cdots \perp v_0 \perp \cdots \perp v_{n-k}$; suppose that the robot starts at a node v in L. The algorithm A^* is defined as follows:

If $n \le 6$, the robot uses *DFS*. Notice that the map specifies the value of n. Suppose that $n > 6$. Any subgraph in L, defined by a $(2n, i')$-code and the edge which connects it to the corresponding node v_i, does not contain an elementary path of length $\lfloor \log_2(2n) \rfloor + 4$. On the other hand, since $n \ge 7, n \ge \lfloor \log_2(2n) \rfloor + 4$. This helps the robot to figure out if it is or not within a $(2n, i')$-code.

In the beginning, the robot looks for a node v_{i_0} by following the *DFS* strategy. More exactly, the robot moves until it reaches a node of degree 3 that has no neighbor of degree 1. The search for it takes at most $n + O(1)$ moves. Upon arriving at v_{i_0}, the robot explores all edges of H_{i_0}, spending $O(\log n)$ moves. This is possible since the robot knows n: When it is at distance $\lfloor \log_2(2n) \rfloor + 2$ or $\lfloor \log_2(2n) \rfloor + 3$ from v_{i_0}, the robot can decide if it is or not within H_{i_0}. If it took the good direction from v_{i_0}, that is, if it used the edge $[v_{i_0}, u_{i_0}]$, the robot explores H_{i_0} and returns at v_{i_0} after $2(|E(H_{i_0})| + 1) = O(\log n)$ moves. If it took the wrong direction from v_{i_0}, the robot returns to v_{i_0} by performing at most $2(\lfloor \log_2(2n) \rfloor + 3)$ more moves than in the first case. After the exploration of H_{i_0}, the robot reaches v_{i_1} using n moves, where i_1 is either $i_0 - 1$ or $i_0 + 1$. It explores H_{i_1} and returns at v_{i_1} using $O(\log n)$ moves. At this moment, the robot learned the values of i_0 and i_1. Hence, it obtained its current position on the map and the positions of the other nodes v_i, with respect to v_{i_1}, in the "real world." The number of moves already performed is $n + O(\log n)$.

Suppose that $i_1 = i_0 - 1$. The other case is similar. Notice that, since the robot has the map, it also knows k. If $i_1 + k \le n - k - i_1$, that is, if the robot is closer to v_{-k} than to v_{n-k}, then it will visit the nodes v_i and the corresponding H_i (if yet unexplored) in the order $i_1 - 1, \ldots, -k, \ldots, i_1 - 1, i_1, \ldots, n-k$. Otherwise, that is, if the robot is closer to v_{n-k} than to v_{-k}, then it will visit the nodes v_i and the corresponding H_i (if yet unexplored) in the order $i_1 + 1, \ldots, n-k, \ldots, i_1 + 1, i_1, \ldots, -k$. Suppose that $i_1 + k \le n - k - i_1$ (the other case is similar). The total number of moves performed to explore L from v is at most $n + O(\log n) + 2(i_1 + k)(n + O(\log n)) + (n - k - i_1)(n + O(\log n)) = (n + i_1 + k)(n + O(\log n)) \le (n + \frac{n}{2})(n + O(\log n)) = \frac{3}{2}n^2 + O(n \log n)$.

Now, let A be an exploration algorithm without a map. By definition, it is sufficient to show that, for every $n \ge 2$, there exists an encoded line G_n of length n such that $c_A(G_n) \ge 2n^2 - an$, where a is a constant independent of G_n.

Again, as in the case of (normal) lines (see the proof of Lemma 2.3), G_n are defined according to the moves of the robot in a frame graph F. Let $n \geq 1$ and let

$$F = v_{-(n-1)} \perp \cdots \perp v_{-1} \perp v_0 \perp v_1 \perp \cdots \perp v_{n-1}.$$

For $k \in \{1, \ldots, n-1\}$, let $L_k = v_{-k} \perp \ldots \perp v_0 \perp \cdots \perp v_{n-k}$, that is, L_k is a k-starting encoded line of length n. Place the robot at the node v_0 and run A in F.

During the run of A in the frame graph F, let l and r denote, respectively, the smallest and the largest indices, among all indices i for which v_i is a visited node. Stop the execution of A at the first moment t when $r - l = n - 1$. If, at this moment, the robot is at v_l, let $k = l$. Otherwise, that is, if the robot is at v_r, let $k = l - 1$. Notice that, in both cases, if the robot starts at v_0 and moves according to A in L_k, after t moves, it is at one of the extremities of L_k and there exists a free edge incident to the other extremity. It follows that $c_A(L_k, v_0) \geq t + n^2$. Since, at the moment t, n nodes v_i are visited, $t \geq (n-1)n$. We conclude that $c_A(L_k, v_0) \geq 2n^2 - n$. ∎

Theorem 4.2. *For every exploration algorithm* $A \in \mathcal{M}^*, dis_{\mathcal{T}}(A) \geq \frac{4}{3}$.

Proof. Let $n \geq 1$ and let L be a line of length n. By Lemma 2.3, $c_A(L) \geq 2n - 1$. For every node v in L, the length of a shortest covering path in L starting at v is at most $n + \lfloor n/2 \rfloor \leq \frac{3}{2}n$. By definition, $dis_{\mathcal{T}}(A) \geq \lim_{n\to\infty}(2n-1)/(\frac{3}{2}n) = \frac{4}{3}$. ∎

The results of this section imply a corollary concerning the optimality of *DFS* in the universe of trees, similar to that formulated in the general case. Again, *DFS* has the best possible map ignorance sensitivity among all exploration algorithms in class $\mathcal{M}'$, and—considered as an algorithm in $\mathcal{M}^*$—it has the best possible direction ignorance sensitivity among all exploration algorithms in class $\mathcal{M}^*$. The difference concerns only the values of these best possible indices: In the case of trees, they both equal $\frac{4}{3}$, while in the general case, they both equal 2.

5. CONCLUSIONS

We investigated the impact of the amount of topographic information available to an algorithm exploring a graph on its performance. To measure this impact, we adopted an approach similar to competitive analysis. The worst-case performance of an algorithm lacking a particular piece of knowledge was compared to the performance of an algorithm supplied with this information. We focused attention on two types of information: the topology of the explored graph and the sense of direction. This yields two indices of performance of exploration algorithms: map ignorance sensitivity and direction ignorance sensitivity. The main results of this paper are lower bounds on these indices which match the performance of exploration by depth-first search.

REFERENCES

[1] S. Albers and M. R. Henzinger, Exploring unknown environments, Proc 29th Symp on Theory of Computing, 1997, pp. 416–425.

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

[3] E. Bar-Eli, P. Berman, A. Fiat, and R. Yan, On-line navigation in a room, Proc 3rd ACM–SIAM Symp on Discrete Algorithms, 1992, pp. 237–249.

[4] P. Berman, A. Blum, A. Fiat, H. Karloff, A. Rosen, and M. Saks, Randomized robot navigation algorithms, Proc 7th ACM–SIAM Symp on Discrete Algorithms, 1996, pp. 74–84.

[5] M. Betke, R. Rivest, and M. Singh, Piecemeal learning of an unknown environment, Proc 5th Conf on Computer Learning Theory, 1993, pp. 277–286.

[6] A. Blum, P. Raghavan, and B. Schieber, Navigating in unfamiliar geometric terrain, Proc 23rd Symp on Theory of Computing, 1991, pp. 494–504.

[7] X. Deng, T. Kameda, and C. H. Papadimitriou, How to learn an unknown environment, Proc 32nd Symp on Foundations of Computer Science, 1991, pp. 298–303.

[8] X. Deng and C. H. Papadimitriou, Exploring an unknown graph, Proc 31st Symp on Foundations of Computer Science, 1990, pp. 356–361.

[9] K. Diks, E. Kranakis, and A. Pelc, Broadcasting in unlabeled tori, Par Proc Lett 8 (1998), 177–188.

[10] K. Diks, E. Kranakis, and A. Pelc, Perfect broadcasting in unlabeled networks, Discr Appl Math 87 (1999), 33–47.

[11] K. Diks, S. Dobrev, E. Kranakis, A. Pelc, and P. Ruzicka, Broadcasting in unlabeled hypercubes with linear number of messages, Info Process Lett 66 (1998), 181–186.

[12] F. Hoffmann, C. Icking, R. Klein, and K. Kriegel, A competitive strategy for learning a polygon, Proc 8th ACM–SIAM Symp on Discrete Algorithms, 1997, pp. 166–174.

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

[14] C. H. Papadimitriou and M. Yannakakis, Shortest paths without a map, Theor Comput Sci 84 (1991), 127–150.

[15] N. S. V. Rao, S. Hareti, W. Shi, and S. S. Iyengar, Robot navigation in unknown terrains: Introductory survey of non-heuristic algorithms, Technical report ORNL/TM-12410, Oak Ridge National Laboratory, July 1993.

[16] N. Santoro, Sense of direction, topological awareness, and communication complexity, ACM SIGACT News 16 (1984), 50–56.

[17] D. Sleator and R. E. Tarjan, Amortized efficiency of list update and paging rules, Commun ACM 28 (1985), 202–208.