Exploration par un automate fini de réseaux anonymes étiquetés[†]

R. Cohen[1], P. Fraigniaud[2], D. Ilcinkas[2], A. Korman[1], et D. Peleg[1]

[1] *Dept. of Computer Science, Weizmann Institute, Israel*
[2] *CNRS, LRI, Université Paris-Sud, France.*

Résumé. Cet article traite de l'exploration d'un réseau par un agent mobile, ou *robot*, modélisé par un automate fini. Le robot ne dispose pas de connaissances préalables sur la topologie du réseau, ni même sur sa taille. Sa tâche consiste à explorer chacun des nœuds du réseau. Il a été montré que, pour tout robot $\mathcal{R}$ à K états, et pour tout $d \geq 3$, il existe un réseau de degré maximum d et d'au plus $K+1$ nœuds que $\mathcal{R}$ ne réussit pas à explorer. Ce papier s'intéresse à la possibilité d'aider le robot en ajoutant des étiquettes de petite taille aux nœuds grâce à un calcul préalable. Nous décrivons un algorithme d'exploration qui, étant donné des étiquettes appropriées sur 2 bits (en fait des étiquettes de trois valeurs différentes), permet à un robot d'explorer tous les réseaux. De plus, nous décrivons un algorithme linéaire calculant un tel étiquetage. Nous montrons également comment modifier notre méthode d'étiquetage afin qu'un robot puisse explorer tous les réseaux de degré constant, avec des étiquettes de seulement 1 bit (donc des étiquettes de deux valeurs différentes seulement). Autrement dit, alors qu'il n'existe pas de robot capable d'explorer tous les réseaux (non étiquetés) de degré maximum 3, il existe un robot $\mathcal{R}^*$ et un moyen de colorier les nœuds en noir et blanc tel que $\mathcal{R}^*$ réussit à explorer tous les réseaux coloriés de degré maximum 3.

Une version étendue de cet article sera publiée dans ICALP 2005 [3].

Keywords: exploration, labyrinthe, automate fini, agent mobile, robot, étiquetage.

1 Background and model

Let $\mathcal{R}$ be a finite automaton, simply referred to in this context as a *robot*, moving in an unknown graph $G = (V, E)$. The robot has no a priori information about the topology of G and its size. To allow the robot $\mathcal{R}$, visiting a node u, to distinguish between its edges, the $d = \deg(u)$ edges incident to u are associated to d distinct *port numbers* in $\{0, \dots, d-1\}$, in a one-to-one manner. The port numbering is given as part of the input graph, and the robot has no a priori information about it. For convenience of terminology, we henceforth refer to "the edge incident to port number l at node u" simply as "edge l of u". (Clearly, if this edge connects u to v, then it may also be referred to as "edge l' of v" for the appropriate l'.) The robot has a transition function f, and a finite number of states. If $\mathcal{R}$ enters a node of degree d through port i in state s, then it switches to state s' and exits the node through port i', where

$$(s', i') = f(s, i, d).$$

The objective of the robot is to *explore* the graph, i.e., to visit all its nodes. The task of visiting all nodes of a network is fundamental in searching for data stored at unknown nodes or when looking for defective components.

The first known algorithm designed for graph exploration was introduced by Shannon [9]. Since then, several papers have been dedicated to the feasibility of graph exploration by a finite automaton. Rabin [7] conjectured that no finite automaton with a finite number of pebbles can explore all graphs (a *pebble* is a marker that can be dropped at and removed from nodes). The first step towards a formal proof of Rabin's

[†]Les deuxièmes et troisièmes auteurs ont reçu le support des projets Fragile de de l'ACI Sécurité Informatique, PairAPair de l'ACI Masse de Données, et Grand Large de l'INRIA.

conjecture is generally attributed to Budach [2], for a robot without pebbles. Blum and Kozen [1] improved Budach's result by proving that a robot with three pebbles cannot perform exploration of all graphs. Kozen [6] proved that a robot with four pebbles cannot explore all graphs. Finally, Rollik [8] gave a complete proof of Rabin's conjecture, showing that no robot with a finite number of pebbles can explore all graphs. The result holds even when restricted to planar 3-regular graphs. Without pebbles, it was proved [5] that a robot needs $\Theta(D\log\Delta)$ bits of memory for exploring all graphs of diameter D and maximum degree Δ. On the other hand, if the class of input graphs is restricted to trees, then exploration is possible even by a robot with no memory (i.e., zero states), simply by DFS using the transition function $f(i,d) = i+1 \bmod d$ (see, e.g., [4]).

The ability of dropping and removing pebbles at nodes can be viewed alternatively as the ability of the robot to dynamically *label* the nodes. If the robot is given k pebbles, then, at any time of the exploration, $\sum_{u\in V} |l_u| \leq k$ where l_u is the label of node u and $|l_u|$ denotes the size of the label in unary. This paper considers the effects of allowing the system designer to assign labels to the nodes in a preprocessing stage, and using these labels to guide the exploration by the robot. The transition function f is augmented to utilize labels as follows. If $\mathcal{R}$ in state s enters a node of degree d, labeled by l, through port i, then it switches to state s' and exits the node through port i', where

$$(s',i') = f(s,i,d,l).$$

This model can be considered stronger than Rabin's pebble model since labels are given in a preprocessing stage whereas in Rabin's model the automaton starts with all its pebbles. But it can also be considered weaker since, once assigned to nodes, the labels cannot be modified.

In this paper, we consider settings where it is expected that the graph will be visited by many exploring robots, and consequently, the system designer would like to preprocess the graph by leaving (preferably small) road-signs, or *labels*, that will aid the robots in their exploration task. As possible scenarios one may consider a network system where finite automata are used for traversing the system and distributing information in a sequential manner.

More formally, we address the design of *exploration labeling schemes*. Such schemes consist of a pair $(\mathcal{L},\mathcal{R})$ such that, given any graph G with any port numbering, the algorithm $\mathcal{L}$ labels the nodes of G, and the robot $\mathcal{R}$ explores G with the help of the labeling produced by $\mathcal{L}$. In particular, we are interested in exploration labeling schemes for which : (1) the preprocessing time required to label the nodes is polynomial, (2) the labels are short, and (3) the exploration is completed after a small number of edge-traversals. Note that we are only interested in the preprocessing time. The algorithm $\mathcal{L}$ may require a global knowledge of the graph or a large amount of memory.

2 Our results

As a consequence of Budach's result, any exploration labeling scheme must use at least *two* different labels. Our main result states that just *three* labels (e.g., three colors) are sufficient for enabling a robot to explore all graphs. Moreover, we show that our labeling scheme gives to the robot the power to stop once exploration is completed, although, in the general setting of graph exploration, the robot is not required to stop once the exploration has been completed, i.e., once all nodes have been visited. In fact, we show that exploration is completed in time $O(m)$, i.e., after $O(m)$ edge traversals, in any m-edge graph.

For the class of bounded degree graphs, we design an exploration scheme using even smaller labels. More precisely, we show that just *two* labels (i.e., 1-bit labels) are sufficient for enabling a robot to explore all bounded degree graphs. The robot is however required to have a memory of size $O(\log\Delta)$ to explore all graphs of maximum degree Δ. The completion time $O(\Delta^{O(1)}m)$ of the exploration is larger than the one of our previous 2-bit labeling scheme, nevertheless it remains polynomial.

All these results are summarized in Table 1. The two mentioned labeling schemes require polynomial preprocessing time.

Label size (#bits)	Robot's memory (#bits)	Time (#edge-traversals)
2	$O(1)$	$O(m)$
1	$O(\log\Delta)$	$O(\Delta^{O(1)}m)$

TAB. 1: Summary of main results.

3 A 2-bit exploration-labeling scheme

Our first result is to describe a 3-valued exploration labeled scheme for a robot with constant memory.

Theorem 1 *There exists a robot with the property that for any graph G, it is possible to color the nodes of G with three colors (or alternatively, assign each node a 2-bit label) so that using the labeling, the robot can explore the entire graph G, starting from any given node and terminating after identifying that the entire graph has been traversed. Moreover, the total number of edge-traversals by the robot is* $\leq 20m$.

We first describe the labeling scheme $\mathcal{L}$ and then the exploration algorithm. The node labeling is in fact very simple ; it uses three labels, called colors, and denoted WHITE, BLACK, and RED. Let D be the diameter of the graph.

Pick an arbitrary node r. Node r is called the *root* of the labeling $\mathcal{L}$. Nodes at distance d from r, $0 \leq d \leq D$, are labeled WHITE if $d \bmod 3 = 0$, BLACK if $d \bmod 3 = 1$, and RED if $d \bmod 3 = 2$.

The neighbor set $\mathcal{N}(u)$ of each node u can be partitioned into three disjoint sets : (1) the set $\mathsf{pred}(u)$ of neighbors closer to r than u ; (2) the set $\mathsf{succ}(u)$ of neighbors farther from r than u ; (3) the set $\mathsf{sibling}(u)$ of neighbors at the same distance from r as u. We also identify the following two special subsets of neighbors :

- $\mathsf{parent}(u)$ is the node $v \in \mathsf{pred}(u)$ such that the edge $\{u,v\}$ has the smallest port number at u among all edges leading to a node in $\mathsf{pred}(u)$.
- $\mathsf{child}(u)$ is the set of nodes $v \in \mathsf{succ}(u)$ such that $\mathsf{parent}(v) = u$.

For the root, set $\mathsf{parent}(r) = \emptyset$.

Note that for every node u with label $\mathcal{L}(u)$, and for every neighbor $v \in \mathcal{N}(u)$, the label $\mathcal{L}(v)$ uniquely determines whether v belongs to $\mathsf{pred}(u)$, $\mathsf{succ}(u)$ or $\mathsf{sibling}(u)$.

Furthermore one can design a procedure called `Check_Edge` that allows a robot with constant memory to identify precisely whether an edge incident to a node u leads to $\mathsf{parent}(u)$ or to a node in $\mathsf{child}(u)$.

The functions parent and child induce a spanning tree of the graph. One can then design a quite simple robot that can perform a depth-first search of the tree, using a constant number of memory bits.

4 A 1-bit exploration-labeling scheme for bounded degree graphs

We now describe an exploration labeling scheme using only 1-bit labels. This scheme requires a robot with $O(\log\Delta)$ bits of memory for the exploration of graphs of maximum degree Δ. More precisely, we prove the following.

Theorem 2 *There exists a robot with the property that for any graph G of degree bounded by a constant* Δ*, it is possible to color the nodes of G with two colors (or alternatively, assign each node a 1-bit label) so that using the labeling, the robot can explore the entire graph G, starting from any given node and terminating after identifying that the entire graph has been traversed. The robot has* $O(\log\Delta)$ *bits of memory, and the total number of edge-traversals by the robot is* $O(\Delta^{(O(1)}m)$.

As for $\mathcal{L}$, pick an arbitrary node $r \in V$, called the *root*. Nodes at distance d from r are labeled as a function of $d \bmod 8$. Partition the nodes into eight *classes* by letting

$$C_i = \{u \in V \mid \mathrm{dist}_G(r,u) \bmod 8 = i\}$$

for $0 \leq i \leq 7$. Node u is colored white if $u \in C_0 \cup C_2 \cup C_3 \cup C_4$, and black otherwise. Let

$$\tilde{C}_1 = \{u \mid \mathrm{dist}_G(r,u) = 1\}$$

$$\widehat{C} = \{r\} \cup \{u \in C_2 \mid \mathrm{dist}_G(r,u) = 2 \text{ and } \mathcal{N}(u) = \tilde{C}_1\}.$$

Lemma 1 *There is a local search procedure enabling a robot of $O(\log\Delta)$ bits of memory to decide whether a node u belongs to $\widehat{C}$ and to $\tilde{C}_1$, and to identify the class C_i of every node $u \notin \widehat{C}$.*

Proof. Let **B** (resp., **W**) be the set of black (resp., white) nodes which have all their neighbors black (resp., white). The class C_1 and the classes $C_3,\ldots,C_7$ can be described using the colors of the node and of the other nodes at distance at most 4. For example, we provide the caracterization of the class C_1 : $u \in C_1 \Leftrightarrow u$ is black, u has no neighbor in **B**, and u has a white neighbor v that has no neighbor in **W**.

Based on those characterizations, the classes C_1 and $C_3,\ldots,C_7$ can be easily identified by a robot of $O(\log\Delta)$ bits, via performing a local search. Moreover, the sets $\tilde{C}_1$ and $\widehat{C}$ can also be characterized as follows :

– $u \in \tilde{C}_1 \Leftrightarrow u \in C_1$ and u has no node in C_7 at distance ≤ 2 ;
– $u \in \widehat{C} \Leftrightarrow N(u) \subseteq \tilde{C}_1$ and every node v at distance ≤ 2 from u satisfies $|N(v) \cap \tilde{C}_1| \leq |N(u)|$.

Using this we can deduce :

– $u \in C_0 \setminus \widehat{C} \Leftrightarrow u \notin (\cup_{i=3}^{7} C_i) \cup C_1$ and u has a neighbor in C_7 ;
– $u \in C_2 \setminus \widehat{C} \Leftrightarrow u \notin \widehat{C}$, has a neighbor in C_1, but has no neighbor in C_7.

It follows that a robot of $O(\log\Delta)$ bits can identify the class of every node except for nodes in $\widehat{C}$. □

Proof of Theorem 2. Due to Lemma 1, all instructions of the exploration algorithm using labeling $\mathcal{L}$ can be executed using labeling $\mathcal{L}'$, but for the cases not captured in Lemma 1, i.e., $\widehat{C}$.

To solve the problem of identifying the root, we notice that each of the nodes in $\widehat{C}$ can be used as a root, and all the others can be considered as leaves in C_2. Thus, when leaving the root, the robot should memorize the port P by which it should return to the root. When the robot arrives at a node $u \in \tilde{C}_1$ through a tree edge and has to explore the root, it leaves immediately through port P and deletes the contents of P, then it goes down through the next unexplored port if one is left. When the robot is in a node $u \in \tilde{C}_1$ and has to explore some child, it will skip the port P. □

5 Concluding Remarks

Our results let open a very nice problem : are single bit labels sufficient for ensuring the traversal of all graphs by a finite ($O(1)$ memory) robot ?

Références

[1] M. Blum and D. Kozen. On the power of the compass (or, why mazes are easier to search than graphs). In 19th Symposium on Foundations of Computer Science (FOCS), pages 132-142, 1978.

[2] L. Budach. Automata and labyrinths. Math. Nachrichten, pages 195-282, 1978.

[3] R. Cohen, P. Fraigniaud, D. Ilcinkas, A. Korman and D. Peleg. Label-guided Graph Exploration by a Finite Automaton. In 32nd International Colloquium on Automata, Languages and Programming (ICALP), 2005, to appear.

[4] K. Diks, P. Fraigniaud, E. Kranakis, and A. Pelc. Tree Exploration with Little Memory. J. Algorithms 51(1) :38-63, 2004.

[5] P. Fraigniaud, D. Ilcinkas, G. Peer, A. Pelc, and D. Peleg. Graph Exploration by a Finite Automaton. In 29th International Symposium on Mathematical Foundations of Computer Science (MFCS), LNCS 3153, 451-462, 2004.

[6] D. Kozen. Automata and planar graphs. In Fund. Computat. Theory (FCT), 243-254, 1979.

[7] M.O. Rabin, Maze threading automata. Seminar talk presented at the University of California at Berkeley, October 1967.

[8] H.A. Rollik. Automaten in planaren Graphen. Acta Informatica 13 :287-298, 1980 (also in LNCS 67, 266-275, 1979).

[9] C. Shannon. Presentation of a maze-solving machine. In 8th Conf. of the Josiah Macy Jr. Found. (Cybernetics), pages 173-180, 1951.