Space lower bounds for graph exploration via reduced automata*

Pierre Fraigniaud* David Ilcinkas* Sergio Rajsbaum† Sébastien Tixeuil*

* CNRS, LRI, Univ. Paris Sud, France
† Instituto de Matemáticas, UNAM, D. F. 04510, Mexico
{pierre|ilcinkas|tixeuil}@lri.fr rajsbaum@math.unam.mx

Abstract. We consider the task of exploring graphs with anonymous nodes by a team of non-cooperative robots modeled as finite automata. These robots have no *a priori* knowledge of the topology of the graph, or of its size. Each edge has to be traversed by at least one robot. We first show that, for any set of q non-cooperative K-state robots, there exists a graph of size $O(qK)$ that no robot of this set can explore. This improves the $O(K^{O(q)})$ bound by Rollik (1980). Our main result is an application of this improvement. It concerns exploration with stop, in which one robot has to explore and stop after completing exploration. For this task, the robot is provided with a pebble, that it can use to mark nodes. We prove that exploration with stop requires $\Omega(\log n)$ bits for the family of graphs with at most n nodes. On the other hand, we prove that there exists an exploration with stop algorithm using a robot with $O(D \log \Delta)$ bits of memory to explore all graphs of diameter at most D and degree at most Δ.

Keywords: Graph exploration, finite automaton, robot, mobile agent.

1 Introduction

The problem of exploring an unknown environment occurs in a variety of situations, like robot navigation, network maintenance, resource discovery, WWW search, etc. The environment is modeled as a graph where one or more mobile agents, called *robots* in this paper, are trying to collectively traverse every one of its edges. There is a large body of work, that considers several variants of the problem, since at least 1951; see *e.g.* [1–8] and references herein.

In this paper we are interested in exploration of undirected graphs where nodes are not uniquely labeled. Besides the theoretical interest of understanding when or at what cost such graphs can be explored, this situation can occur in practice, due to *e.g.* privacy concerns, limited capabilities of the robots, or simply anonymous edge intersections. We do assume that a robot can identify

* This work has been supported by the projects: INRIA "Grand Large", "PairAPair" of the ACI "Masses de Données", "FRAGILE" of the ACI "Sécurité et Informatique," LAFMI (Franco-Mexican lab in Computer Science), and PAPIIT-UNAM.

the edges incident to a node through unique port labels. Our main goal is to compute complexity bounds on the amount of memory needed by a set of robots as a function of the size of the graphs that they can explore.

A robot moves from one node to another along the edges. When in a node, it (deterministically) decides on the port number of an incident edge to move to the node at the other end of the corresponding edge. It is easy to see that a robot can traverse all edges of some graphs, say a cycle, but that it cannot recognize when it has visited a node twice, so it explores all the graph but never stops. Thus we consider also robots that can mark nodes; as in previous work *e.g.* [1, 2] the robot can drop a pebble in a node and later identify it and pick it up. In this case the robot can explore a graph and stop.

1.1 Collective Exploration

A graph that a set of robots cannot explore when they all star from some given node (or set of nodes) is said to be a *trap* for them. The first trap for a finite state robot is generally attributed to Budach [3] (the trap is actually a planar graph). The trap constructed by Budach is of large size. A much smaller trap was described in [6]: for any K-state robot, there exists a trap of at most $K+1$ nodes. In [7], Rollik proved that no finite set of finite cooperative robots, *i.e.*, automata that exchange information only when they meet at a node, can explore all graphs. In the proof of this result, the author uses as a tool a trap for a set of q non-cooperative K-state robots. This latter trap is of size $O(K^{O(q)})$ nodes. The size[1] $\tilde{O}(K^{K^{\cdot^{\cdot^{K}}}})$, with $2q+1$ levels of exponential, of the trap constructed for cooperative robots depends highly on the size of a trap for non-cooperative robots.

In this paper, we first show (cf. Theorem 1) that for any set of q non-cooperative K-state robots, there exists a 3-regular graph G, and two pairs $\{u, u'\}$ and $\{v, v'\}$ of neighboring nodes, such that any robot of the set, starting from u or u', fails to traverse the edge $\{v, v'\}$. The graph G has $O(qK)$ nodes, thus improving the $O(K^{O(q)})$ bound of [7] (cf. Corollary 1). By simply plugging this new trap for non-cooperative robots in the trap for cooperative robots by Rollik, we get a new trap of size $\tilde{O}(K^{K^{\cdot^{\cdot^{K}}}})$, with $q+1$ levels of exponential, thus smaller than the one in [7] (cf. Corollary 2).

1.2 Exploration by a Single Robot

Theorem 1 has a significant impact on the space complexity of graph exploration by a single robot. We distinguish the two types of exploration mentioned above *perpetual exploration* and *exploration with stop*, where the robot has to stop once exploration is completed.

[1] The $\tilde{O}$ notation hides logarithmic factors.

In acyclic graphs, exploration with stop is strictly more difficult than perpetual exploration. In particular, it is shown in [4] that exploration with stop in n-node bounded degree trees requires a robot with memory size $\Omega(\log\log\log n)$, whereas perpetual exploration requires $O(1)$ bits.

As mentioned above, when exploration with stop is required, the robot is provided with a pebble. We prove (cf. Theorem 2) that exploration with stop requires a robot with $\Omega(\log n)$ bits for the family of graphs with at most n nodes. Note that, in arbitrary graphs, perpetual exploration and exploration with stop are not comparable because even if perpetual exploration is a simpler task than exploration with stop, in the latter case the robot is given a pebble. Therefore, even if the existence of a trap of at most $K + 1$ nodes for any K-state robot described in [6] implies an $\Omega(\log n)$ bits lower bound for the memory size of a robot that performs perpetual exploration in all graphs with at most n nodes, our $\Omega(\log n)$ lower bound is not a consequence of the result in [6].

Finally, we prove (cf. Theorem 3) that there exists an exploration with stop algorithm using a robot with $O(D \log \Delta)$ bits of memory for the exploration with stop of all graphs of diameter at most D and degree at most Δ.

2 Preliminaries

In Section 2.1 we define formally what we mean by a robot exploring a graph. In Section 2.2 we describe the basic properties of a robot. In Section 2.3 we show how to simplify the structure of a robot, for the proofs of the following sections.

2.1 Graphs and Robots

A robot considered in this paper traverses a graph by moving from node to node along the edges of the graph. All nodes are identical and hence indistinguishable to the robot. However, each edge has two labels, each one associated to one of its two endpoints. The labels are arbitrary, except that the edges incident to a node are required to have different labels in their endpoints corresponding to the node. When a robot is in a node, it sees only the labels at the endpoints of the edges incident to the node. This allows the robot to distinguish the edges incident to the node through their unique labels, called *local port numbers.*

An edge may have different port numbers in its two endpoints. When a robot is in a node s and traverses an edge to get to the node t at the other end of the edge, it learns the label at t's endpoint of the edge once it enters t. The robot decides which edge to take to leave t based on this label, as well as on the other local port numbers at t (and hence the degree of t). To compute memory lower bounds, it suffices to consider graphs where both port numbers coincide, and where all nodes have the same degree. In such a graph a robot can be described by a very simple automaton, as we shall see next. Thus, the graphs considered in this paper are δ-*homogeneous* undirected graphs: δ-regular and edge-colored. A graph is δ-*regular* if each of its nodes has degree δ, and it is *edge-colored* if each edge is labeled with one of the integers in the set $\Delta = \{0, 1, \ldots, \delta - 1\}$ in a way that no two edges incident to the same node have the same color.

When a robot traverses a δ-homogeneous graph, each time it arrives to a node the local environment looks exactly the same as in any other node: all nodes are equal and in each node all local ports are $0, 1, \ldots, \delta - 1$. Thus, the robot decides which edge to take to exit the node based only on its current state. Formally, a *δ-robot* or simply *robot* when δ is understood, is an automaton $A = (\Delta, \mathcal{S}, f, s_0)$, with a finite set of states $\mathcal{S}$, an initial state $s_0 \in \mathcal{S}$, and a transition function $f : \mathcal{S} \to \mathcal{S} \times \Delta$. For a state $s \in \mathcal{S}$ with $f(s) = (s', i)$, denote $f_{st}(s) = s'$ and $f_\ell(s) = i$. The δ-robot A moves on a δ-regular graph as follows. Initially A is placed on a node of the graph in state s_0. If A is in a node v in state s then A moves to the node v' such that the edge $\{v, v'\}$ is labeled $f_\ell(s)$, and changes to state $f_{st}(s)$. We say that A *traversed* the edge $\{v, v'\}$. We assume that every state $s \in \mathcal{S}$ of A is reachable from s_0 (unreachable states do not affect the behavior of A and can be ignored). In Section 4 we will consider an extended definition of a robot that can drop a pebble in a node and pick it up when it returns to the node to drop it somewhere else.

A *trap* for a set of δ-robots is a pair (G, U), where G is a δ-homogeneous graph and U is a set of nodes, such that if all the robots are placed in nodes of $u \in U$, each in its initial state, then there will be an edge $\{v, v'\}$ that is never traversed by the robots.

2.2 Basic Properties

Consider a robot $A = (\Delta, \mathcal{S}, f, s_0)$. The transition function f defines a directed labeled graph $G(A) = (\mathcal{S}, F)$ with node set $\mathcal{S}$ and arc set F, such that the arc $s \to t \in F$ iff $f_{st}(s) = t$, and the arc has label $f_\ell(s)$. Notice that the labeled graph $G(A)$ together with the starting node s_0 completely determine the robot A.

Each node of $G(A)$ has out-degree 1 because f is a function. It follows that $G(A)$ consists of a simple, possibly empty path starting in s_0 and ending in some node s_1, followed by a simple cycle starting and ending in s_1. This is because we assume that A has no unreachable states and $\mathcal{S}$ is finite. Thus, the arc labels of the path define a *path word* W_0 over Δ, $|W_0| \geq 0$, and the arc labels of the cycle define a *cycle word* W over Δ, $|W| \geq 1$. Clearly, $|W_0 W| = |\mathcal{S}|$. The *footprint* of A is $fp(A) = W_0 W^*$. When A is placed on a node of a graph in state s_0, $fp(A)$ is the sequence of labels of edges traversed by A. The next lemma says that once A reaches a node x of the graph in some state s that belongs to the cycle of $G(A)$, the path that A follows in G is a closed path that includes x; moreover, A returns to x in the same state s.

Lemma 1. *Consider a robot A with path and cycle words W_0, W placed in a node of a graph G. Let x be a node reached by A after at least $|W_0|$ steps, and assume A is in state s at this moment. Then A will eventually be back in x in state s.*

Proof. Consider the behavior of A after at least $|W_0|$ steps. The robot A is thus changing from state to state along the cycle of $G(A)$. Since this cycle is finite, and G is also finite, the robot must be twice in the same node in the same state. Assume for contradiction that A is not twice in (x, s), *i.e.* in node x in state s. Then let x' be the first node (after x) for which A is twice in the same state, s'. Consider the path taken by A from the initial node to the first time it is in node x' in state s', and let x_1 be the last node before entering x' in state s' for the first time. Suppose A is in state s_1 at this time. So A is never twice in (x_1, s_1). When A eventually returns to x' in s', the last node visited is x'' in state s''. But since all the states considered in the lemma are in the cycle of $G(A)$ (because A has taken at least $|W_0|$ steps), it must be that $s'' = s_1$. Thus, $f_\ell(s'') = f_\ell(s_1)$, which implies that $x_1 = x''$ (since G is homogeneous). Then A is twice in (x_1, s_1), a contradiction. □

2.3 Reduced Robots

A robot A is *irreducible* if $G(A)$ satisfies two properties: (i) for any two consecutive (distinct) arcs $s \to s_1 \to s_2$, it holds $f_\ell(s) \neq f_\ell(s_1)$, and (ii) for the two arcs with the same end-node $s \to s_1$, $s_2 \to s_1$, it holds $f_\ell(s) \neq f_\ell(s_2)$. We show here how to obtain an irreducible robot $A' = (\Delta, \mathcal{S}', f', s'_0)$ from a robot A. The behavior of A and of A' on a graph will not be exactly the same, but will be related in the sense that the region of a graph traversed by A cannot be much larger than the region traversed by A'.

Let $\bar{G}(A)$ be the undirected graph corresponding to $G(A)$. Then, if A is irreducible and its simple cycle is of length at least 2, then $\bar{G}(A)$ is edge-colored. Roughly speaking, we want the robot to be irreducible to construct a graph based on $\bar{G}(A)$ on which the robot will be moving. Since the constructed graph must be homogeneous, $\bar{G}(A)$ must be homogeneous. Then we can place A at the beginning of the path of $\bar{G}(A)$ and it will never try to go out of $\bar{G}(A)$. To obtain an irreducible robot A' from A we perform a series of reduction steps that modify its transition function and reachable states. When A, A' are placed on the same node of a graph, the path traversed by A' is contained in the the path traversed by A; essentially A' skips some closed walks of A. These reductions are formally defined next.

A *reduction step* is the operation consisting of transforming a robot $A = (\Delta, \mathcal{S}, f, s_0)$ into another robot $A' = (\Delta, \mathcal{S}', f', s'_0)$ where one of the above properties (i) or (ii) is enforced for two arcs, each corresponding to a type-i or type-ii reduction step. The idea is to repeat type-i steps until no more are possible, and hence the robot satisfies property (i), and then if property (ii) is not satisfied, do a single type-ii step to enforce property (ii). Only type-i reductions change the path traversed by the robot.

A *type-i* reduction step is applicable if $G(A)$ has two consecutive distinct arcs $s \to s_1 \to s_2$ with $f_\ell(s) = f_\ell(s_1)$. First, if $s = s_2$ (so the cycle is of length 2 with same labels), A' is obtained from A by letting $f'(s) = (s, i)$, where $i = f_\ell(s)$; and if $s_1 = s_0$ then $s'_0 = s$. For other states $f' = f$. Otherwise, if $s \neq s_2$, it is possible that s has $0, 1$, or 2 in-neighbors. In each of these cases A' is obtained from A by the following modifications. If s has 0 in-neighbors, then $s = s_0$; let $s'_0 = s_2$. If s has 1 in-neighbor t ($t \neq s_1$), with $f(t) = (s, i)$, then let $f'(t) = (s_2, i)$; If $s = s_0$ then let $s'_0 = s_2$. If s has 2 in-neighbors t_1, t_2, with $f(t_1) = (s, i)$, $f(t_2) = (s, j)$, then $s \neq s_0$; Let $f'(t_1) = (s_2, i)$ and $f'(t_2) = (s_2, j)$. For other states $f' = f$. After doing these modifications, A' is obtained by removing any unreachable states. Notice that for each one of the previous 3 cases at least one unreachable state is removed, namely s. Thus, at most $K - 1$ type-i reductions are possible, starting from a K-state robot.

We will use the following properties of a type-i reduction. Since $f_\ell(s) = f_\ell(s_1) = i$, if the robot is in a node v of the graph in state s, then it moves to v', where $\{v, v'\}$ is colored i, changes to state s_1, and moves back to v, in state s_2. Thus, it is easy to check that a type-i reduction eliminates this v, v', v loop from the path traversed by the robot in the graph, and makes no other changes to the path; that is, if the path arrives to v from w and then proceeds to w' after traversing the v, v', v loop, after the type-i reduction the robot will go from w to v and then directly to w'. Therefore, before the reduction step, the robot explores a node at most distance 1 from the nodes explored by the robot after the reduction.

Once a type-i reduction step is not applicable in $G(A)$, a single *type-ii* reduction step is used, defined as follows. Assume there are two states such that $f(s) = f(s_1)$, that is, $G(A)$ has two arcs with the same end-node $s \to t$, $s_1 \to t$, and $f_\ell(s) = f_\ell(s_1)$; otherwise the reduction does nothing. Exactly one of s, s_1 must be in the cycle of $G(A)$, let's say s_1. So there is a path from t to s_1. Notice that this path is of length at least 1 (i.e. the cycle of $G(A)$ is of length at least 2), because otherwise $t = s_1$ and there is a loop from t to itself labeled $f_\ell(s)$, and a type-i reduction is applicable.

Recall that $fp(A) = W_0 W^*$. Let W' be the longest common postfix of W_0 and W; $|W'| > 0$ by the type-ii assumption. We consider two cases: $|W_0| > |W'|$ and $|W_0| = |W'|$. In the first case W is a postfix of W_0; let t_1 be the in-neighbor of the node just before W' starts in the simple path of $G(A)$ and let t_2 be the node just before W' starts in the cycle of $G(A)$; thus $f_\ell(f_{st}(t_1)) = f_\ell(t_2)$ is the first letter of W'. In both cases A' is obtained from A by the following modifications: If $|W_0| > |W'|$, let $f'(t_1) = (t_2, f_\ell(t_1))$. If $|W_0| = |W'|$ let $s'_0 = t_2$, and removing any unreachable states.

We use the following two properties of a type-ii reduction. A type-ii reduction does not change at all the path traversed by the robot in the graph. After a type-ii reduction is executed property (ii) is satisfied, and property (i) is not violated.

The previous arguments imply:

Lemma 2. *For any robot $A = (\Delta, \mathcal{S}, f, s_0)$ the robot A' obtained through the longest possible sequence of type-i reductions followed by a type-ii reduction is irreducible. Let k be the number of type-i reduction steps in this sequence. Then $k \leq |\mathcal{S}| - 1$. Assume both start at some node of a given graph. Then any edge traversed by A is at distance at most k from some edge traversed by A'.*

3 A Trap for a Team of Non-Cooperative Robots

In this section, we focus on graph exploration by a team of non-cooperative robots.

Theorem 1. *For any set $\mathcal{A}$ of q non-cooperative K-state robots, there exist a 3-homogeneous graph G and two pairs of neighboring nodes $\{u, u'\}$ and $\{v, v'\}$ such that (1) the edge $\{u, u'\}$ is labeled 0, (2) starting at u or at u', any robot in $\mathcal{A}$ fails to traverse the edge $\{v, v'\}$, and (3) G has $10qK + O(q)$ nodes.*

Proof. The proof is by induction on $q \geq 0$. The basic step is $q = 0$. The corresponding graph G is displayed on Figure 1.

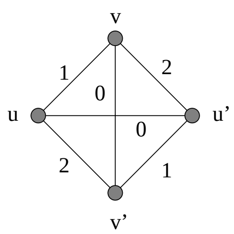

Fig. 1. Basic step of the induction

For the induction step, assume that Theorem 1 holds for q, and let us show that it holds for $q + 1$. Let $\mathcal{A}$ be a set of $q + 1$ non-cooperative K-state robots, and let $A \in \mathcal{A}$. By induction hypothesis, let G_q be an n-node 3-homogeneous graph (where n is $10qK + O(q)$) having two pairs of neighboring nodes $\{u, u'\}$ and $\{v, v'\}$ with the edge $\{u, u'\}$ labeled 0, such that, starting at u or at u', any robot in $\mathcal{A} \setminus \{A\}$ fails to traverse the edge $\{v, v'\}$. We construct a graph G_{q+1} that satisfies Theorem 1 for $\mathcal{A}$.

Let $\widehat{A}$ be an irreducible robot obtained from A as in Lemma 2. Consider its footprint $fp(\widehat{A}) = W_0 W^*$, $|W_0 W| \leq K$. We concentrate first our attention on $\widehat{A}$, and will come back later to the original robot A. Let us denote by p_i the i-th letter in $fp(\widehat{A})$. Recall that since $\widehat{A}$ is irreducible, its associated undirected graph $\bar{G}(\widehat{A})$ is homogeneous.

Let us place $\widehat{A}$ at node u of G_q, and let us observe its behavior. Let H_1 be the graph obtained from G_q by "cutting" the edge $\{v, v'\}$. In H_1, nodes v and v' are both connected to a pending "half-edge." If $\widehat{A}$ traverses the edge $\{v, v'\}$ in G_q, then in H_1 it traverses one of these two half-edges, say the half-edge e pending at v. We consider two cases, depending on when $\widehat{A}$ traverses e.

Case 1. If $\widehat{A}$ traverses e at step $i \leq |W_0|$ (so p_i is the label of e), then we connect to e a path of length $|W_0| - i$ whose extremity is denoted by w. The edges of this path are labeled $p_{i+1}, \ldots, p_{|W_0|}$. Note that since $\widehat{A}$ is irreducible, two consecutive labels of this path are distinct. At w, we add a ring of length $|W|$. The edges of this ring are labeled $p_{|W_0|+1}, \ldots, p_{|W_0|+|W|}$ starting and ending at w. Note that since $\widehat{A}$ is irreducible, two consecutive labels of the ring are distinct, and the three labels $p_{|W_0|}, p_{|W_0|+1}$, and $p_{|W_0|+|W|}$ are pairwise distinct. To avoid parallel edges we add a ring of length $2|W|$ at w if $|W| = 2$; to avoid loops when $|W| = 1$, we add a ring with labels $abab$, where a is equal to the single letter of W, and b is different from a and from the label p_i of e.

Case 2. If $\widehat{A}$ traverses e at step $i > |W_0|$, then it traverses e to get into some state s of the cycle in $G(\widehat{A})$; assume this is the j-th state of the cycle (recall that the cycle is assumed to start in the last state of the path of $G(\widehat{A})$). That is, after traversing e, $\widehat{A}$ would traverse edges labeled $p_{|W_0|+j}, p_{|W_0|+j+1}, \ldots$

Let x be the node of H_1 reached by $\widehat{A}$ after $|W_0|$ steps, let W^{-1} be the sequence W written in reverse order, and let $\widehat{A}^{-1}$ be the robot that traverses edges labeled $(W^{-1})^*$. Thus, when $\widehat{A}^{-1}$ starts at x and $\widehat{A}$ reaches x, $\widehat{A}^{-1}$ proceeds as $\widehat{A}$, but backwards. Let $\widehat{A}^*$ be the robot that traverses edges labeled W^*, *i.e.* the robot derived from $\widehat{A}$ by removing states and transitions that involved W_0.

Claim. Starting from x, $\widehat{A}^{-1}$ eventually traverses one of the half-edges pending at v or v'.

Proof. Assume for contradiction that $\widehat{A}^{-1}$ does not traverse any of the half-edges pending at v or v'. By Lemma 1, $\widehat{A}^{-1}$ returns to x in the same state, and hence its path in H_1 is a closed path. This path traversed backwards is exactly what $\widehat{A}^*$ traverses from x. So $\widehat{A}^*$ does not traverse any of the half-edges pending at v or v'. Thus, $\widehat{A}$ also does not traverse them, a contradiction. $\diamond$

By Claim 3 we can consider the state reached by $\widehat{A}^{-1}$ after it traverses one of the pending half-edges; assume this is the k-th state of the cycle in $G(\widehat{A})$. We consider two sub-cases, depending on whether $\widehat{A}^{-1}$ traverses the same half-edge as $\widehat{A}$, or not.

Case 2.1. The robot $\widehat{A}^{-1}$ traverses the half-edge e pending at v (*i.e.*, the same as $\widehat{A}$). This implies that the k-th label in W is equal to the $(j-1)$-th label in W, which is the label of e. We consider the section of the cycle of $G(\widehat{A})$ from the j-th state to the k-th state. The end edges of this section have the label of e. We now consider the following word: $W' = W(j-1)W(j)W(j+1)\ldots W(k-1)W(k)W(k+1)\ldots W(j-1)W(j)W(j+1)\ldots W(k-1)W(k)W(k+1)\ldots W(j-1)W(j)W(j+1)\ldots W(k-1)W(k)$ (Note that $W(j-1) = W(k)$ and $|W'| \geq 2 \times |W| + 2$). The two robots $\widehat{A}$ and $\widehat{A}^{-1}$ cannot follow the same path forever after crossing edge e: otherwise, it would mean that moving them both backwards, they would also follow the same path forever (which is impossible since the two robots took different paths at node x in the past). Moreover, the two robots must separate after at most $|W|$ steps, and since $|W'| \geq 2 \times |W| + 2$, they must separate after at least 1 step and at most $|W| - 1$ steps. Now, if the two robots separate from each other at some point after crossing edge e, let us consider the smallest l such that $W(j+l) \neq W(k-1-l)$, *i.e.* the nearest place where the two robots separate from one another. Since $W(j-1)=W(k)$, $l \geq 1$. By definition of l, we have $W(j+l-1)=W(k-l)$. Since the considered robots are reduced, we also have $W(j+l-1) \neq W(j+l)$. Still by definition of l, we get $W(j+l) \neq W(k-1-l)$. Finally, because we consider reduced robots and we have $W(j+l-1)=W(k-l)$, we get $W(j+l-1) \neq W(k-1-l)$. Overall, this means that $W(j+l-1)$, $W(j+l)$, and $W(k-1-l)$ are pairwise disjoint. We are now ready to construct the following graph: from e, there is a chain that ends in $W(j+l-1)$ at node w, and from this last node a circle W'' goes from $W(j+l)$ to $W(k-l-1)$. Since $W(j+l) \neq W(k-1-l)$ (see above), $|W''| > 2$. When $|W''| > 2$, we add at w a ring of length $|W''|$ labeled W'', starting and ending at w, so that once $\widehat{A}$ and $\widehat{A}^{-1}$ reach w, each one traverses this ring in the opposite direction, and gets back to w in the appropriate state to proceed along the path back to the half-edge e.

Case 2.2. The robot $\widehat{A}^{-1}$ traverses the half-edge e' pending at v' (*i.e.*, not the same as $\widehat{A}$). Suppose when $\widehat{A}^{-1}$ goes through v' it is in state s. We consider again the section of the cycle of $G(\widehat{A})$ from the j-th state to the k-th state (if the section is of length 1, we extend it with W to make sure there is at least one internal node). We connect e and e' by a path with the labels of this section. Thus, when $\mathcal{A}$ traverses the half-edge e, it follows the newly added path, and gets to v' in the appropriate state, namely s, to proceed along the same path of $\widehat{A}^{-1}$ but backwards, and return to x.

In all three cases, every node of degree 2 in the resulting graph is complemented by a pending half-edge, and every node of degree 1 is complemented by two pending half-edges. Every half-edge is labeled consistently so that the resulting graph, denoted by H_2, is 3-homogeneous.

Finally, if $\widehat{A}$ does not traverse any of the two pending half-edges, then we set $H_2 = H_1$. For $l = 0, 1, 2$, let $parity(l)$ be the parity of the number of pending half-edges labeled l in H_2.

Claim. For any $l, l' \in \{0,1,2\}$, $parity(l) = parity(l')$.

Proof. An edge of H_2 can be considered as two non-pending half-edges. For $l \in \{0,1,2\}$, let t_l be the total number of half-edges of H_2 labeled l, and p_l, resp. np_l, be the number of pending, resp. non-pending, half-edges of H_2 labeled l. All nodes in H_2 are exactly of degree 3 and are incident to one half-edge of each label. Thus $t_0 = t_1 = t_2 = |H_2|$ where $|H_2|$ is the number of nodes of H_2. In H_2, if an half-edge is not pending, then it forms an edge with another non-pending edge of H_2 with the same label. Therefore, all the np_l's are even. Since $t_l = p_l + np_l$, t_l and p_l have the same parity, and thus all the p_l's have the same parity. $\diamond$

The parity of the number of pending half-edges of a given label in H_2 is denoted by ϱ. If ϱ is odd, then we add to H_2 a node connected to one of the half-edges, labeled say l, and add two half-edges pending from this node, labeled $l' \neq l$ and $l'' \notin \{l, l'\}$. As a consequence, ϱ becomes even. Now, we pair the half-edges with identical labels. For every pair but one, we connect the two half-edges of the pair by the gadget displayed in Figure 2. (It could be possible to connect the half-edges by just one edge, but the resulting graph may then not be simple). By labeling the edges of every gadget appropriately, we obtain a 3-homogeneous graph H_3 with only two pending half-edges, of the same label.

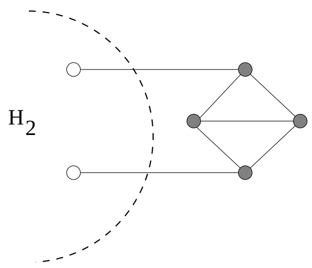

Fig. 2. The gadget for connecting half-edges

Claim. Starting from u, $\widehat{A}$ does not traverse any of the two half-edges of H_3.

Proof. By construction, $\widehat{A}$ travels in G_q, and outside G_q it follows the trajectory defined by paths and/or the ring attached to G_q at edges e and e'. Therefore, $\widehat{A}$ does not traverse any of the half-edges of H_2. In particular, it does not traverse any of the two half-edges of H_3. $\diamond$

We define H_4 as the graph obtained from H_3 by adding a "tower" of height $K+1$ connected to the two remaining half-edges, and a gadget closing the tower (see Figure 3):

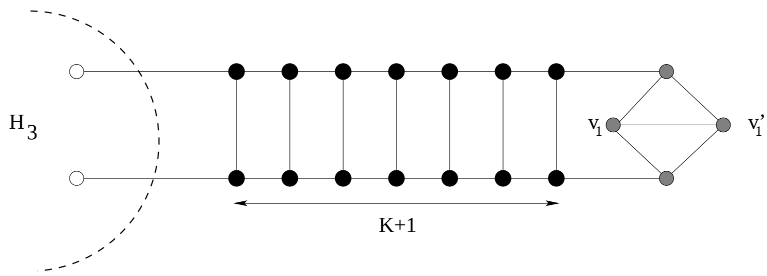

Fig. 3. The "tower" added to H_3

Finally, the two internal nodes of the gadget at the top of the tower in H_4 are denoted by v_1 and v_1' (see Figure 3).

Claim. The edge $\{v_1, v_1'\}$ of H_4 is not traversed by A when starting from u.

Proof. By Claim 3, starting from u in H_4, $\widehat{A}$ does not traverse any of the two edges leading from H_3 to the tower. By Lemma 2, the trajectory of A is never at distance greater than K (where K is the number of states of A) from the trajectory of $\widehat{A}$. Thus, since the tower is of height $K+1$, A never reaches the top of the tower. Therefore, A does not traverse the edge $\{v_1, v_1'\}$. $\diamond$

We repeat the same construction by considering the robot $\widehat{A}$ launched from u' in H_4. More precisely, we construct G_{q+1} from H_4 in the same way H_4 was constructed from G_q. In particular, there is a tower in G_{q+1}, and we define the nodes v_2 and v_2' of G_{q+1} as the two internal nodes of the gadget at the top of this tower. By construction G_{q+1} is 3-homogeneous.

Claim. Any robot in $\mathcal{A}$ fails to traverse the edge $\{v_2, v_2'\}$ of G_{q+1} when starting from u or u'.

Proof. By induction hypothesis, starting from u or u', a robot in $\mathcal{A} \setminus \{A\}$ never traverses v, v' in G_q and so will never traverse any of the edges added to obtain G_{q+1}, and hence does not traverse the edge $\{v_2, v_2'\}$ of G_{q+1}. From Claim 3, starting from u, A fails to traverse the edge $\{v_1, v_1'\}$ of H_4. This edge being the one that is "opened" to construct G_{q+1} from H_4, A fails to reach any of the two nodes v_2 or v_2' in G_{q+1}. Finally, by construction of G_{q+1} from H_4, A fails to reach any of the two nodes v_2 or v_2' in G_{q+1} when starting from u', in the same way A fails to reach any of the two nodes w or w' in H_4 when starting from u. $\diamond$

To complete the proof, it just remains to compute the size of G_{q+1}.

Claim. $|G_{q+1}| \leq |G_q| + 10K + O(1)$.

Proof. We give simple upper bounds on the size of the intermediate graphs. First, we have $|H_2| \leq |G_q| + K + O(1)$. Moreover, there are at most one half-edge pending from every added node in H_2. For each pair of pending half-edges, we added four nodes, and thus $|H_3| \leq |H_2| + 2K + O(1)$. Finally, the tower has $2K + O(1)$ nodes and thus $|H_4| \leq |G_q| + 5K + O(1)$. The same procedure for the starting node u' contributes to another $5K + O(1)$ additional nodes. The result follows. $\diamond$

As a direct consequence of the previous claim, $|G_{q+1}| \leq 10qK + O(q)$, which completes the proof of Theorem 1. $\square$

By simply rewriting Theorem 1, we derive a bound of the size of the smallest trap for a set of q non-cooperative K-state robots, improving the one by Rollik [7]:

Corollary 1. *For any set of q non-cooperative K-state robots, there exists a trap of size $O(qK)$.*

By simply plugging this latter bound in the construction by Rollik [7] for team of cooperative robots, we get:

Corollary 2. *For any set of q cooperative K-state robots, there exists a trap of size $\tilde{O}(K^{K^{\cdot^{\cdot^{\cdot^{K}}}}})$, with $q+1$ levels of exponential.*

4 Bounds for Exploration With Stop

In this section, we consider the *exploration with stop* problem, in which a robot must traverse all edges of the graph, and eventually stop once this task has been achieved. A robot cannot solve this task in graphs with more nodes than its number of states, by Lemma 1. Thus, the robot is given pebbles that it can drop and take to/from any node in the graph. It is known that any finite robot with a finite source of pebbles cannot explore all graphs [7]. On the other hand, it is known that a robot with unbounded memory can explore all graphs, using only one pebble [5]. An important issue is to bound the size of the robot as a function of the size of the explored graphs.

A *δ-p-robot with a pebble* or simply *p-robot* when δ is understood, is an automaton $A = (\Delta, \mathcal{S}, f, s_0, s_f)$, with a finite set of states $\mathcal{S}$, $s_0, s_f \in \mathcal{S}$, and

$$f : \mathcal{S} \times \{0,1\} \rightarrow \mathcal{S} \times \Delta \times \{pick, drop\}.$$

Every state $s \in \mathcal{S}$ has a component $p(s) \in \{0,1\}$ that indicates if A has the pebble, $p(s) = 1$, or not, $p(s) = 0$. For the *initial state,* s_0, $p(s_0) = 0$; for the *stop state,* s_f, $p(s_f) = 1$. Each node v of the graph is in some state $p(v) \in \{0,1\}$ that indicates if the pebble is in v, $p(v) = 1$, or not, $p(v) = 0$. The initial state of the graph satisfies: $p(v) = 1$ for exactly one node v. We will assume the robot is placed initially in the node with the pebble.

The movement of a δ-p-robot A on a δ-regular graph is represented by a sequence of *configurations*, each one consisting of the state of the robot and the state of the graph. For the initial configuration, A is placed on some node of the graph in state s_0, and the pebble is in exactly one node. In general, if A is in a node v in state s in some configuration, we compute $f(s, p(v)) = (s', i, b)$. In the next configuration A will be in the node v' such that the edge $\{v, v'\}$ is colored i, in state s'. Also in the next configuration: if $b = drop$ then $p(v) = 1$ and $p(s') = 0$, and if $b = pick$ then $p(v) = 0$ and $p(s') = 1$. It is assumed that b can be equal to *drop* only if $p(s) = 1$ and b can be equal to *pick* only if $p(v) = 1$.

A robot A *explores with stop* a graph if after starting in any node of the graph that has the pebble, it traverses all its edges and enters a stop state. A graph which A does not explore with stop is called a *trap* for A.

The next theorem shows that a p-robot that performs exploration with stop in all graphs of at most n nodes requires $\Omega(n^{1/3})$ states, or equivalently $\Omega(\log n)$ bits of memory.

Theorem 2. *For any K-state p-robot there exists a trap of size $O(K^3)$.*

Proof. Let $A = (\Delta, \mathcal{S}, f, s_0, s_f)$ be a K-state p-robot. We construct a trap of size $O(K^3)$ for A. For that purpose, we consider the restriction of A to states s

such that $p(s) = 0$ and input 0 (on nodes with no pebble). This defines a robot (with no pebble, as in Section 2.1) except that some states may be unreachable from s_0. For every state s of this robot, we consider the robot A_s that has s as initial state, and includes only reachable states from s. Let $\mathcal{A} = \{A_s\}$ be the set of all these robots. Thus, $|\mathcal{A}| \leq K$.

Let G be a graph satisfying Theorem 1 for the set $\mathcal{A}$. Remove edges $\{u, u'\}$ and $\{v, v'\}$ from G. Consider two copies of the resulting graph, with the four nodes of degree 2 indexed by the index of the copy, 1 and 2. These nodes are re-connected as follows. Let c be the color of the deleted edge $\{v, v'\}$. Create two edges $\{v_1, v_2'\}$ and $\{v_1', v_2\}$ with color c. The resulting graph is denoted by G_1 (see Figure 4).

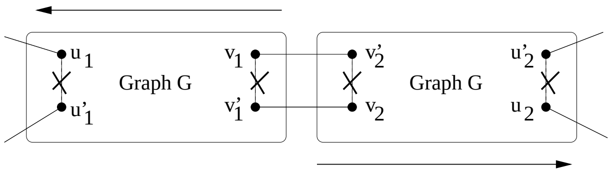

Fig. 4. The graph G_1

Consider an infinite ternary tree modified as follows. Each node is replaced by a 6-cycle. Edges of the cycles are labeled alternatively 1 and 2. Then, edges of the infinite tree are replaced by two "parallel" edges labeled 0, as depicted on Figure 5. The resulting graph is denoted by T.

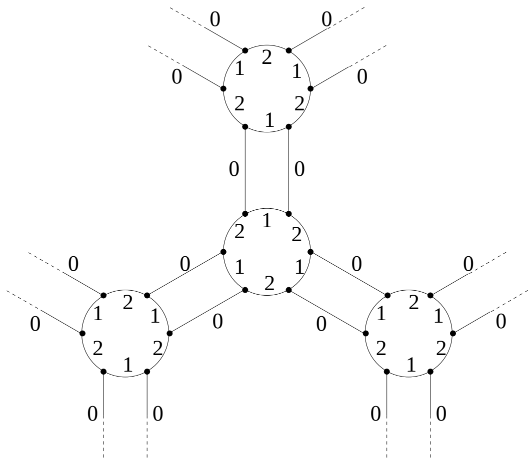

Fig. 5. The modified infinite tree T

The two graphs G_1 and T are composed by replacing every pair $\{\{x, y\}, \{x', y'\}\}$ of parallel edges in T by a copy of G_1. More precisely, x, y, x', y' are respectively connected to nodes u_1, u_2', u_1', u_2 in G_1. These new edges are labeled 0. The resulting graph is denoted by G_2. A "meta-edge" of G_2 is defined as a copy of G_1 replacing a parallel edge of T.

By definition of G and $\mathcal{A}$, the p-robot A is unable to traverse a meta-edge of G_2 without the help of the pebble[2]. We now modify G_2 to obtain a graph G_3 such that the p-robot A is unable to explore G_3, even with the pebble. G_3 contains $O(K)$ 6-cycles of T, and thus has at most $O(K^3)$ nodes. The transformation from G_2 to G_3 is technical and very similar to the transformation used in [6] and in [7]. Thus we only sketch the construction of G_3, skipping technical details. Since any p-robot cannot go from a 6-cycle to another 6-node cycle of G_2 without using the pebble, we define *key* steps as those for which the last time the p-robot leaves a 6-cycle with the pebble, go through a meta-edge, and enters another 6-cycle with the pebble. Because the number of states is finite, A will eventually be twice in the same state at these key steps, at two nodes w and w'. With the same technique as in [6], we identify the nodes w and w'. This leads to the graph G_3 with the desired properties, that is G_3 has $O(K)$ 6-cycles, and thus $O(K)$ "parallel" edges. In each pair of "parallel" edges, there is a copy of G_1. Since G_1 has $O(K^2)$ nodes, then G_3 has $O(K^3)$ nodes. □

Theorem 3. *There exists an exploration with stop algorithm which requires $O(D \log \Delta)$ bits of memory when performed in the family of graphs with diameter at most D and degree at most Δ.*

Proof. We describe an algorithm called `DFS-with-stop`, that enables a robot to explore all graphs, with stop. Exploration is achieved by a traversal of the graph similar to DFS. Let u_0 be the initial position of the robot. The pebble is dropped at u_0, and will remain there until exploration is completed. The exploration proceeds in a sequence of phases. At phase $i \geq 1$, the robot performs a DFS at depth i. At any time during each phase the robot keeps in memory the current sequence of port numbers leading back to u_0 in the DFS tree. This takes $O(i \log \Delta)$ bits of memory during phase i, in a graph of maximum degree Δ. At the beginning of Phase i, the robot sets the variable *stop* $\leftarrow$ *true*. The robot traverses the edges incident to a node u in increasing order of their labels. When the robot leaves the current node u, and enters some node v, it proceeds as follows. If the pebble is at v, then the robot backtracks. Otherwise, if the current depth of the DFS is $\leq i - 1$, then the robot carries on the DFS traversal. If the current depth of the DFS is equal to i, then the robot checks whether v has already been visited or not during a previous phase. For that purpose, the robot performs an auxiliary DFS of depth $i - 1$ from v. This again requires $O(i \log \Delta)$ bits of memory for storing the sequence of port numbers leading back to v in the auxiliary DFS tree. If the robot finds the pebble during the execution of the

[2] Since the $\{u, u'\}$ edges are "open", the proof requires to consider the last time the p-robot is in a u node; this is deferred to the full version of the paper.

auxiliary DFS, then v is at distance $\leq i-1$ from u_0, and thus it has already been explored during a previous phase. If the robot does not find the pebble during the execution of the auxiliary DFS from v, then it sets the variable $stop \leftarrow false$. After completion of the DFS at Phase i, the robot stops if and only if $stop = true$. Else, it carries on exploration, by starting Phase $i+1$. Clearly, the robot stops after Phase $D+1$ in a graph of diameter D. The memory requirement of this exploration algorithm is dominated by the storage of the sequences of port labels corresponding to two paths (one for the DFS, one for the auxiliary DFS). These paths are of length at most $D+1$ in the family of graphs with diameter D, and thus contributes for $O(D \log \Delta)$ when the degree of the graph is at most Δ. □

5 Conclusions

We have proved that exploration with stop (using one pebble) requires $\Omega(\log n)$ bits for the family of graphs with at most n nodes. In [6], the same lower bound holds for perpetual exploration. In fact, [6] proves that perpetual exploration requires $\Theta(D \log \Delta)$ for the family of graphs of diameter at most D and degree at most Δ. This latter result is obtained by proving that DFS-exploration is space-optimal. We thus ask the following question: is $\Omega(D \log \Delta)$ bits of memory required to explore with stop all graphs of diameter at most D and degree at most Δ?

References

1. M. Bender, A. Fernandez, D. Ron, A. Sahai and S. Vadhan. The power of a pebble: Exploring and mapping directed graphs. *Information and Computation* **176**: 1–21, 2002. Prel. Version in STOC 1998.
2. 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.
3. L. Budach. Automata and labyrinths. *Math. Nachrichten,* pages 195–282, 1978.
4. 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.
5. 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.
6. 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, pages 451–462, 2004.
7. H.-A. Rollik. Automaten in planaren graphen. *Acta Informatica* **13**: 287–298, 1980.
8. C.-E. Shannon. Presentation of a Maze-Solving Machine. In 8th Conf. of the Josiah Macy Jr. Found. (Cybernetics), pages 173–180, 1951.