Tree Exploration with an Oracle

Pierre Fraigniaud[1], David Ilcinkas[2], and Andrzej Pelc[3]

[1] CNRS, Laboratoire de Recherche en Informatique (LRI)
Université Paris-Sud
91405 Orsay, France.
pierre@lri.fr

[2] Laboratoire de Recherche en Informatique (LRI)
Université Paris-Sud
91405 Orsay, France.
ilcinkas@lri.fr

[3] Département d'informatique, Université du Québec en Outaouais
Gatineau, Québec J8X 3X7, Canada.
pelc@uqo.ca

Abstract. We study the amount of knowledge about the network that is required in order to efficiently solve a task concerning this network. The impact of available information on the efficiency of solving network problems, such as communication or exploration, has been investigated before but assumptions concerned availability of *particular* items of information about the network, such as the size, the diameter, or a map of the network. In contrast, our approach is *quantitative*: we investigate the minimum number of bits of information (minimum oracle size) that has to be given to an algorithm in order to perform a task with given efficiency.

We illustrate this quantitative approach to available knowledge by the task of tree exploration. A mobile entity (robot) has to traverse all edges of an unknown tree, using as few edge traversals as possible. The quality of an exploration algorithm $\mathcal{A}$ is measured by its *competitive ratio*, i.e., by comparing its cost (number of edge traversals) to the length of the shortest path containing all edges of the tree. Depth-First-Search has competitive ratio 2 and, in the absence of any information about the tree, no algorithm can beat this value.

We determine the minimum number of bits of information that has to be given to an exploration algorithm in order to achieve competitive ratio strictly smaller than 2. Our main result establishes an exact threshold oracle size that turns out to be roughly $\log \log D$, where D is the diameter of the tree. More precisely, for any constant c, we construct an exploration algorithm with competitive ratio smaller than 2, using an oracle of size at most $\log \log D - c$, and we show that every algorithm using an oracle of size $\log \log D - g(D)$, for any function g unbounded from above, has competitive ratio at least 2.

1 Introduction

For many network problems (such as leader election, minimum spanning tree, rendezvous, wakeup, broadcasting, etc.), the quality of the algorithmic solutions often depends on the amount of knowledge given to nodes of the network, or given to mobile entities moving in the network, about its topology. *Local* knowledge given to every node and/or to every mobile entity is its identity and, for a node, its degree (or the list of neighbor identities). Any other knowledge (e.g., the total number of nodes, network diameter, the total number of mobile entities, partial maps of the network, etc.) is *global* knowledge. Many results illustrate the impact of global knowledge on the ability and efficiency of solving network problems. For instance, it is proved in [4] that, if an upper bound $\hat{n}$ on the number n of nodes of a graph is known, then a robot can explore this graph in time polynomial in $\hat{n}$, using one pebble, while without this knowledge, $\Theta(\log\log n)$ pebbles are necessary and sufficient. Broadcasting in radio networks is another subject where global information significantly influences efficiency. In [22] it is shown that if nodes have complete knowledge of the network then deterministic broadcasting can be done in time $O(D + \log^3 n)$, for n-node radio networks with diameter D. (This result has been recently improved to $O(D + \log^2 n)$ in [24]). On the other hand, in [9] a lower bound of $\Omega(n \log D)$ is proved on deterministic broadcasting time in radio networks in which nodes know only their own identity. (An almost matching upper bound of $O(n \log^2 D)$ is proved in [10]). In fact, the impact of global knowledge is significant in many areas of distributed computing, as witnessed by [19, 25] where hundreds of impossibility results and lower bounds for distributed computing are surveyed, many of them depending on whether or not the nodes are given exact or approximate values of global parameters providing partial knowledge of the topology of the network. Finally, notice that the amount of global knowledge has also a strong impact on computing in anonymous networks (cf., e.g., [23], where the impact of knowing the total number of nodes is studied in depth).

We model global knowledge, given to the nodes or to the mobile entities, by an *oracle*. Given a problem $\mathcal{P}$ with the set of instances $\mathcal{I}$, an oracle is a function $\mathcal{O} : \mathcal{I} \mapsto \{0,1\}^*$ that maps any instance I to a binary string $\mathcal{O}(I)$. Solving problem $\mathcal{P}$ using oracle $\mathcal{O}$ consists in designing an algorithm that, given the binary string $\mathcal{O}(I)$, but unaware of I, returns a $\mathcal{P}$*-scheme* for I, i.e., a sequence of instructions executed by the nodes or the mobiles entities, solving $\mathcal{P}$ for I. In this setting, the amount of global knowledge is measured by the *size* of the oracle on every instance I, i.e., the length of the binary string $\mathcal{O}(I)$. Typical questions of interest are then: "What is the minimum size of an oracle for solving problem $\mathcal{P}$?" or "What is the minimum size of an oracle for solving $\mathcal{P}$ within some amount of time?". The novelty and significance of our modeling of global knowledge is that it enables asking such *quantitative* questions about the required knowledge, regardless of what *kind* of knowledge is supplied. This should be contrasted with the traditional approach that assumes availability of particular items of global information.

Modeling knowledge about the network by an oracle has already proved useful in the context of communication problems. In a recent paper [21], we showed tight bounds on oracle size required for an efficient execution of two fundamental communication tasks: broadcast and wakeup. It turns out that the minimum oracle size required for broadcast with a linear number of messages is strictly larger than that required for wakeup with a linear number of messages. In this paper, we address similar quantitative questions about knowledge required for one of the fundamental problems in mobile computing: the exploration problem. We prove a tight bound of roughly $\log\log D$ on the size of an oracle enabling the design of an exploration algorithm with competitive ratio strictly less than 2, on trees of diameter D.

1.1 The background of tree exploration

A robot has to traverse all edges of an undirected connected graph, using as few edge traversals as possible. Graph exploration is most often performed when the robot lacks some essential information on the explored graph. In such case, the quality of an exploration algorithm $\mathcal{A}$ is measured by comparing its cost (number of edge traversals) to the length of the shortest *covering walk* (i.e., the shortest path containing all edges of the graph). This ratio, maximized over all graphs and all starting nodes, is called the *competitive ratio* $\mathcal{R}(\mathcal{A})$ of algorithm $\mathcal{A}$. The situation here is similar to the context of online algorithms, where competitive ratio first appeared. In both cases, the performance of an algorithm lacking some essential knowledge about the environment is compared to that of an algorithm that has this knowledge: in the case of online algorithms, this knowledge concerns future events, and in the case of exploration, it concerns the topology of the graph and its labeling. (An algorithm provided with a fully labeled copy of the explored graph, showing which port at a visited node leads to which neighbor, can find the shortest covering walk off line.)

Depth-First-Search has competitive ratio 2 and it was shown in [14] that no exploration algorithm can beat this value for arbitrary graphs, even when provided with an unlabeled isomorphic copy of the explored graph with the starting node marked. It turns out that merely the absence of labels of ports and nodes in the map is sufficient to confuse any algorithm on some graphs, making it not better than DFS. On the other hand, in the absence of any global information whatsoever, beating competitive ratio 2 was shown impossible even for the family of trees. This leads to the question if competitive ratio smaller than 2 is possible to achieve for tree exploration, if the algorithm is provided with some partial information concerning the explored environment. In [14] a positive answer to this question was given in the case of very large additional information: the robot was provided with an unlabeled map of the tree. However, this assumption is not very realistic. Indeed, exploration is often used as a tool to construct a map of an unknown network, and usually a priori information about the explored network is much more restricted.

1.2 The problem

We consider the problem of the *amount of information* needed to achieve tree exploration with competitive ratio smaller than 2. (Recall that the reason of restricting attention to trees is the above mentioned negative result for general graphs, showing that already relatively simple graphs force competitive ratio at least 2 even with extensive additional information, namely an entire unlabeled copy of the explored graph.)

The problem is formalized as follows. In the framework of tree exploration, we define an *oracle* to be a function $\mathcal{O}$ from the class of all trees to the class of binary strings. Specifically, for every tree T, an exploration algorithm is provided with the string $\mathcal{O}(T)$ and returns an *exploration scheme* for T. Such a scheme, starting at any node u, traverses all edges of T. The size of the oracle for tree T is the length of the string $\mathcal{O}(T)$. We ask what is the minimum size of an oracle for which there exists an exploration algorithm achieving competitive ratio smaller than 2, for all trees.

1.3 Our results

We use the notion of oracle to measure the minimum amount of information required for the design of an efficient exploration algorithm. Our main result establishes an exact threshold oracle size to achieve competitive ratio smaller than 2 for tree exploration. This threshold turns out to be roughly $\log\log D$, where D is the diameter of the tree. More precisely, for any constant c we construct an exploration algorithm with competitive ratio smaller than 2, using an oracle of size at most $\log\log D - c$, and we show that every algorithm using an oracle of size $\log\log D - g(D)$, for any function g unbounded from above, has competitive ratio at least 2.

It is interesting to note the *structure* of the oracle in our positive result. For any tree T, this is a string s of bits depending only on D, and giving an approximation of it, plus an additional bit b that allows the robot to choose between two types of exploration. This additional bit b (depending on D and on the size of the tree) is very important. Indeed, while the string s depends only on D and has length smaller than $\log\log D$, we show that even the full knowledge of D, but without b, is not sufficient to beat competitive ratio 2. More precisely, we show that every exploration algorithm knowing only the diameter of the tree must have competitive ratio at least 2.

1.4 Related work

Exploration of unknown environments has been extensively studied in the literature, both in the geometric and in the graph setting. In the first scenario the environment is modeled, e.g., as a terrain with obstacles that may be convex [7], polygonal [11] or rectangular [3]. Another way is to represent the unknown environment as a graph, assuming that the robot may only move along its edges. The graph model is further specified in two different ways. In [1, 4, 5, 13, 20] the robot explores strongly connected directed graphs and it can move only in the direction from tail to head of an edge, not vice-versa. In [1, 13] the authors study competitive ratio of algorithms exploring directed graphs. The constructed algorithms have competitive ratio exponential in the deficiency d of the graph [13], or competitive ratio $d^{O(\log d)}m$, where m is the number of edges [1]. Recently, the first exploration algorithm with competitive ratio polynomial in the deficiency of the graph has been given in [20].

In [2, 8, 14, 18, 26, 27] the explored graph is undirected and the robot can traverse edges in both directions. In some papers additional restrictions on the moves of the robot are imposed. It is assumed that the robot has either a restricted tank [2, 8], 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 [18].

Another direction of research concerns exploration of anonymous graphs (directed or undirected). In this case it is impossible to explore arbitrary graphs and stop, if no marking of nodes is allowed. Hence the scenario adopted in [4, 5] is to allow 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 of arbitrary directed graphs. Exploring anonymous trees without the possibility of marking nodes is investigated in [15]. The authors concentrate attention not on the cost of exploration but on the minimum amount of memory sufficient to carry out this task. Exploration of anonymous graphs was also considered in [12, 16, 17].

2 Terminology and preliminaries

For any tree T we denote by $|T|$ the number of nodes of T, and call it the *size* of this tree. For a given tree T and starting node u, we denote by $opt(T, u)$ the length of the shortest covering walk of T starting from u, i.e., the length of the shortest path in T starting from u and containing all edges of T. Clearly, $opt(T, u) = 2(n-1) - ecc(u)$, where n is the size of T and $ecc(u)$ is the eccentricity of the starting node u, i.e., the distance from u to the farthest leaf. Depth-First-Search ending in the leaf farthest from the starting node u uses fewest edge traversals.

We assume that all ports at a node v are numbered 1,...,deg(v). Hence the robot can recognize already visited nodes and traversed edges. However, it cannot tell the difference between yet unexplored edges incident to its current position. The robot executes a given *exploration scheme* that, at every node v, makes one of the following decisions: take a specific already explored edge, or take an unexplored edge. If the scheme decides to take an unexplored edge, the actual choice of the edge belongs to an adversary, as we are interested in worst-case performance.

We want an oracle to provide information on the topology of the explored tree, independently of any labeling, hence we define it as a function $\mathcal{O}$ from the class of all *unlabeled* trees to the class of binary strings. For any string s, a tree T such that $\mathcal{O}(T) = s$ is called *compatible* with s. If a tree exploration algorithm $\mathcal{A}$ takes the string $\mathcal{O}(T)$ as input for any tree T, we say that $\mathcal{A}$ *uses* $\mathcal{O}$.

Consider an exploration algorithm $\mathcal{A}$ using oracle $\mathcal{O}$. For any string s in the range of $\mathcal{O}$, algorithm $\mathcal{A}$ produces an exploration scheme that explores all trees compatible with s. For any such tree T and starting node u, the *cost* $\mathcal{A}(T,u)$ of this scheme, run on tree T from the starting node u, is the worst-case number of edge traversals taken over all of the above mentioned choices of an adversary. The competitive ratio of $\mathcal{A}$ is defined as

$$\mathcal{R}(\mathcal{A}) = \sup_{T,u} \frac{\mathcal{A}(T,u)}{opt(T,u)} \ ,$$

where the supremum is taken over all trees T and all starting nodes u of T.

The fact that an oracle is defined on unlabeled rather than labeled trees is an important distinction. For example, for the class of lines, we will prove that an oracle of (asymptotic) size $\log\log n$ is needed to achieve competitive ratio smaller than 2, where n is the length of the line. However, for a *given* labeling, a single bit (indicating the port at the starting node leading to the closer endpoint of the line) is enough to achieve competitive ratio 1: DFS starting toward the closer endpoint achieves it.

The following remark will be useful for proving lower bounds on the competitive ratio of exploration algorithms. Suppose that the robot, at some point of the exploration, is at node v, then moves along an already explored edge e incident to v, and immediately returns to v. For any set of decisions of an adversary, an algorithm causing such a pair of moves, when run on a tree T from some starting node u, has cost strictly larger than the algorithm that skips these two moves. Hence, we restrict attention to exploration algorithms that never perform such returns. We call them *regular*.

In [14] the authors introduced the following classification of exploration algorithms for the class of lines (they considered exploration algorithms that know the length n of the line). Fix n and let *type* k be the set of algorithms that always do at most k returns before reaching an endpoint, and that do exactly this many returns for some combination of starting node and (adversary) choice of the initial direction. They proved the following result that permits to restrict attention to relatively simple algorithms exploring lines, when looking for minimum competitive ratio.

Lemma 1. [14] *Fix* $n \geq 11$. *For every exploration algorithm* $\mathcal{A}$ *for the line* L_n *of length* n *there exists an algorithm* $\mathcal{A}'$ *for* L_n, *such that* $\mathcal{A}'$ *is of type* 1 *and* $\max_{u\in L_n} \frac{\mathcal{A}'(L_n,u)}{opt(L_n,u)} \leq \max_{u\in L_n} \frac{\mathcal{A}(L_n,u)}{opt(L_n,u)}$.

In our setting, an algorithm does not know the length of the line but only the value of the oracle. Hence we change the notion of type in the following way. Consider an algorithm $\mathcal{A}$ using oracle $\mathcal{O}$. Fix a string s in the range of $\mathcal{O}$ and consider the exploration scheme produced by $\mathcal{A}$ for this string. This scheme is of type k if it always does at most k returns before reaching an endpoint, for any line L_n of length n compatible with s, and any starting node u, and if it does exactly this many returns for some line compatible with s, some starting node and some adversary choice of the initial direction.

In the proof of Lemma 1, the algorithm $\mathcal{A}'$ is obtained from $\mathcal{A}$ independently of n. Hence this lemma implies that in our setting the best competitive ratio for the class of lines is achieved by an exploration algorithm that, for any string s, produces a scheme of type 1. This type consists of simple exploration schemes that go x steps in one direction (unless an endpoint is met), then return and go to an endpoint, then return and go to the other endpoint. For any scheme of type 1, this integer x will be called the *probing distance* of the scheme.

The next lemma describes the performance of schemes of type 1 as a function of the probing distance. The proof of the lemma will appear in the full version of the paper.

Lemma 2. *For any positive integer n and any $\alpha < 1$, let $S_{\alpha,n}$ be the exploration scheme of type 1 for the line L_n of length n, with probing distance $\lfloor \alpha n \rfloor$, and let $t_{\alpha,n}(u)$ be the cost of this scheme for starting node u. Let $F_n(\alpha) = \max_{u \in L_n} \frac{t_{\alpha,n}(u)}{opt(L_n,u)}$. Then, there exists a positive integer N_0, such that for any $n \geq N_0$, the function F_n is strictly decreasing in the interval $(0, \frac{\sqrt{3}-1}{2}]$, and $\sup_{n>0} F_n(\alpha) < 2$, for any α in this interval.*

3 The upper bound

In this and the next section, we prove our main result, establishing the exact threshold on the size of an oracle for which an exploration algorithm can have competitive ratio smaller than 2. This result is presented in two theorems, one of which establishes an upper bound on the size of such an oracle, by constructing an appropriate exploration algorithm, and the other, in section 4, proves a matching lower bound. In this section, we establish the upper bound, by constructing exploration algorithm `SKE`(c) (for SMALL-KNOWLEDGE-EXPLORATION(c)), for an arbitrary positive integer constant c. This algorithm has competitive ratio smaller than 2, and uses an oracle $\mathcal{O}_c$ of size at most $\max(1, \log\log D - c)$, for any tree of diameter D.

We first describe the oracle $\mathcal{O}_c$. Fix $c > 0$. Given a tree T of diameter D, the oracle $\mathcal{O}_c$ outputs a bit called `choice` and, if `choice` $= 1$, an integer k using $\lceil \log \lceil \log D \rceil \rceil - (c+3)$ bits. The bit `choice` is used by the algorithm to make a decision concerning two alternative ways of exploration, and the integer k is used to obtain an approximation D_0 of the diameter.

Let N_0 be an integer (whose existence is guaranteed by Lemma 2) such that, for all $n \geq N_0$, the function F_n is strictly decreasing in the interval $(0, \frac{\sqrt{3}-1}{2}]$, and $\sup_{n>0} F_n(\alpha) < 2$, for any α in this interval. For $\alpha \in (0, \frac{\sqrt{3}-1}{2}]$, let $\beta(\alpha) = \sup_{n>0} F_n(\alpha)$. Let T be any tree and let n and D be, respectively, its number of nodes and its diameter. Take ϵ such that $D = (1-\epsilon)n$. We will use the following abbreviations: $\lambda = \frac{\sqrt{3}-1}{2}$, and $\gamma = 2^{2^{c+3}+1}$. We now define a threshold ϵ^* on the value of ϵ that will serve to define the bit `choice`. Let $\epsilon_1 = \frac{\lambda}{16\gamma}$, $\beta_1 = \beta(\epsilon_1)$, $\epsilon_2 = \frac{2-\beta_1}{624}$, and $\epsilon^* = \min(\epsilon_1, \epsilon_2)$. The oracle sets `choice` to 1 if

$$(\epsilon < \epsilon^*) \wedge (D \geq 2^{2^{c+3}}) \wedge (n \geq N_0) \ ,$$

and sets `choice` to 0 otherwise. If `choice` $= 1$, the oracle computes $k = \lfloor \frac{\lceil \log D \rceil}{2^{c+3}} \rfloor$.

Given `choice` and k, Algorithm `SKE`(c) returns an exploration scheme. If `choice` $= 0$, then this scheme is an arbitrary DFS. To fix attention, we take the DFS that always chooses the smallest yet unused port number at every node. Note that `choice` is set to 0 when the diameter of the tree is significantly smaller than its size, or when the diameter is bounded, or when the tree itself is small.

We now describe the much more subtle scheme $\mathcal{X}_c$ produced by the algorithm when `choice` $= 1$. The scheme $\mathcal{X}_c$ uses Procedure `DPDFS`(v) (for Doubling-Partial-Depth-First-Search(v)) that is called at a node v of the explored tree, outputs the two edges connecting v to the two largest subtrees rooted at neighbors of v, completely explores all other subtrees, and eventually returns to v. In the sequel, we will use the notion of a subtree pending from v as an equivalent to the notion of a subtree rooted at a neighbor of v. Procedure `DPDFS`(v) is described in Figure 1.

Procedure `DPDFS`(v)
    $i \leftarrow 1$;
    $S \leftarrow$ set of edges incident to v, connecting v to subtrees
        not yet completely explored;
    while $|S| \geq 3$ **do**
        $S' \leftarrow S$;
        while $S' \neq \emptyset$ **do**
            let $e \in S'$ and let $T(e)$ be the subtree connected to v by edge e;
            explore $T(e)$ by DFS until $\min(|T(e)|, 2^i - 1)$ nodes are visited;
            return to v;
            $S' \leftarrow S' \setminus \{e\}$;
            if $T(e)$ is completely explored **then** $S \leftarrow S \setminus \{e\}$;
        $i \leftarrow i + 1$;
    if $|S| = 2$ **then return** S;
    if $|S| = 1$ **then** let e' be the edge connecting v to the largest
        explored subtree and **return** $S \cup \{e'\}$;
    if $S = \emptyset$ **then** let e' and e'' be the edges connecting v to the two largest
        explored subtrees and **return** $\{e', e''\}$;

Fig. 1. Procedure `DPDFS`

The proof of the following lemma will appear in the full version of the paper.

Lemma 3. *Let v be any node of degree at least 3. Let $T_1, \ldots, T_p$ be the enumeration of the subtrees pending from v in decreasing order of their sizes. Procedure*

$\texttt{DPDFS}(v)$ *returns two edges corresponding to two largest subtrees (up to size equality), and completely explores all other subtrees pending from* v. *Moreover, the cost of Procedure* $\texttt{DPDFS}(v)$ *is at most* $22\sum_{i\geq 3}^{p}|T_i|$.

The intuitive idea of the exploration scheme $\mathcal{X}_c$ (returned by Algorithm $\texttt{SKE}(c)$ when $\texttt{choice} = 1$) is the following. Let $D_0 = 2^{k\cdot 2^{c+3}-1}$. We will prove that D_0 approximates the diameter D as follows: $D_0 \leq D < \gamma D_0$. The robot uses Procedure $\texttt{DPDFS}(v)$ to identify the two edges connecting the current node v to the largest subtrees pending from it. Then the robot moves along one of the edges and applies the procedure again. These consecutive applications define a path of length approximately equal to the diameter of the tree. On this path the robot applies a scheme of type 1 for lines: go at probing distance $\lfloor \lambda D_0/2 \rfloor$, return and go to the endpoint of the path, return and go to the other endpoint of the path. The approximation D_0 of the diameter is tight enough to guarantee good performance of the scheme on this path. On the other hand, the part of the tree disjoint from this path is negligible (this is implied by the conditions of setting $\texttt{choice}$ to 1). These two facts (shown in detail in the proof of Theorem 1) imply that the competitive ratio of scheme $\mathcal{X}_c$ is smaller than 2.

The description of the exploration scheme $\mathcal{X}_c$ is provided in Figure 2. In the description, moves performed during the calls to Procedure $\texttt{DPDFS}$ are called *internal*, and all other moves are called *external*. During the entire exploration, the robot stores the results of all previous actions, and constructs a map of the portion of the tree that has been explored so far.

The proof of the following lemma will appear in the full version of the paper.

Lemma 4. *Algorithm* $\texttt{SKE}(c)$ *is correct.*

Theorem 1. *Let* c *be an arbitrary positive integer constant. Algorithm* $\texttt{SKE}(c)$ *uses an oracle of size at most* $\max(1, \log\log D - c)$, *for any tree of diameter* D, *and has competitive ratio smaller than 2.*

Proof. The result is true for $n = 1$ or $n = 2$, as any algorithm is optimal in this case. In the following, we assume that $n \geq 3$.

Recall that $k = \lfloor \frac{\lceil \log D \rceil}{2^{c+3}} \rfloor$. The oracle $\mathcal{O}_c$ uses at most $\lceil \log k \rceil + 1$ bits, hence at most $\max(1, \log\log D - c)$ bits. The definition of k implies the inequality $k \leq \frac{\lceil \log D \rceil}{2^{c+3}} < k+1$, hence $k \cdot 2^{c+3} \leq \lceil \log D \rceil < k \cdot 2^{c+3} + 2^{c+3}$, and finally $2^{k\cdot 2^{c+3}-1} \leq D < 2^{k\cdot 2^{c+3}} \cdot 2^{2^{c+3}}$. From the definition of D_0 and γ we get $D_0 \leq D < \gamma D_0$.

First assume that the oracle sets $\texttt{choice}$ to 0. Since the exploration scheme in this case is a DFS, the cost of the scheme is at most $2(n-1) - 1 = 2n-3$. The cost $opt(T, u)$, where u is the starting node, is $2(n-1) - ecc(u)$. We have $ecc(u) \leq D = (1-\epsilon)n$. We obtain

$$opt(T, u) \geq 2(n-1) - (1-\epsilon)n = (1+\epsilon)n - 2 \ .$$

Since $D \leq n-1$, we have $\epsilon \geq 1/n$, or equivalently $\epsilon n \geq 1$. Hence the ratio in this case is at most

$$\frac{2n-3}{(1+\epsilon)n-2} = \frac{2n-3}{(1+\epsilon/2)n - 2 + \epsilon n/2} \leq \frac{2n-3}{(1+\epsilon/2)n - 1.5} \leq \frac{2}{1+\epsilon/2} \ .$$

Exploration scheme $\mathcal{X}_c$
while the exploration is not completed **do**
  let v be the current node;
  {*Internal moves:*}
  if $\deg(v) \geq 3$ **then**
    unless Procedure DPDFS(v) has already been applied in a previous step, apply it to get edges e, e' connecting v to the two largest subtrees pending from v;
  {*External move:*}
  if there is only one edge connecting v to a subtree not completely explored **then**
    leave v by this edge;
  else
    {e, e' *are the two edges connecting* v *to the two subtrees not yet completely explored*}
    if during the last external move (if any), the robot did not come to v by e or e' **then**
      leave v by edge e;
    else
      assume w.l.o.g. that the robot came to v by edge e;
      if the robot is for the first time at distance $\lfloor \lambda D_0/2 \rfloor$ from the starting node **then**
        leave v by edge e;
      else leave v by edge e';
endwhile

Fig. 2. Exploration scheme $\mathcal{X}_c$

If $\epsilon \geq \epsilon^*$, then $\frac{2}{1+\epsilon/2} \leq \frac{2}{1+\epsilon^*/2} < 2$. Assume that $\epsilon < \epsilon^*$. Let $D^* = 2^{2^{c+3}}$. Therefore, choice is set to 0 because either $D < D^*$ or $n < N_0$. If $D < D^*$, then $n < \frac{D^*}{1-\epsilon}$. Let $N_1 = \frac{D^*}{1-\epsilon^*}$. Then we have $n < \frac{D^*}{1-\epsilon} \leq \frac{D^*}{1-\epsilon^*} = N_1$. Hence, both when $D < D^*$ and when $n < N_0$, we have $n < N^* = \max(N_0, N_1)$. Let $\epsilon_3 = 1/N^*$. We have $\epsilon \geq 1/n \geq 1/N^* = \epsilon_3$. We obtain $\frac{2}{1+\epsilon/2} \leq \frac{2}{1+\epsilon_3/2} < 2$. Hence the ratio of the cost of DFS (returned by Algorithm SKE(c) when choice is set to 0) to $opt(T,u)$, is at most $\max(\frac{2}{1+\epsilon^*/2}, \frac{2}{1+\epsilon_3/2}) < 2$.

From now on, we assume that the oracle sets choice to 1, hence Algorithm SKE(c) returns exploration scheme $\mathcal{X}_c$.

In the analysis of the cost of exploration scheme $\mathcal{X}_c$, we use the following terminology. Assume that the robot enters some node of degree at least 3 by edge e and applies Procedure DPDFS(v). If the procedure outputs two edges different from e, then we say that the current node v is a *fork*. Now consider edges traversed during external moves. These edges form a subtree T' of T. For any node v, there exist at most two incident edges such that any external move of the robot leaving v takes one of them. Hence, all nodes are of maximal degree 3 in this subtree. Nodes of degree exactly 3 in T' are forks. Let $v_1, \ldots, v_q$ be the forks of T', if any, in order of their first visit by the robot. Let e_i be the edge connecting v_i to the subtree pending from it and containing the starting node u. In view of the definition of a fork, the robot never makes an external move on edge e_i from node v_i. Let u' be the last fork v_q, if any, or $u' = u$, if T' does not contain any fork. Finally, let P be a path of length D in the tree and let P' be the set of nodes in T' visited by an external move after u'. P' is a path because it does not contain any fork other than possibly u'. Finally, let C be the length of a shortest covering walk in path P' starting at node u'.

In our analysis, the cost of the scheme $\mathcal{X}_c$ is split into the cost of internal moves, and the cost of external moves. The proofs of the following claims will appear in the full version of the paper.

Claim 1. The total number of internal moves is at most $154\epsilon n$.

Claim 2. The total number of external moves is at most $\beta_1 C + 2\epsilon n$.

Claim 1 and Claim 2 imply that the total cost of the scheme $\mathcal{X}_c$ is at most $\beta_1 \cdot C + (154+2)\epsilon n = \beta_1 \cdot C + 156\epsilon n$.

It remains to bound the ratio ρ of this cost to $opt(T,u)$. The shortest covering walk starting at u visits u' before any other node of P'. It has then to visit the path P' starting from u'. Therefore, the length of the shortest covering walk starting at u cannot be less than C (the optimal number of moves on P' starting from u'). This gives $\rho \le \frac{\beta_1 \cdot C + 156\epsilon n}{C}$. We have $\epsilon \le \epsilon_2 = \frac{2-\beta_1}{624}$. Together with $C \ge |P'| \ge n/2$ (for the latter inequality, see the proof of Claim 2), this implies

$$\beta_1 + \frac{156\epsilon n}{C} \le \beta_1 + \frac{156\epsilon n}{n/2} \le \beta_1 + 2 \cdot 156 \frac{2-\beta_1}{624} = \beta_1 + \frac{2-\beta_1}{2} = 1 + \frac{\beta_1}{2} < 2 \ .$$

It follows from the above obtained estimates that the competitive ratio of Algorithm $\mathtt{SKE}(c)$ is at most

$$\max\left(\frac{2}{1+\epsilon^*/2}, \frac{2}{1+\epsilon_3/2}, 1+\frac{\beta_1}{2}\right) < 2 \ ,$$

which completes the proof of the theorem. □

4 The lower bound

This section is devoted to establishing a lower bound on the size of an oracle for which there exists an algorithm with competitive ratio smaller than 2. This lower bound exactly matches the upper bound shown previously, and it holds even for the class of lines. Indeed, we show that for oracles whose size for all lines L_k, of diameter (i.e., length) $k \le n$, is smaller than $\log\log n$, and differs from it by an unbounded number of bits, every algorithm has competitive ratio at least 2.

Theorem 2. *Let $\mathcal{O}$ be an oracle and let $f(n)$ denote the maximum of sizes of $\mathcal{O}(L_k)$, for $k \le n$. Let $g : \mathbb{N} \mapsto \mathbb{R}$ be defined by the formula $f(n) = \log\log n - g(n)$. If g is a function unbounded from above, then every exploration algorithm using oracle $\mathcal{O}$ has competitive ratio at least 2.*

Proof. We will use the following claim.

Claim 3. For every positive integers M and γ, there exist integers $n_1 > n_2 \geq M$, such that $\mathcal{O}(L_{n_1}) = \mathcal{O}(L_{n_2})$ and $n_1/n_2 \geq \gamma$.

Suppose that the claim does not hold. Take M and γ that refute it. Let $\psi : \{n : n > M\} \mapsto \mathbb{R}$ be the sequence defined by the formula $\psi(n) = \frac{\log\gamma \log n}{\log n - \log M}$. The sequence ψ converges to $\log\gamma$, hence it is bounded. Let A be such that $\psi(n) < A$ for all n. Since g is an unbounded function, there exists $n_0 > M$ for which $g(n_0) > \log A$. Let x be the size of the set $\{\mathcal{O}(L_k) : k \leq n_0\}$. We have

$$x \leq 2^{f(n_0)} = 2^{\log\log n_0 - g(n_0)} < 2^{\log\log n_0 - \log A} < 2^{\log\log n_0 - \log\frac{\log\gamma\log n_0}{\log n_0 - \log M}}$$

and therefore $x < \frac{\log n_0 - \log M}{\log\gamma}$. All integers k with $\mathcal{O}(L_k) = \mathcal{O}(L_{M\gamma^i})$ must be smaller than $M\gamma^{i+1}$, for $i \geq 0$. Hence all oracle values for lines $L_{M\gamma^i}$ are distinct, and there are x such values. We have $n_0 < M\gamma^x$ because $\mathcal{O}(L_{n_0}) = \mathcal{O}(L_{M\gamma^i})$, for some $i < x$, and hence $n_0 < M\gamma^{i+1} \leq M\gamma^x$. Consequently, $\log n_0 \leq \log M + x\log\gamma < \log M + \log n_0 - \log M$. This contradiction proves Claim 3.

We will now show that any algorithm using oracle $\mathcal{O}$ must have competitive ratio at least 2. In view of Lemma 1 it is enough to restrict attention to algorithms producing exploration schemes of type 1 for the class of lines. The probing distance of such a scheme for line L_n depends only on $\mathcal{O}(L_n)$. Consider an algorithm $\mathcal{A}$ producing a scheme of type 1 with probing distance $\phi(\mathcal{O}(L_n))$. Fix any constant $3/2 < \beta < 2$. Choose γ such that $\frac{2\gamma}{\gamma+2} > \beta$ and M such that $\frac{2M-1}{M+1} > \beta$. Hence $\gamma > 6$. Let $n_1 > n_2 \geq M$ be integers for which $\mathcal{O}(L_{n_1}) = \mathcal{O}(L_{n_2})$ and $n_1 \geq \gamma n_2$. Their existence is guaranteed by Claim 3. Let $y = \phi(\mathcal{O}(L_{n_1}))$. Hence the scheme makes the first change of direction after y steps, both in L_{n_1} and in L_{n_2}, unless an endpoint is encountered earlier. Consider two cases.

If $y \leq n_2$ then consider the behavior of $\mathcal{A}$ on L_{n_1}, with the starting node u at distance $y+1$ from the endpoint toward which the robot starts. Since $\gamma > 6$, this is the endpoint closer to u. Then

$$\begin{aligned}\frac{\mathcal{A}(L_{n_1},u)}{opt(L_{n_1},u)} = \frac{y+2n_1-1}{y+n_1+1} &\geq \frac{n_2+2n_1-1}{n_2+n_1+1} \\ &\geq \frac{(2\gamma+1)n_2-1}{(\gamma+1)n_2+1} \geq \frac{(2\gamma+1)-1}{(\gamma+1)+1} = \frac{2\gamma}{\gamma+2} > \beta \ .\end{aligned}$$

If $y > n_2$ then consider the behavior of $\mathcal{A}$ on L_{n_2} with the starting node u at distance $n_2 - 1$ from the endpoint toward which the robot starts. Then

$$\frac{\mathcal{A}(L_{n_2},u)}{opt(L_{n_2},u)} = \frac{2n_2-1}{n_2+1} \geq \frac{2M-1}{M+1} > \beta \ .$$

This proves that the competitive ratio of algorithm $\mathcal{A}$ is at least 2. □

5 Exploration knowing the diameter

We have shown in Section 3 that very little information (less than $\log\log D$ bits) is needed to beat competitive ratio 2, and in fact, most of this information (all bits except one) concerns the value of the diameter D itself, and is used to establish a lower bound on it. This extra bit, however, cannot be deduced from D alone, and turns out to be crucial. In this section we prove a surprising result that even an algorithm that knows D *exactly* (i.e., is provided with all $\lceil \log D \rceil$ bits of it), but does not have any additional knowledge, cannot beat competitive ratio 2. Notice that a similar argument proves that the exact knowledge of the number n of nodes, with no extra information, is not enough for this purpose either. The proof of the following theorem will appear in the full version of the paper.

Theorem 3. *Let $\mathcal{A}$ be any tree exploration algorithm that, for every tree T, is given the diameter of T as input. Then $\mathcal{A}$ has competitive ratio at least 2.*

Acknowledgment. Pierre Fraigniaud and David Ilcinkas were partially supported by the projects PairAPair of the ACI Masses de Données, and FRAGILE of the ACI Sécurité Informatique. Additional support from the INRIA project "Grand Large".
Andrzej Pelc was partially supported by NSERC discovery grant and by the Research Chair in Distributed Computing of the Université du Québec en Outaouais. This work was done during the visit of Andrzej Pelc at LRI. Additional supports from the Ministère des Relations Internationales du Québec, from University of Paris-Sud, and from the project PairAPair of the ACI Masses de Données.

References

1. S. Albers and M. R. Henzinger, Exploring unknown environments, SIAM Journal on Computing 29 (2000), 1164-1188.
2. B. Awerbuch, M. Betke, R. Rivest and M. Singh, Piecemeal graph learning by a mobile robot, Proc. 8th Conf. on Comput. Learning Theory (1995), 321-328.
3. E. Bar-Eli, P. Berman, A. Fiat and R. Yan, On-line navigation in a room, Journal of Algorithms 17 (1994), 319-341.
4. M.A. Bender, A. Fernandez, D. Ron, A. Sahai and S. Vadhan, The power of a pebble: Exploring and mapping directed graphs, Information and Computation 176 (2002), 1-21.
5. M.A. Bender and D. Slonim, The power of team exploration: Two robots can learn unlabeled directed graphs, Proc. 35th Ann. Symp. on Foundations of Computer Science (FOCS 1994), 75-85.
6. 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 (SODA 1996), 74-84.
7. A. Blum, P. Raghavan and B. Schieber, Navigating in unfamiliar geometric terrain, SIAM Journal on Computing 26 (1997), 110-137.
8. M. Betke, R. Rivest and M. Singh, Piecemeal learning of an unknown environment, Machine Learning 18 (1995), 231-254.

9. A.E.F. Clementi, A. Monti and R. Silvestri, Selective families, superimposed codes, and broadcasting on unknown radio networks, Proc. 12th Ann. ACM-SIAM Symposium on Discrete Algorithms (SODA 2001), 709-718.
10. A. Czumaj and W. Rytter, Broadcasting algorithms in radio networks with unknown topology, Proc. 44th Ann. Symposium on Foundations of Computer Science (FOCS 2003), 492-501.
11. X. Deng, T. Kameda and C. H. Papadimitriou, How to learn an unknown environment I: the rectilinear case, Journal of the ACM 45 (1998), 215-245.
12. X. Deng and A. Mirzaian, Competitive robot mapping with homogeneous markers, IEEE Transactions on Robotics and Automation 12 (1996), 532-542.
13. X. Deng and C. H. Papadimitriou, Exploring an unknown graph, Journal of Graph Theory 32 (1999), 265-297.
14. A. Dessmark and A. Pelc, Optimal graph exploration without good maps, Theoretical Computer Science 326 (2004), 343-362.
15. K. Diks, P. Fraigniaud, E. Kranakis and A. Pelc, Tree exploration with little memory, Journal of Algorithms 51 (2004), 38-63.
16. G. Dudek, M. Jenkin, E. Milios and D. Wilkes, Robotic exploration as graph construction, IEEE Transactions on Robotics and Automation 7 (1991), 859-865.
17. V. Dujmović and S. Whitesides, On validating planar worlds, Proc. 12th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2001), 791-792.
18. C.A. Duncan, S.G. Kobourov and V.S.A. Kumar, Optimal constrained graph exploration, Proc. 12th Ann. ACM-SIAM Symp. on Discrete Algorithms (SODA 2001), 807-814.
19. F. Fich and E. Ruppert. Hundreds of impossibility results for distributed computing, Distributed Computing, 16 (2003), 121–163.
20. R. Fleischer and G. Trippen, Exploring an unknown graph efficiently, Proc. 13th Ann. European Symposium on Algorithms (ESA 2005), LNCS 3669, 11-22.
21. P. Fraigniaud, D. Ilcinkas and A. Pelc. Oracle size: a new measure of difficulty for communication tasks, Proc. 25th Annual ACM Symposium on Principles of Distributed Computing (PODC 2006), to appear.
22. L. Gasieniec, D. Peleg and Q. Xin, Faster communication in known topology radio networks, Proc. 24th Annual ACM Symposium on Principles of Distributed Computing (PODC 2005), 129-137.
23. T. Kameda and M. Yamashita. Computing on anonymous networks: Part I – characterizing the solvable cases. IEEE Transactions on Parallel and Distributed Systems, 7 (1996), 69-89.
24. D. Kowalski and A. Pelc, Optimal deterministic broadcasting in known topology radio networks, manuscript.
25. N. Lynch. A hundred impossibility proofs for distributed computing. Proc. 8th Ann. ACM Symposium on Principles of Distributed Computing (PODC 1989),1-28.
26. P. Panaite and A. Pelc, Exploring unknown undirected graphs, Journal of Algorithms 33 (1999), 281-295.
27. P. Panaite and A. Pelc, Impact of topographic information on graph exploration efficiency, Networks, 36 (2000), 96-103.
28. C. H. Papadimitriou and M. Yannakakis, Shortest paths without a map, Theoretical Computer Science 84 (1991), 127-150.