Graph Exploration by a Finite Automaton

P. Fraigniaud[1], D. Ilcinkas[1], G. Peer[2], A. Pelc[3], and D. Peleg[2]

[1] CNRS, LRI, Université Paris-Sud, France
[2] Dept. of Computer Science, Weizmann Institute, Israel
[3] Dép. d'informatique, Univ. du Québec en Outaouais, Canada

Abstract. A finite automaton, simply referred to as a *robot*, has to explore a graph whose nodes are unlabeled and whose edge ports are locally labeled at each node. The robot has no a priori knowledge of the topology of the graph or of its size. Its task is to traverse all the edges of the graph. We first show that, for any K-state robot and any $d \geq 3$, there exists a planar graph of maximum degree d with at most $K + 1$ nodes that the robot cannot explore. This bound improves all previous bounds in the literature. More interestingly, we show that, in order to explore all graphs of diameter D and maximum degree d, a robot needs $\Omega(D \log d)$ memory bits, even if we restrict the exploration to planar graphs. This latter bound is tight. Indeed, a simple DFS at depth $D + 1$ enables a robot to explore any graph of diameter D and maximum degree d using a memory of size $O(D \log d)$ bits. We thus prove that the worst case space complexity of graph exploration is $\Theta(D \log d)$ bits.

1 Introduction

1.1 Background and motivation

A mobile entity, e.g., a software agent or a robot, has to *explore* an undirected graph by visiting all its nodes and traversing all its edges, without any a priori knowledge of the topology of the graph or of its size. The task of visiting all nodes is fundamental in searching for data stored at unknown nodes of a network, and traversing all edges is often required in network maintenance and when looking for defective components. More precisely, we consider the task of "perpetual" exploration in which the robot has to traverse all edges of the graph but is not required to stop. That is, the robot moves from node to node, traversing edges, so that eventually all edges have been traversed. Perpetual exploration is of practical interest, e.g., if regular control of a network for the presence of faults is required, and all edges must be periodically traversed over long periods of time.

If nodes and edges have unique labels, exploration can be easily achieved (e.g., by depth-first search). However, in some navigation problems in unknown environments, such unique labeling may not be available, or limited sensory capabilities of the robot may prevent it from perceiving such labels. Hence it is important to be able to program the robot to explore *anonymous* graphs, i.e., graphs without unique labeling of nodes or edges. Clearly, the robot has to be able to locally distinguish ports at a node: otherwise it is impossible to explore even the star with 3 leaves (after visiting the second leaf, the robot cannot distinguish the port leading to the first visited leaf from that leading to the unvisited one). Hence we make a natural assumption that all ports at a node are locally labeled $1, \ldots, d$, where d is the degree of the node. No consistency between those local labelings is assumed.

In many applications, robots and mobile agents are meant to be simple, often small and inexpensive devices. This limits the amount of memory with which they can be equipped. As opposed to numerous papers that imposed no restrictions on the memory of the robot and sought exploration algorithms minimizing time, i.e., the number of edge traversals, we investigate the minimum memory size of the robot that allows exploration of graphs of given (unknown) size, regardless of the time of exploration. That is, we want to find an algorithm for a robot performing exploration, using as little memory as possible.

A robot with a k-bit memory is modeled as a finite automaton. The first known finite automaton algorithm designed for graph exploration was introduced by Shannon [35] in 1951. Since then several papers have been dedicated to the graph exploration problem. In 1967, during his talk at Berkeley, Rabin [32] presented a proof 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). In 1971, Müller [29] gave some formal arguments to support Rabin's claim, in the restricted case of a robot without pebble. In 1977, Coy [16] presented another proof, but some parts of it are fuzzy. The first formal proof of Rabin's claim is generally attributed to Budach [15], in 1978, for a robot without pebble. Actually, the long and technical paper by Budach is concerned with labyrinths. A *labyrinth* is a two-dimensional obstructed chess-board (i.e., $\mathbb{Z}^2$ with forbidden cells). The forbidden cells in $\mathbb{Z}^2$ are described by a set L. If L (resp., $\mathbb{Z}^2 \setminus L$) is finite, then the labyrinth is called finite (resp., co-finite). Exploring a finite labyrinth means that the automaton is able to go arbitrarily far away from its starting position, for any starting position. The edges of the labyrinth are consistently labeled North, South, East, West. (Budach's result applies also to graphs because a co-finite labyrinth is a finite graph.) The same year, Blum and Kozen [10] improved Budach's result by proving that three finite automata cannot cooperatively perform exploration of all graphs. In 1979, Kozen [27] proved that four cooperative robots cannot explore all graphs. Finally, in 1980, Rollik [34] gave a complete proof of Rabin's claim. More precisely, Rollik proved that no finite set of finite automata can cooperatively perform exploration of all cubic planar graphs. Since a finite automaton is more powerful than a pebble, Rabin's claim is a corollary of Rollik's theorem. In all proofs, including the one by Budach and the one by Rollik, the size of the smallest *trap* for an automaton with no pebble (i.e., the smallest graph that an automaton with no pebble cannot explore) is large. One of the objectives of the current paper is to revisit Rabin's claim in the case of a robot with no pebble, specifically for improving the size of traps, and for designing traps with specific topological properties.

1.2 Our results

Our first result is the design of a trap with at most $K+1$ vertices for any K-state automaton. More precisely, we prove that, for any $d \geq 3$ and for any K-state automaton, there exists a planar graph of $K+1$ nodes and maximum degree d that the automaton cannot explore. (We assume $d \geq 3$ since, obviously, all connected graphs of maximum degree $d \leq 2$ can be explored by a robot with a constant memory size.) This construction improves –in terms of size– the best bound known so far, i.e., $2K$, due to Rollik.

More importantly, our construction methodology is quite generic and can be adapted for the minimization of other graph parameters. In particular, we prove that, for any $d \geq 3$ and for any K-state automaton, there exists a planar graph of $O(K)$ nodes, maximum degree d, and diameter $O(\frac{\log K}{\log d})$ that the automaton cannot explore. This latter result has an important corollary, namely that for any $d \geq 3$ and any D, a robot requires $\Omega(D \log d)$ memory bits to explore all graphs of maximum degree d and diameter D. This bound is tight. Indeed, a simple DFS at depth $D+1$ enables a robot with $O(D \log d)$ memory bits to explore all graphs of maximum degree d and diameter D.

To summarize, we prove that the worst case space complexity of graph exploration is $\Theta(D \log d)$ bits.

1.3 Related work

Exploration and navigation problems for robots in an unknown environment have been extensively studied in the literature (cf. [25,33]). There are two groups of models for these problems. In one of them a particular geometric setting is assumed (see, e.g., [5,9,17]). Another approach is to model the environment as a graph, assuming that the robot may only move along its edges. The graph setting can be further specified in two different ways. In [1,8,7,18,24] the robot explores strongly connected directed graphs and it can move only in the head-to-tail direction of an edge, not vice-versa. In [4,12,15,19,21–23,28,31,34] the explored graph is undirected and the robot can traverse edges in both directions. Graph exploration scenarios considered in the literature differ in an important way: it is either assumed that nodes of the graph have unique labels which the robot can recognize (as in, e.g., [18,22,31]), or it is assumed that nodes are anonymous (as in, e.g., [7,8,15,34]). We are concerned with the latter context. The efficiency measure adopted in papers dealing with graph exploration is either the completion time of this task, measured by the number of edge traversals, (cf., e.g., [31]), or the memory size of the robot, measured either in bits or by the number of states of the finite automaton modeling the robot (cf., e.g.[19, 24]). Time is not an issue in our approach, and we address the latter efficiency measure, i.e., memory space. Three versions of the exploration problem have been addressed in the literature: exploration with return (in which the robot has to perform exploration and return to its starting position), exploration with stop (in which the robot has to complete exploration and eventually stop), and perpetual exploration (the type of exploration considered in this paper). For instance, it is shown in [19] that exploration with stop in n-node trees requires a robot with memory size $\Omega(\log\log\log n)$, and that exploration with return in n-node trees can be achieved by a robot with $O(\log^2 n)$ memory bits. Minimizing the memory of the robot for the exploration of anonymous undirected graphs has been addressed in, e.g., [10, 15, 19, 27, 34].

Also, a large part of the literature is concerned with labyrinth exploration. The exploration problem in such labyrinths is known to be strictly simpler than graph exploration [10]. In [20], Döpp proved that a robot is able to explore all finite one-component labyrinths (i.e., where the set L of forbidden cells is finite and connected), and asked whether there exists a universal finite automaton, i.e., one able to explore every finite labyrinth. Budach proved (see the sketch in [13] and the complete version in [15]) that no finite automaton can explore all finite labyrinths. The same result holds for co-finite labyrinths. In [11], Blum and Sakoda have shown that there exists a finite automaton able to explore all finite labyrinths using 4 pebbles (the automaton is universal for all labyrinths if 7 pebbles are allowed). Blum and Kozen [10] proved that a finite automaton with only 2 pebbles can explore all co-finite labyrinths. The problem was finally closed by Hoffmann [26] who showed that a finite automaton with a unique pebble cannot explore all finite labyrinths. Again, the same result holds for co-finite labyrinths. Furthermore, a trap for a finite automaton (using no pebble) can be constructed such that L has only three connected components (cf. [30]). Finally, for any finite set of non-cooperative automata, there exists a finite labyrinth that these automata cannot explore [2].

It is worth mentioning that our work has connections with derandomized random walks. There, the objective is to produce an explicit universal traversal sequence (UTS), i.e., a sequence of port labels, such that the path guided by this sequence visits all edges of any graph. However, even if bounds on the length of these sequences have been derived, they provide little knowledge on the minimum number of states for graph exploration by a robot. For instance, sequences of length $\Omega(n \log n)$ are required to traverse all degree 2 graphs with n nodes [6], although a 2-state robot can explore all degree 2 graphs.

2 Terminology and model

An anonymous undirected graph with locally labeled ports is a graph whose nodes are unlabeled and where the edges incident to a node v have distinct labels $1, \ldots, d_v$, where d_v is the degree of v. Thus every undirected edge $\{u, v\}$ has two labels which are called its *port numbers* at u and at v. Port numbering is *local*, i.e., there is no relation between port numbers at u and at v. Unless specified otherwise, all considered graphs are supposed to be connected.

We are given a mobile entity traveling in an anonymous graph with locally labeled ports. The graph and its size are a priori unknown to the entity. The mobile entity is referred to as a *robot*. More precisely, a K-state robot is a finite Moore automaton $\mathcal{R} = (X, Y, \mathcal{S}, \delta, \lambda, S_0)$ where $X \subseteq \mathbb{N}^2$, $Y \subseteq \mathbb{N}$, $\mathcal{S}$ is a set of K states among which is a specified state S_0 called the *initial* state, $\delta : \mathcal{S} \times X \to \mathcal{S}$, and $\lambda : \mathcal{S} \to Y$. Initially the robot is at some node u_0 in the initial state $S_0 \in \mathcal{S}$. S_0 determines a local port number $p = \lambda(S_0) \in Y$, by which the robot leaves u_0. When incoming to a node v, the behavior of the robot is as follows. It reads the number i of the port through which it entered v and the degree d_v of v. The pair $(i, d_v) \in X$ is an input symbol that causes the transition from state S to state $S' = \delta(S, (i, d_v))$. S' determines a local port number $p = \lambda(S')$, by which the robot leaves v. The robot continues moving in this way, possibly infinitely.

As mentioned before, we consider the task of "perpetual" exploration in which the robot has to traverse all edges of the graph but is not required to stop. That is, it is not required that a final state be in $\mathcal{S}$. A robot is said to perform an *exploration* of a graph G, if starting at *any* node of G in the initial state S_0, it completes traversing all edges of G in finitely many steps.

3 Traps and lower bounds

In order to prove lower bounds for the exploration problem, we first study the maximum size of graphs that a given robot can explore. Let $\mathcal{R}$ be a robot. A *trap* for $\mathcal{R}$ is a pair (G, u_0) where $G = (V, E)$ is a graph, $u_0 \in V$, and starting at vertex u_0 the robot $\mathcal{R}$ fails to explore G, i.e., there exists an edge $e \in E$ such that, for any $t \geq 0$, the robot has not traversed e during the first t steps of the exploration. Given a K-state robot $\mathcal{R}$ (hence with $\lceil \log K \rceil$ memory bits), we construct a trap for this robot. Our objective is to construct small traps, or traps with small diameter.

For the purpose of constructing traps, let us introduce some tools. A graph G of maximum degree d is *edge-colored* if every edge of G is given a color, every two incident edges have different colors, and there are d colors used in total. There is a clear correspondence between regular edge-colored graphs and regular edge-labeled graphs in which the labels at the two extremities of each edge are identical.

Definition 1. *A sequence L of labels is a* pseudo-palindrome *if any of the following two conditions is satisfied:* (1) $L = \emptyset$, *or* (2) $L = L' \circ (\ell, \ell) \circ L''$, *where $L' \circ L''$ is a pseudo-palindrome, ℓ is a label, and $\circ$ denotes concatenation. In particular, a palindrome is a pseudo-palindrome precisely if its length is even.*

A sequence L' is a *reduction* of L if $L' = A \circ B$ and $L = A \circ L'' \circ B$ where L'' is a nonempty pseudo-palindrome, and A and B are two arbitrary sequences (possibly empty). A sequence is said *pp-free* if it has no reduction. A sequence L' is the *pp-reduction* of a sequence L if L' is pp-free and obtained from L by successive reductions. One can easily check that the pp-reduction of a sequence is unique (cf., e.g., Section 1.7 in [15]). For instance the pp-reduction of 11221211213223311131332311221 is 1231. Obviously, given any d-regular edge-colored graph $G = (V, E)$ and any node $u \in V$, each sequence L of edge labels defines a path P from u in G. If L is a pseudo-palindrome then P starts and ends at u. (If G is a tree, L is a pseudo-palindrome if and only if P starts and ends at u.)

Theorem 1. *For every K-state robot and every $d \geq 3$, there exists a planar graph of maximum degree d with at most $K+1$ nodes that the robot cannot explore.*

Proof. Let T_d be the infinite edge-colored regular tree of degree d. Let u_0 be any node. Assume that the robot starts from u_0 in state S_0. After at most K steps (hence after visiting at most $K+1$ nodes), the robot has been twice in the same state. Let S be such a state, and let u and u' be the first occurrences of two nodes where the robot is in state S. The robot $\mathcal{R}$ is at u at step t and at u' at step t'. Since the robot is in an edge-colored regular graph, the sequence of states becomes periodic after t. Let $p = t' - t$ be the period, and let u'' be the node reached by $\mathcal{R}$ at step $t + 2p = t' + p$. At this step, the robot is again in state S. Finally, let L be the ordered sequence of labels of the edges traversed by the robot from step $t+1$ to step t', and let L' be its pp-reduction.

Intuitively, to construct a trap based on T_d, we modify T_d by merging two nodes so that the robot is trapped in a periodic movement in the modified graph. More precisely, we proceed according to the pp-reduction L' of the sequence of labels L visited while going from u to u' in T_d. First we define an intermediate graph G'', whose definition differs according to the structure of L'.

Case 1: L' is not a palindrome (in particular, L' is not empty). Then L' describes the simple path from u to u' in T_d. Since L' is not a palindrome, it can be written as $L' = (l_1, \ldots, l_q, l, \ldots, l', l_q, \ldots, l_1)$ with $l \neq l'$. Moreover, since L' is pp-free, we have also $l' \neq l_q$. Let v (resp., v') be the node reached from u (resp., u') after following the sequence of edge labels $l_1, \ldots, l_q$ (see Figure 1). Note that we may have $v = u$ and $v' = u'$, but not $v = v'$. Let w be the neighbor of v' such that $\{w, v'\}$ is labeled l'. Since L' is not a palindrome, we have $w \neq v$. We construct G'' as follows. We delete edges labeled l' incident to v and to w, and we replace these two edges by an edge between v and w. This edge is labeled l'. Note that G'' has exactly three connected components.

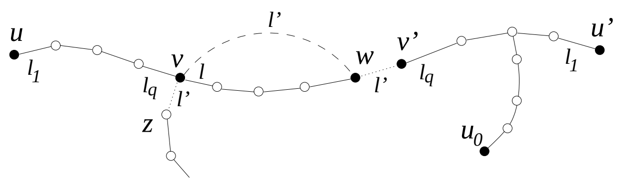

Fig. 1. Construction of G'' in the case where L' is not a palindrome. The dotted edges (v, z) and (w, v') are removed and the dashed edge (v, w) is added instead.

We prove that the behavior of the robot becomes periodic in G''. For that purpose, let us first recompute a starting node of the robot $\mathcal{R}$ such that $\mathcal{R}$ is at u in state S at step t in G''. (Note that the original starting node u_0 may be in a connected component different from the one of u and v. However, u and v are in the same component because $l_q \neq l'$.) To do that, let L'' be the sequence of edge labels corresponding to the walk of the robot from u_0 to u in T_d. Starting from u, let v_0 be the node reached when the robot traverses the edges labeled by L'' in the reverse order. The robot starts in v_0 and, by construction, reaches u in state S at step t. Let us consider the next p steps of the exploration. Since the connected component of G'' containing u is a regular graph of degree d, the sequence of robot's states is the same in G'' as in T_d. Thus, at step t' the robot is in state S in G''. Any pseudo-palindrome defines a closed walk in G''. Recall that the pp-reduction L' of L can be written $L' = (l_1, \ldots, l_q, l, \ldots, l'', l', l_q, \ldots, l_1)$. The sequence $(l_1, \ldots, l_q)$ leads from u to v, and the sequence $(l, \ldots, l'')$ leads from v to w. Indeed, the modification of T_d does not modify the path from u to w because $l' \neq l_q$ and $l' \neq l$. From w, the robot takes the edge labeled l', which is the edge that was added between v and w during the construction of G''. Hence, the robot is back at v in G''. Finally $(l_q, \ldots, l_1)$ leads back from v to u, and $\mathcal{R}$ is in state S at u at step t'. The robot's behavior is thus periodic in G'', as claimed.

Let G' be the graph consisting of all edges traversed by the robot in G'' when starting from v_0. More precisely, G' is the graph composed of all nodes and edges that the robot traverses at least once during its journey from v_0 in G''. Since the robot's behavior is periodic in G'', G' is a finite graph. Actually, G' has at most K nodes. Indeed, after t' steps, the robot is trapped in a cycle. Thus, it does not visit new nodes after step t'. During the first t' steps, the robot visits at most $t' + 1$ nodes. However, it is at the same node u at step t and t'. Hence, the robot visits at most $t' \leq K$ nodes.

To complete the construction of the trap, we add edges to make the degrees of every node in G' exactly d, so that the sequence of robot's states is the same in G' as in T_d. Since G'' is infinite and d-regular, and G' is a finite subgraph of G'', there are necessarily some nodes in G' with degree less than d. Thus, we now complete G' by pairing nodes (possibly including self-loops) until every node of G' (i.e., visited by the robot) is of degree exactly d.

More precisely, let x be a node that needs r additional incident edges. If r is even, we create $r/2$ self-loops around x. If r is odd, we create $(r-1)/2$ self-loops around x. Then every node needs at most one additional edge. G' is a tree, so one can match these nodes, adding one edge for each pair, so that the resulting graph remains planar. After that, there remains at most one unmatched node. We connect this node to an additional (new) node y of degree 1. (As y is never visited by the robot, its degree is immaterial). Therefore, we obtain a planar graph G with at most $K + 1$ nodes (recall that G' has at most K nodes). The added edges are labeled locally. This labeling can be chosen arbitrarily because these edges are not traversed by the robot anyway (the robot only traverses edges of G'). Since exploration means traversing all edges, the robot fails to explore G, and thus (G, v_0) is a trap for $\mathcal{R}$.

Case 2: L' is a palindrome. There are two subcases.

Subcase 2.1: L' is empty (i.e., L is a pseudo-palindrome). Then $u = u'$. The behavior of the robot $\mathcal{R}$ becomes periodic in T_d because $u = u'$ and the robot is in the same state at u in steps t and t'. Hence, G' is defined as in the previous case, i.e., as the graph consisting of all edges traversed by the robot in T_d when starting from u_0. G' is then transformed into G as before. Since, for the same reasons as for the general case, G' has at most K nodes, we get that G has at most $K+1$ nodes. Not all edges of G are visited. Thus (G, u_0) is a trap for $\mathcal{R}$.

Subcase 2.2: L' is a nonempty (odd length) palindrome. Then L' concatenated with itself is a pseudo-palindrome, and thus $u = u''$. As in the previous cases, the behavior of the robot becomes periodic in T_d. However, unlike what happened in the previous cases, the period is $2p$, and the end of the first period occurs at step $t'+p$, instead of step t'. Hence a graph G' defined as in the previous cases may have more than K nodes. To keep G' small, we slightly change the definition of G'' compared to the previous cases. During the $2p$ steps following step t, the robot starts and ends at u in T_d. Since T_d is a tree, the robot visits every edge at least twice, and thus it visits at most p edges and $p+1$ nodes. Thus in total, the robot visits at most $t+p+1 = t'+1$ nodes. Since $t' \leq K$, at most $K+1$ nodes are visited. If the robot actually visits at most K nodes of T_d, then we set $G'' = T_d$. If $\mathcal{R}$ visits exactly $K+1$ nodes of T_d, then we modify T_d as follows. Let $\hat{L}$ be the sequence of edge labels seen by the robot during the first $t+2p$ steps of its journey. Note that $\hat{L}$ contains at least two different labels (i.e., it is not a sequence $(l, l, \ldots, l)$). Indeed, $K \geq d \geq 3$ and thus a sequence $\hat{L} = (l, l, \ldots, l)$ would imply that some edge is visited at least three times. Therefore the robot would have visited at most K nodes, a contradiction with our assumption that $\mathcal{R}$ visits exactly $K+1$ nodes. Let $\hat{L} = (\hat{l}_1, \hat{l}_2, \ldots, \hat{l}_{t+2p})$. Choose the first $i \geq 1$ such that $\hat{l}_i \neq \hat{l}_{i+1}$. For every j, let $\hat{u}_j$ be the node reached by the robot at the end of step j. We merge $\hat{u}_{i-1}$ and $\hat{u}_{i+1}$ by constructing two parallel edges between $\hat{u}_{i-1}$ and $\hat{u}_i$, one of which is labeled $\hat{l}_i$, while the other is labeled $\hat{l}_{i+1}$. The resulting graph is denoted G''. Clearly, the robot visits at most K nodes in G''. We now define G' and G as in the previous cases. G' has at most K nodes, and thus G has at most $K+1$ nodes. In spite of the double edge, the behavior of the robot is periodic in G'' because the sequence LL is a pseudo-palindrome and thus it defines a closed walk in any edge-colored graph. For the same reasons as in the previous cases, not all edges of G are traversed by $\mathcal{R}$, and thus (G, u_0) is a trap for $\mathcal{R}$, which completes the proof of Theorem 1. $\square$

We can rephrase Theorem 1 as follows:

Corollary 1. *A robot that explores all n-node planar graphs requires at least $\lceil \log n \rceil$ memory bits.*

The next result links the number of states of a robot with the maximum diameter of the graphs that it can explore.

Theorem 2. *For every K-state robot and every $d \geq 3$, there exists a planar graph of maximum degree d and diameter at most $4\lceil \log_{d-1} K \rceil + 2$ that the robot cannot explore.*

Proof. We start from the intermediate graph G' defined in the proof of Theorem 1. We complete G' so that all nodes of G' are of degree d as follows. G'

has at most K nodes and for each node there are at most $d-1$ missing edges. Hence, we consider the d-regular tree B of depth $h = \lceil \log_{d-1} K \rceil$. B has at least $d(d-1)^{h-1} > K$ leaves. We add edges from every node of G' to different leaves of B, so that all visited nodes (the nodes of G') are of degree exactly d. The added edges are labeled locally arbitrarily. The resulting graph is denoted by G. Clearly, the pairing between the nodes of G' and the nodes of B can be done so that G is planar. To compute an upper bound on the diameter of G, let us consider an arbitrary node x of G'. During the construction of G', we used T_d, and we constructed G'' in which at most two nodes were modified. Therefore, since $d \geq 3$, at least one edge leads in G'' from x to a node r which is the root of an unmodified infinite subtree T of T_d. At distance at most h from node r, there are at least $(d-1)^h$ nodes in T. Since $(d-1)^h \geq K$, we get that there is a node of T, at distance at most h from r, that is in G'' but not in G'. Therefore, there exists a node in G', at distance at most $h-1$ from r, that has degree smaller than d in G'. This node is connected to the tree B in G. Thus, any node of G' is at distance at most $h+1$ from a node of B. The diameter of B is $2h$. Thus, the diameter of G is at most $(h+1)+2h+(h+1) = 4h+2$, which completes the proof. (As in Theorem 1, the graph G is a trap for $\mathcal{R}$ because the vertices of B are not visited by the robot.) □

As a direct consequence of Theorem 2, we have:

Corollary 2. *A robot that explores all graphs of diameter D and maximum degree d requires at least $\Omega(D \log d)$ memory bits.*

By Corollary 2, the best that a k-bit memory robot can do is to explore all graphs of diameter D and maximum degree d such that $k = \Omega(D \log d)$. In the next section, we show that this goal can be achieved.

4 An exploration algorithm

In this section, we present an algorithm called `Increasing-DFS`, that enables a robot to explore all graphs of sufficiently small diameter and maximum degree. Roughly speaking, exploration is achieved by using a sequence of *depth-first search (DFS)* operations at increasing depths from the initial position u_0 of the robot. The robot keeps in memory the current sequence of port numbers leading back to u_0 in the DFS tree. At Phase i, $i \geq 1$, the robot performs a DFS of depth bounded by i. In the case where one is given a robot $\mathcal{R}$ with k memory bits, we use the variant k-`Increasing-DFS`, that is `Increasing-DFS` in which the robot perpetually checks the size of the currently allocated memory. If this size exceeds k bits, then the robot stops.

Theorem 3. *Algorithm* `Increasing-DFS` *allows a robot to explore every graph. Moreover, Algorithm k-*`Increasing-DFS` *explores all graphs of diameter D and maximum degree d, whenever $k \geq \alpha D \log d$, for some positive constant α.*

Proof. Let the robot $\mathcal{R}$ start from node u_0 in graph G. After $\mathcal{R}$ has performed a DFS of depth i, it has visited all nodes at distance at most i from u_0. Let $i = D + 1$ where D is the diameter of G. Thus, after the ith phase of Algorithm `Increasing-DFS`, all edges have been traversed, and thus exploration has been completed. If $k \geq \alpha D \log d$, then a stack of $D + 1$ elements on $\log d$ bits, and a constant number of scalar variables, can be stored in the robot's memory, for $\alpha = O(1)$ large enough. Thus, when $i = D+1$, the exploration is completed using no more than k bits. Hence any graph of diameter D and maximum degree d can be explored. □

A direct consequence of Theorem 3 is the following:

Corollary 3. *All graphs of diameter D and maximum degree d can be explored by a robot using $O(D \log d)$ memory bits.*

The bound of Corollary 3 is tight (cf. Corollary 2).

As a final remark, observe that algorithm `Increasing-DFS` uses an infinite memory to explore some graphs of bounded size. Nevertheless, this phenomenon cannot be overcome by any exploration algorithm. Indeed, surprisingly, any infinite automaton that explores all graphs is required to use an infinite amount of memory to explore some finite graphs. In particular, for $d \geq 0$, let $\mathcal{G}_d$ be the set of all edge-colored d-regular graphs ($\mathcal{G}_d \neq \emptyset$ as witnessed by, e.g., the hypercube Q_d, or two nodes linked by d parallel edges). We have the following:

Theorem 4. *For any (infinite deterministic) automaton $\mathcal{R}$ that explores all graphs, and for any $G \in \mathcal{G}_d$, $\mathcal{R}$ uses infinitely many memory states when exploring G.*

Proof. Let $\mathcal{R}$ be an automaton that explores all graphs, and let $G \in \mathcal{G}_d$. As a consequence of Theorem 1, $\mathcal{R}$ is an *infinite* automaton $(X, Y, \mathcal{S}, \delta, \lambda, S_0)$, i.e., $|\mathcal{S}|$ is unbounded. Assume, for the purpose of contradiction, that $\mathcal{R}$ uses K states of $\mathcal{S}$ when executed in G, starting from some node, say u_0. Let $\mathcal{R}'$ be the automaton obtained by restricting $\mathcal{R}$ to the diagram induced by these K states of $\mathcal{S}$. More precisely, $\mathcal{R}' = (X, Y, \mathcal{S}', \delta', \lambda', S_0)$ where $\mathcal{S}'$ is the set of the K states visited by $\mathcal{R}$ when exploring G starting from u_0, λ' is λ restricted to $\mathcal{S}'$, and δ' is δ restricted to $\mathcal{S}' \times X$. Let $\mathcal{G}_d(\mathcal{R}')$ be the set of pairs (H, v_0) where $H = (V, E)$ is an edge-labeled graph and $v_0 \in V$, such that, starting at v_0 in H, $\mathcal{R}'$ visits only nodes of degree d and traverses only edges that have identical labels at their two extremities. Let (H, v_0) be the trap for $\mathcal{R}'$ constructed in the proof of Theorem 1. By our construction, we have $(H, v_0) \in \mathcal{G}_d(\mathcal{R}')$. Moreover, since $G \in \mathcal{G}_d$, we also have $(G, u_0) \in \mathcal{G}_d(\mathcal{R}')$. Let $(S_i)_{i \geq 0}$ be the sequence of states of $\mathcal{R}'$ when exploring G starting from u_0. By construction of $\mathcal{R}'$, $(S_i)_{i \geq 0}$ is also the sequence of states of $\mathcal{R}$ when exploring G starting from u_0. In fact, we have $\{S_i, i \geq 0\} = \mathcal{S}'$, and $S_{i+1} = \delta'(S_i, \lambda'(S_i, d)) = \delta(S_i, \lambda(S_i, d))$. Therefore, the sequence $(S_i)_{i \geq 0}$ is independent of any instance (graph, starting node) $\in \mathcal{G}_d(\mathcal{R}')$, and is independent of which automaton $\mathcal{R}$ or $\mathcal{R}'$ is exploring that instance. In particular, the sequence $(S_i)_{i \geq 0}$ is the same for $\mathcal{R}$ and $\mathcal{R}'$ in (H, v_0). Therefore, the sequences of nodes visited by $\mathcal{R}$ and $\mathcal{R}'$ when exploring H starting from v_0 are identical. Since (H, v_0) is a trap for $\mathcal{R}'$, this latter fact is in contradiction with the fact that $\mathcal{R}$ is universal, and thus explores all graphs, including H. Hence $\mathcal{R}$ uses an infinite number of states when exploring G. $\square$

5 Conclusion and future work

As mentioned in the introduction, Rollik [34] proved that no finite set of finite automata can separately (i.e., non-cooperatively) explore all undirected graphs. For his proof, Rollik constructed a trap for q robots of K states each, that is, a graph that none of the q robots explores completely. This trap is of size $O(K^q)$. Thus, an interesting direction of research is to look for smaller traps. In particular, we raise the question of whether there exists a trap of polynomial size for any set of q robots of K states each.

References

1. S. Albers and M. R. Henzinger. Exploring unknown environments. SIAM J. Computing 29:1164-1188, 2000.
2. H. Antelmann, L. Budach and H.A. Rollik. On universal traps. Elektronische Informationsverarbeitung und Kybernetic, EIK 15(3):123-131, 1979.
3. G. Asser. Bemerkungen zum Labyrinth-Problem. Elektronische Informationsverarbeitung und Kybernetic, EIK 13(4-5):203-216, 1977.
4. B. Awerbuch, M. Betke, R. Rivest and M. Singh. Piecemeal graph learning by a mobile robot. In 8th Conf. on Comput. Learning Theory, pages 321-328, 1995.
5. E. Bar-Eli, P. Berman, A. Fiat and R. Yan. On-line navigation in a room. J. Algorithms 17:319-341, 1994.
6. A. Bar-Noy, A. Borodin, M. Karchmer, N. Linial, and M.Werman. Bounds on universal sequences. SIAM J. Computing 18(2):268-277, 1989.
7. M. Bender, A. Fernandez, D. Ron, A. Sahai and S. Vadhan. The power of a pebble: Exploring and mapping directed graphs. In 30th Ann. Symp. on Theory of Computing (STOC), pages 269-278, 1998.
8. M. Bender and D. Slonim. The power of team exploration: Two robots can learn unlabeled directed graphs. In 35th Ann. Symp. on Foundations of Computer Science (FOCS), pages 75-85, 1994.
9. A. Blum, P. Raghavan and B. Schieber. Navigating in unfamiliar geometric terrain. SIAM J. Computing 26:110-137, 1997.
10. 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.
11. M. Blum and W. Sakoda. On the capability of finite automata in 2 and 3 dimensional space. In 18th Ann. Symp. on Foundations of Computer Science (FOCS), pages 147-161, 1977.
12. M. Betke, R. Rivest and M. Singh. Piecemeal learning of an unknown environment. Machine Learning 18:231-254, 1995.
13. L. Budach. On the solution of the labyrinth problem for finite automata. Elektronische Informationsverarbeitung und Kybernetic, EIK 11(10-12):661-672, 1975.

14. L. Budach. Environments, labyrinths and automata. In Fund. Computat. Theory (FCT), LNCS 56, 54-64, 1977.
15. L. Budach. Automata and labyrinths. Math. Nachrichten, pages 195-282, 1978.
16. W. Coy. Automata in labyrinths. In Fund. Computat. Theory (FCT), LNCS 56, 65-71, 1977.
17. X. Deng, T. Kameda and C. H. Papadimitriou How to learn an unknown environment I: the rectilinear case. J. ACM 45:215-245, 1998.
18. X. Deng and C. H. Papadimitriou. Exploring an unknown graph. J. Graph Theory 32:265-297, 1999.
19. K. Diks, P. Fraigniaud, E. Kranakis, and A. Pelc. Tree Exploration with Little Memory. In 13th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA), pages 588-597, 2002.
20. K. Döpp. Automaten in Labyrinthen. Elektronische Informationsverarbeitung und Kybernetic, EIK 7(2):79-94 & 7(3):167-190, 1971.
21. G. Dudek, M. Jenkins, E. Milios, and D. Wilkes. Robotic Exploration as Graph Construction. IEEE Transaction on Robotics and Automation 7(6):859-865, 1991.
22. C. Duncan, S. Kobourov and V. Kumar. Optimal constrained graph exploration. In 12th Ann. ACM-SIAM Symp. on Discrete Algorithms (SODA), pages 807-814, 2001.
23. P. Fraigniaud, L. Gasieniec, D. Kowalski, and A. Pelc. Collective Tree Exploration To appear in 6th Latin American Theoretical Informatics Symposium (LATIN), 2004.
24. P. Fraigniaud, and D. Ilcinkas. Directed Graphs Exploration with Little Memory. In 21st Symposium on Theoretical Aspects of Computer Science (STACS), LNCS 1996, pages 246-257, 2004.
25. A. Hemmerling. Labyrinth Problems: Labyrinth-Searching Abilities of Automata. Volume 114 of Teubner-Texte zur Mathematik. B. G. Teubner Verlagsgesellschaft, Leipzig, 1989.
26. F. Hoffmann. One pebble does not suffice to search plane labyrinths. In Fund. Computat. Theory (FCT), LNCS 117, 433-444, 1981.
27. D. Kozen. Automata and planar graphs. In Fund. Computat. Theory (FCT), 243-254, 1979.
28. A. López-Ortiz and S. Schuierer. On-line parallel heuristics and robot searching under the competitive framework. In 8th Scandinavian Workshop on Algorithm Theory (SWAT), 2002.
29. H. Müller. Endliche Automaten und Labyrinthe. Elektronische Informationsverarbeitung und Kybernetic, EIK 7/4, 261-264, 1971.
30. H. Müller. Automata catching labyrinths with at most three components. Elektronische Informationsverarbeitung und Kybernetic, EIK 15(1-2):3-9, 1979.
31. P. Panaite and A. Pelc, Exploring unknown undirected graphs, J. Algorithms 33:281-295, 1999.
32. M.O. Rabin, Maze threading automata. Seminar talk presented at the University of California at Berkeley, October 1967.
33. N. Rao, S. Kareti, W. Shi, and S. Iyengar. Robot navigation in unknown terrains: Introductory survey of length,non-heuristic algorithms. Tech. Report ORNL/TM-12410, Oak Ridge National Lab., 1993.
34. H.A. Rollik. Automaten in planaren Graphen. Acta Informatica 13:287-298, 1980 (also in LNCS 67, pages 266-275, 1979).
35. CL. E. Shannon. Presentation of a maze-solving machine. In 8th Conf. of the Josiah Macy Jr. Found. (Cybernetics), pages 173-180, 1951.