Exploration d'arbres avec oracle

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

[1] *CNRS, LRI, Université Paris-Sud, France. E-mail :* pierre@lri.fr
[2] *LRI, Université Paris-Sud, France. E-mail :* ilcinkas@lri.fr
[3] *Dép. d'informatique, Univ. du Québec en Outaouais, Canada. E-mail :* pelc@uqo.ca

Résumé. Nous étudions la quantité de connaissance nécessaire sur le réseau pour résoudre efficacement une tâche sur celui-ci. L'impact de l'information disponible sur l'efficacité de la résolution des problèmes, comme la communication ou l'exploration, a déjà été étudié auparavant mais les hypothèses concernaient la disponibilité d'informations *particulières* sur le réseau, comme sa taille, son diamètre ou une carte de celui-ci. Au contraire, notre approche est *quantitative* : nous étudions le nombre minimum de bits d'information (taille d'oracle minimum) qui doivent être fournis à l'algorithme pour effectuer une tâche avec une certaine efficacité.

Nous illustrons cette approche quantitative de l'information disponible par le problème de l'exploration d'arbres. Une entité mobile (robot) doit traverser toutes les arêtes d'un arbre inconnu, en utilisant aussi peu de traversées d'arêtes que possible. La qualité d'un algorithme d'exploration $\mathcal{A}$ est mesuré par son *rapport compétitif*, i.e., par comparaison de son coût (nombre de traversées d'arêtes) à la longueur du plus court chemin contenant toutes les arêtes de l'arbre. Le parcours en profondeur d'abord a un rapport compétitif de 2 et en l'absence d'information sur l'arbre aucun algorithme ne peut faire mieux.

Nous déterminons le nombre minimal de bits d'information qui doit être donné à un algorithme d'exploration afin d'obtenir un rapport compétitif strictement plus petit que 2. Notre résultat principal établit un seuil exact sur la taille de l'oracle, qui est approximativement $\log\log D$ bits, où D est le diamètre de l'arbre. Plus précisément, pour toute constante c, nous construisons un algorithme d'exploration de rapport compétitif plus petit que 2, utilisant un oracle de taille au plus $\log\log D - c$, et nous montrons que tout algorithme utilisant un oracle de taille $\log\log D - g(D)$, pour toute fonction g non bornée supérieurement, a un rapport compétitif au moins égal à 2.

Keywords: algorithme, exploration, oracle, arbre, robot.

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 [BFR+98] 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 [GPX05] 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 [KP]). On the other hand, in [CMS01] 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. In fact, the impact of global knowledge is significant in many areas of distributed computing, as witnessed by [FR03, Lyn89] where hundreds of impossibility

†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".

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.

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 I, an oracle is a function $O : I \mapsto \{0,1\}^*$ that maps any instance I to a binary string $O(I)$. Solving problem $\mathcal{P}$ using oracle O consists in designing an algorithm that, given the binary string $O(I)$, 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 $O(I)$. 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.

In this paper, we address the above quantitative questions about knowledge required for the exploration problem, and 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 (with unbounded memory) 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 $\mathcal{A}$.

Depth-First-Search has competitive ratio 2 and it was shown in [DP04] 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 [DP04] 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. The problem is formalized as follows. In the framework of tree exploration, we define an *oracle* to be a function 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 $O(T)$ 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 $O(T)$. 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.

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 O from the class of all *unlabeled* trees to the class of binary strings. For any string s, a tree T such that $O(T) = s$ is called *compatible* with s. If a tree exploration algorithm $\mathcal{A}$ takes the string $O(T)$ as input for any tree T, we say that $\mathcal{A}$ *uses* O.

Consider an exploration algorithm $\mathcal{A}$ using oracle O. For any string s in the range of 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 following property will be useful for proving lower bounds on the competitive ratio of exploration algorithms. By using some technical results in [DP04], one can show 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 the following type. 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 this type, this integer x will be called the *probing distance* of the scheme.

It turns out that an algorithm using such an exploration scheme with probing distance $\lfloor \alpha n \rfloor$ for the line L_n of length n, has competitive ratio strictly less than 2 if α satisifies $0 < \alpha \leq \frac{\sqrt{3}-1}{2}$.

3 The upper bound

In this section, we establish the upper bound, by constructing exploration algorithm $\mathtt{SKE}(c)$ (for SMALL-KNOWLEDGE-EXPLORATION(c)), for an arbitrary positive integer constant c. This algorithm has competitive ratio smaller than 2, and uses an oracle O_c of size at most $\max(1, \log\log D - c)$, for any tree of diameter D. We first describe the oracle O_c. Fix $c > 0$. Given a tree T of diameter D, the oracle 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 T be any tree and let n and D be, respectively, its number of nodes and its diameter. Take ε such that $D = (1-\varepsilon)n$. The oracle computes three constants ε^*, D^* and N^*, that depend only on c. Then the oracle sets `choice` to 1 if $(\varepsilon < \varepsilon^*) \wedge (D \geq D^*) \wedge (n \geq N^*)$, 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 $\mathtt{SKE}(c)$ returns an exploration scheme. If `choice` $= 0$, then this scheme is an arbitrary DFS. 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. In these particular cases, the competitive ratio of DFS is less than 2.

We now describe the much more subtle scheme $\mathcal{X}_c$ produced by the algorithm when `choice` $= 1$. The condition on ε in this case intuitively means that the tree is a main path with very small trees attached to it. At a node v, the scheme $\mathcal{X}_c$ uses Procedure $\mathtt{DPDFS}(v)$ (for DOUBLING-PARTIAL-DEPTH-FIRST-SEARCH(v)) to find the edges of the main path and to explore the small trees pending from v. The scheme $\mathcal{X}_c$ uses k to compute an approximation D_0 on the path's length. Finally, the robot applies on the main path the previously seen scheme for lines: go at probing distance $\lfloor \lambda D_0/2 \rfloor$, where $\lambda = \frac{\sqrt{3}-1}{2}$, 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 `choice` to 1). These two facts imply that the competitive ratio of scheme $\mathcal{X}_c$ is smaller than 2.

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

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.

Theorem 2 *Let O be an oracle and let* $f(n)$ *denote the maximum of sizes of* $O(L_k)$*, for* $k \leq n$*. 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 O has competitive ratio at least 2.*

The proof is based on the existence of two arbitrarily large lengths n_1 and n_2, whose ratio is also arbitrarily large, and such that the oracle outputs the same string for the two lines L_{n_1} and L_{n_2}. Depending on the strategy of the exploration algorithm $\mathcal{A}$, there is a setting of the starting node u in one of the two lines such that the ratio $\frac{\mathcal{A}(L_{n_i},u)}{opt(L_{n_i},u)}$ is arbitrarily close to 2. Thus any algorithm using oracle O must have competitive ratio 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.

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.*

References

[BFR+98] M.A. Bender, A. Fernandez, D. Ron, A. Sahai and S. Vadhan, The power of a pebble: Exploring and mapping directed graphs, Proc. 30th Ann. Symp. on Theory of Computing (STOC 1998), 269-278.

[CMS01] 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.

[DP04] A. Dessmark and A. Pelc, Optimal graph exploration without good maps, Theoretical Computer Science 326 (2004), 343-362.

[FR03] F. Fich and E. Ruppert. Hundreds of impossibility results for distributed computing, Distributed Computing, 16 (2003), 121–163.

[GPX05] 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.

[KP] D. Kowalski and A. Pelc, Optimal deterministic broadcasting in known topology radio networks, manuscript.

[Lyn89] N. Lynch. A hundred impossibility proofs for distributed computing. Proc. 8th Ann. ACM Symposium on Principles of Distributed Computing (PODC 1989),1-28.