The reduced automata technique for graph exploration space lower bounds*

Pierre Fraigniaud[1] **, David Ilcinkas[2] **, Sergio Rajsbaum[3] ***, and Sébastien Tixeuil[4] †

[1] CNRS, LRI, Université Paris-Sud, France
pierre@lri.fr
[2] LRI, Université Paris-Sud, France
ilcinkas@lri.fr
[3] Instituto de Matemáticas, Univ. Nacional Autónoma de México, Mexico
rajsbaum@math.unam.mx
[4] LRI & INRIA, Université Paris-Sud, France
tixeuil@lri.fr

Abstract. We consider the task of exploring graphs with anonymous nodes by a team of non-cooperative robots, modeled as finite automata. For exploration to be completed, each edge of the graph has to be traversed by at least one robot. In this paper, the robots have no a priori knowledge of the topology of the graph, nor of its size, and we are interested in the amount of memory the robots need to accomplish exploration, We introduce the so-called *reduced automata technique*, and we show how to use this technique for deriving several space lower bounds for exploration. Informally speaking, the reduced automata technique consists in reducing a robot to a simpler form that preserves its "core" behavior on some graphs. Using this technique, we first show that any set of $q \geq 1$ non-cooperative robots, requires $\Omega(\log(\frac{n}{q}))$ memory bits to explore all n-node graphs. The proof implies that, for any set of q K-state robots, there exists a graph of size $O(qK)$ that no robot of this set can explore, which improves the $O(K^{O(q)})$ bound by Rollik (1980). Our main result is an application of this latter result, concerning *terminating* graph exploration with one robot, i.e., in which the robot is requested to stop after completing exploration. For this task, the robot is provided with a pebble, that it can use to mark nodes (without such a marker, even terminating exploration of cycles cannot be achieved). We prove that terminating exploration requires $\Omega(\log n)$ bits of memory for a robot achieving this task in all n-node graphs.

* A preliminary version of this paper appears in the proceedings of the 12th International Colloquium on Structural Information and Communication Complexity (SIROCCO), Mont Saint-Michel, France, May 24-26, 2005, as part of [13].

** Supported by the INRIA project "Grand Large", and the projects "PairAPair" of the ACI "Masses de Données", and "FRAGILE" of the ACI "Sécurité et Informatique".

*** Supported by LAFMI and PAPIIT projects. Part of this work was done while visiting LRI, Univ. Paris Sud, Orsay.

† Supported by the INRIA project "Grand Large". Additional support from the project "FRAGILE" of the ACI "Sécurité et Informatique".

1 Introduction

The problem of exploring an unknown environment occurs in a variety of situations, like robot navigation, network maintenance, resource discovery, and WWW search. In these situations the entities performing exploration can be either a physical mobile device or a software agent. In this paper, we restrict our attention to the case where the environment in which the mobile entities are moving is modeled as a graph. At an abstract level, graph exploration is the task where one or more mobile entities, called *robots* in this paper, are trying to collectively traverse every edge of a graph. In addition to the aforementioned applications, graph exploration is important due to its strong relation to complexity theory, and in particular to the undirected *st*-connectivity (USTCON) problem (cf., e.g., [6]). Given an undirected graph G and two vertices s and t, the USTCON problem is to decide whether s and t are in the same connected component of G. The directed version of the problem is denoted STCON. It is known that STCON is complete for NL, the class of non-deterministic log-space solvable problems. Whether USTCON is complete for L, the class of problems solvable by deterministic log-space algorithms, has been a challenging open problem for quite a long time, and it is only very recently that Reingold proved that USTCON is indeed complete for L [15]. Note that the existence of a finite set of finite-state automata able to explore all graphs would have put USTCON in L, and proving or disproving this existence had therefore motivated quite a long sequence of studies. Cook and Rackoff [6] eventually proved that even a more powerful machine, called JAG, for "Jumping Automaton for Graphs", cannot explore all graphs (a JAG is a finite set of globally cooperative finite-state automata enhanced with the ability, for every automaton, to "jump" from its current position to any node occupied by another automaton). Since this latter result, the exploration graph problem is focussing on determining the space complexity of robots able to explore all graphs.

As far as upper bounds in concerned, Reingold showed in [15] that his log-space algorithm for USTCON implies the existence of log-space constructible *universal exploration sequences* (UXS) of polynomial length. Roughly speaking, a UXS [14] is a sequence of integers that (1) tell a robot how to move from node to node in a graph (the exit port at the kth step of the traversal is obtained by adding the kth integer of the UXS to the entry port), and (2) guarantee to explore every node of a graph of appropriate size (a UXS is defined for a given size, and a given degree). Rephrasing this latter result, there is a $O(\log n)$-space robot that explores all the graphs of size n. The extend to which this bound can be decreased by using a set of $q > 1$ cooperative robots remains open. Also, the question of the existence of log-space constructible *universal traversal sequences* (UTS) [1] remains open (a UTS is a sequence of port-numbers so that the output port at the kth step of the traversal is the kth element of the sequence).

As far as lower bounds are concerned, most papers are dealing with the design of small *traps* for arbitrary teams of robots, i.e., small graphs that no robot of the team can explore. (Formally, a trap consists of a graph and a node from where the robots start the exploration.) The first trap for a finite state robot is generally attributed to Budach [5] (the trap is actually a planar graph). The trap constructed by Budach is however of large size, and a much smaller trap was described in [12] which proved that, for any K-state robot, there exists a trap of at most $K+1$ nodes. In [16], Rollik proved that no finite set of finite locally-cooperative automata, 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 (such robots may have different transition functions, hence they will follow different paths in the explored graph). This latter trap is of size $O(K^{O(q)})$ nodes. Rollik's trap for cooperative robots is even larger: $\tilde{O}(K^{K^{\cdot^{\cdot^{K}}}})$ nodes, with $2q+1$ levels of exponentials where the $\tilde{O}$ notation hides logarithmic factors. In this paper, we present a new lower bound technique for graph exploration, called *reduced automata technique*. Roughly, this technique consists in reducing a robot to a simpler form that preserves its "core" behavior on some graphs: except for some easily described closed paths, the reduced robot follows the path of the original robot, on any such graph.

The interested reader can find other pointers to the literature in, e.g., [3–5, 7, 8, 12]. To complete the picture, and before describing our results in more details, let us point out that Shimon Even, whom this book is dedicated to, was interested in graph exploration problems early on in his career. In particular, in his 1976 seminal paper with Tarjan [11], he presented a way of numbering nodes during a DFS traversal that proved to be useful in many algorithms. In collaboration with A. Litman and P. Winkler [10], he then studied traversal in directed networks. With G. Itkis and S. Rajsbaum [9], he described a traversal strategy for undirected graphs that constructs a subgraph with good connectivity but few edges. And recently, in collaboration with S. Bhatt, D. Greenberg, and R. Tayard [2], he studied the problem of using a robot as simple as possible (with access to some local memory stored in the vertices) to find an Eulerian cycle in mazes and graphs.

1.1 Problem settings

As in [6, 16], we are interested in exploration of undirected graphs where nodes are not uniquely labeled. Note that, besides the theoretical interest of understanding when or at what cost such graphs can be explored, the unlabeled-node setting can occur in practice, due to, e.g., privacy concerns, limited capabilities of the robots, or simply anonymous edge intersections. The robots, modeled as a deterministic automata, can however identify the edges incident to a node through unique port labels, from 1 to the degree of the node. We consider two types of exploration:

- Perpetual exploration, in which the task of the robots is to, eventually, traverse all edges.
- Terminating exploration, in which the robots, after completing exploration, must eventually stop.

In acyclic graphs, terminating exploration is strictly more difficult than perpetual exploration. In particular, it is shown in [7] that terminating exploration in n-node bounded degree trees requires a robot with memory size $\Omega(\log\log\log n)$, whereas perpetual exploration is possible with $O(1)$ bits. In arbitrary graphs, terminating exploration cannot be achieved. Indeed, 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, because nodes are not uniquely labeled. That is, there are graphs that a robot can explore perpetually, but it can never stops. Thus, as in previous work in this setting, e.g., [3, 4, 8], we assume that, for terminating exploration, robots can mark nodes: a robot can drop a pebble in a node and later identify it and pick it up.

Following the common conventions in the literature, the robots aiming at performing perpetual exploration are not given pebbles, whereas robots aiming at performing terminating exploration are given one or more pebbles. As a consequence, the two problems becomes incomparable. Indeed, terminating exploration is more demanding than perpetual exploration, but the "machines" designed for these two tasks do not have the same power.

A *team* of robots is a set of deterministic automata with possibly different transition functions, all starting from the same starting point. When sets or teams of robots are considered, the robots of a team can communicate in various manners. Four cases are considered in the literature:

- Non-cooperative robots: the robots are oblivious of each other, and each of them acts independently from the others.
- Locally cooperative robots: robots meeting at a node can exchange information, including their identities and their current states.
- Globally cooperative robots: the robots are perpetually aware of the states of the others, of whether they meet and who they meet, and of the degrees of the nodes occupied by the robots.
- Jumping Automaton: the robots are globally cooperative, and any robot is able to jump from the node it is currently occupying to a node currently occupied by any other robot.

In this paper, we restrict our attention to the two weakest models: non-cooperative robots, and locally cooperative robots.

1.2 Our results

In this paper, we present a new lower bound technique for graph exploration, called *reduced automata technique*. Based on this technique, the lower bounds presented in this paper are obtained as follows. Assume a set of q robots is given. Then construct the smallest possible graph, called a *trap* for this set of robots, such that if the robots are placed in some specified nodes of the graphs, then there is at least one edge that is not traversed by any of the robots. If the q robots have K states each, and the trap has $f_q(K)$ nodes, then the space lower bound for a set of q robots exploring all n-node graphs is $\Omega(\log f_q^{-1}(n))$ bits.

The reduced automata technique for the design of space lower bounds for graph exploration is described in Section 2. This lower bound technique allows us to concentrate on a subclass of graphs, called *homogeneous:* edge-colored and regular. For such graphs, a robot can be described by a very simple automaton, whose transition function consists of a graph formed by a directed path followed by a directed cycle. The reduced automata technique applies to homogeneous graphs. Roughly speaking, a *reduced robot* has the property that if it traverses an edge $\{u,v\}$ at some step of the exploration, say from u to v, then its next move will not be traversing the edge back to u. This property is achieved by transforming a robot into a reduced robot whose transition function never has two consecutive edges with the same label. We construct a *trap core* directly from the transition function of a reduced robot, which is then easily extended to a trap for the original robot.

In Section 3 we use the technique of reducing a robot to construct a degree 3 trap for a K-state robot, of size $K+3$. The proof technique can be generalized to produce traps of any degree, but for illustrating the technique, it is sufficient to work with degree 3 graphs. Indeed, [12] presents a trap of size $K+1$, planar and valid for graphs of any degree. The proof we present is somewhat simpler than the one of [12], and moreover, it illustrates the technique used to prove our results in the following sections.

In Section 4 we present our new results about traps for collective exploration by a set of *non* cooperative robots. The robots do not communicate at all, and every edge must be traversed by at least one robot. We show (cf. Theorem 4) 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. This improves the $O(K^{O(q)})$ bound of Rollik [16] (cf. Corollary 2).

By simply plugging this new trap for non-cooperative robots into Rollik's construction, we get (cf. Corollary 4) a new trap for locally-cooperative exploration of size $\tilde{O}(K^{K^{\cdot^{\cdot^{K}}}})$ with $q+1$ levels of exponential, to be compared with the $2q+1$ levels of [16]. Our trap is thus smaller than the one in [16].

In Section 5 we show that Theorem 4 has a significant impact on the space complexity of terminating graph exploration by a single robot. As mentioned above, when terminating exploration is required, the robot is provided with a pebble. We prove (cf. Theorem 5) that terminating exploration requires a robot with $\Omega(\log n)$ bits for the family of graphs with at most n nodes. As mentioned before, in arbitrary graphs, perpetual exploration and terminating exploration are not comparable because even if perpetual exploration is a simpler task than terminating exploration, in the latter case the robot is given a pebble. Therefore, even if the existence of traps with at most $K+O(1)$ nodes for any K-state robot 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, the $\Omega(\log n)$ lower bound for terminating exploration is not a consequence of the first result about perpetual exploration.

2 Preliminaries

In Section 2.1 we define formally what we mean by a robot exploring a graph. In Section 2.2 we describe 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, Robots and Traps

A robot considered in this paper traverses a graph by moving from node to node along the edges of the graph. We first describe the basic model of a robot traversing a graph, and what we mean by a trap, namely a pair (G, u) where G is a graph and $u \in V(G)$ such that a robot starting from u cannot explore G. To construct a trap for a robot, we first design a graph that the robot cannot leave, called a trap core, and then we add to it edges that the robot does not explore. We explain how the description of a robot is simplified when traversing a more symmetric kind of graph, called homogeneous. The simpler description will be crucial in the rest of the paper.

The basic model of a robot traversing a graph. In a graph where nodes have no identifiers, two nodes are indistinguishable to the robot, unless they have different degree. However, edges have local port numbers, so the robot can distinguish two different edges incident to a node. In more detail, each edge has two labels, each one associated to one of its two endpoints. The labels of the edges incident to a node v are arbitrary and pairwise distinct in the set $\{0, \ldots, \delta_v - 1\}$, where δ_v denotes the degree of v. 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.* Notice that an edge may have different port numbers in its two endpoints. We refer to those graphs as *port-labeled graphs.*

A robot is an automaton with a single initial state; at the beginning, it is placed on an arbitrary starting node of the graph in this state. When a robot is in a node u and *traverses* an edge $\{u, v\}$ to get to v, it learns the label at v's endpoint of the edge once it enters v. The robot decides which edge to take to leave v based on this label, as well as on the degree of v, and of course, based on its local state. We do not define formally such a robot because we will study its behavior only on a special class of graphs, called homogeneous, for which a very simple representation of a robot is possible, that we *will* define formally. In Section 5 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 robots is a pair (G, U), where G is a port-labeled graph and U is a set of nodes of G, such that if all the robots are placed in nodes $u \in U$, each in its initial state, then there will be an edge $\{u, v\}$ that is never traversed by the robots. To make our lower bound results stronger, sometimes we present a *simple trap,* namely with no parallel edges and self-loops.

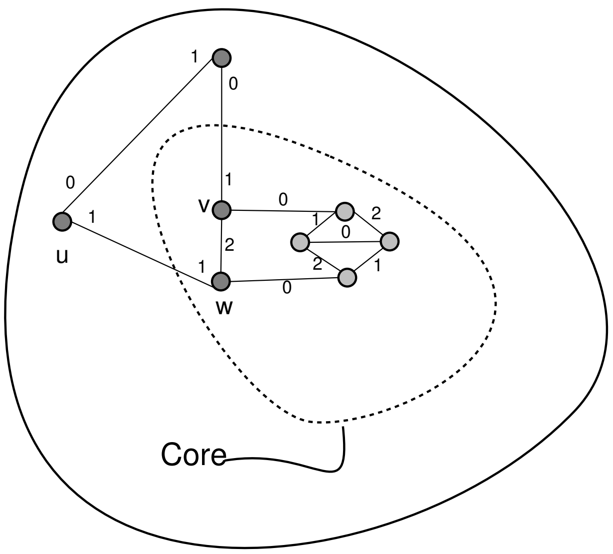

Fig. 1. A trap and its core

Homogeneous graphs and trap cores. We will study the behavior of a robot in a graph where both port numbers of an edge coincide. In such a graph a robot can be described by a very simple automaton, as we shall see next. A δ-*homogeneous* undirected graph is a graph that is δ-regular and δ-edge-colored. A graph is δ-*regular* if each of its nodes has degree δ, and it is δ-*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. For the sake of clarity, we mainly focus on graphs with maximum degree three.

When a robot traverses a 3-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, or 2. Thus, the robot decides which edge to take to exit the node based only on its current state. Formally, a *robot* is an automaton $A = (\Delta, \mathcal{S}, f, \hat{s})$, with a finite set of states $\mathcal{S}$, an initial state $\hat{s} \in \mathcal{S}$, 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 robot A moves on a 3-regular graph as follows. Initially A is placed on a node of the graph in state $\hat{s}$. 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 $f_{st}(s)$.

When considering the formal definition of a robot for homogeneous graphs, one can construct a trap by first defining a graph G that is edge-colored, but not necessarily 3-regular, and then adding some edges and nodes to obtain a trap in which the trap core looks homogeneous to the robot. We do not demand that a trap is 3-homogeneous as long as a robot never tries to take an edge that is not defined in the graph. Formally:

Definition 1 (Trap Core). *A* trap core *for a set of robots is a pair* (G, U), *where* G *is a* 3*-edge-colored graph and* U *is a set of nodes of* G, *such that if all the robots are placed in nodes* $u \in U$, *each in its initial state, then each time a robot* $A = (\Delta, \mathcal{S}, f, \hat{s})$ *is in some node* u *in some state* s, *if* $f_\ell(s) = i$ *then an edge* $\{u, v\}$ *labeled* i *must be in* G.

From a trap core to a trap. Once we have built a trap core (G, U) it is not difficult to construct a trap (G', U), by adding to it some edges and a constant number of nodes. Notice that if (G, U) is a trap core for a set of robots, then (G', U) is a trap for the same set of robots, because G is a strict subgraph of G' that the robots never leave. We first show how to construct G' from a 3-edge colored graph G, by adding at most 2 nodes, and adding edges that guarantee that every node of G has degree exactly 3, and we define local port labels for the newly added edges. Thus, as in Figure 1, edges that were originally in G have the same port labels in both endpoints (e.g. $\{v, w\}$ in the figure), while newly added edges may have different port labels (e.g. $\{u, w\}$ in the figure). Afterward we show how to construct an homogeneous G' with at most 13 new nodes.

Definition 2 (Simple Trap Extension). *Given a* 3*-edge-colored graph* $G = (V, E)$, *the labeled simple graph* $G' = (V', E')$, $|V'| \leq |V| + 2$, *obtained from* G *in the following construction is called the* simple trap extension *of* G.

To construct G' first we can assume that there are at most 2 nodes of degree less than 3. Otherwise, there are two nodes of degree less than 3 that are not connected by an edge, and we may add an edge connecting them, with appropriate local port labels. Now, we add at most 2 new nodes. Each time we add a new node, we connect it to nodes with degree less than 3, with appropriate local port labels. If all nodes of G have now degree exactly 3, we are done, else we add a new node and repeat the procedure. At the end we obtain the desired 3-edge colored graph G', where all original nodes have exactly degree 3, while the new nodes have degree at most 3. Moreover, G' is a simple graph.

Remark. Using the same type of arguments as above, it is possible to construct a simple trap extension for arbitrary degree δ, by adding at most $\delta - 1$ nodes.

We can construct a trap extension G^{hom} from G that is homogeneous, by adding a few more nodes.

Definition 3 (Homogenous Extension). *Given a* 3*-edge colored graph* $G = (V, E)$, *the graph* $G^{hom} = (V', E')$, $|V'| \leq |V| + 13$, *obtained from* G *in the following construction is called the* homogeneous extension *of* G.

Add to each node of G of degree i less than 3, $3 - i$ pending "half-edges" colored differently from each other and from the colors of edges incident to the node. For $\ell = 0, 1, 2$, let $parity(\ell)$ be the parity of the number of pending half-edges labeled ℓ in the resulting graph[5] G'.

[5] We will use this notion of "graph" with "half-edges" several times in this paper.

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

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

We now construct the desired homogeneous graph G^{hom}. Let ϱ be the parity of the number of pending half-edges of a given label in G'. If ϱ is odd, then we add to G' a node connected to one of the half-edges, labeled say ℓ, and add two half-edges pending from this node, labeled $\ell' \neq \ell$ and $\ell'' \notin \{\ell, \ell'\}$. As a consequence, ϱ becomes even. Now, we pair the half-edges with identical labels, and connect them to form one edge. Parallel edges can be avoided, unless for some ℓ there are only two pending half-edges with label ℓ, and these are incident to the same edge. In this case the pair is connected by the gadget displayed in Figure 2, where $\ell = 0$. By labeling the edges of every gadget appropriately (as in the figure), we obtain a 3-homogeneous graph G^{hom}.

Claim. G^{hom} has at most 13 nodes more than G.

Proof. We added at most 1 node to correct the parities, and at most 3 gadgets to avoid parallel edges, each one with 4 nodes. Thus the total number of nodes added is at most 13. □

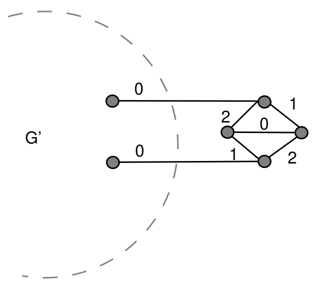

Fig. 2. The gadget for connecting half-edges

Remark. As for the simple trap extension, it is easy to check that one can construct an homogeneous extension for arbitrary degree δ, by using a specific gadgets for every δ.

2.2 Basic Properties

Consider a robot $A = (\Delta, \mathcal{S}, f, \hat{s})$. 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 $(s,t) \in F$ 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 $\hat{s}$ completely determine the robot A. We assume in the rest of the paper that every state $s \in \mathcal{S}$ of A is reachable from $\hat{s}$; unreachable states do not affect the behavior of A and can be ignored. Namely, there is in $G(A)$ a directed path from $\hat{s}$ to every other node.

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 $\hat{s}$ 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 homogeneous graph G in state $\hat{s}$, $fp(A)$ 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. A *configuration* (x,s) denotes the fact that A is in node x in state s. Also, if $f_{st}(s) = s'$, $f_\ell(s) = i$, and the label of the edge $\{x, x'\}$ is i then we write $(x,s) \to (x', s')$.

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

Proof. Assuming A is in state s when it is in x, consider the sequence of configurations starting with (x,s)

$$(x_0, s_0) \to (x_1, s_1) \to \cdots$$

where $(x,s) = (x_0, s_0)$. The sequence of configurations must contain two equal configurations, say $(x_i, s_i) = (x_{i+k}, s_{i+k})$, for some $k > 1$, because both G and A are finite. Assume k is as small as possible. If $i = 0$ we are done, so suppose $i > 0$. We will prove that $(x_{i-1}, s_{i-1}) = (x_{i+k-1}, s_{i+k-1})$, which implies that $(x_0, s_0) = (x_k, s_k)$, and the lemma follows.

Notice that A moves from x_{i-1} to x_i along the edge labeled $f_\ell(s_{i-1})$. Now, when A eventually returns to the same configuration (x_{i+k}, s_{i+k}), the state $s_{i+k-1} = s_{i-1}$ (all the states considered are in the cycle of $G(A)$ because the state s belongs to the cycle of $G(A)$). Thus, $f_\ell(s_{i-1}) = f_\ell(s_{i+k-1})$. It follows that the edge $\{x_{i+k-1}, x_{i+k}\}$ must be labeled $f_\ell(s_{i-1})$. Finally, $x_{i+k-1} = x_{i-1}$ since $x_{i+k} = x_i$ and G is edge-colored. $\square$

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 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' 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 $G(A)$. 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 edge-colored, $\bar{G}(A)$ must be edge-colored. 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 and A' are placed on the same node of a graph, the path traversed by A' is contained in 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 $A = (\Delta, \mathcal{S}, f, \hat{s})$ into another robot $A' = (\Delta, \mathcal{S}', f', \hat{s}')$, $\mathcal{S}' \subseteq \mathcal{S}$, where one of the above properties (i) or (ii) is enforced for two arcs. There are two types of reduction steps, corresponding to the two properties. 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.

Type-i Reduction. A type-i reduction step is *applicable* if $G(A)$ has two consecutive distinct arcs $s \to s_1 \to s_2$ with $f_\ell(s) = f_\ell(s_1)$. The basic idea is illus-

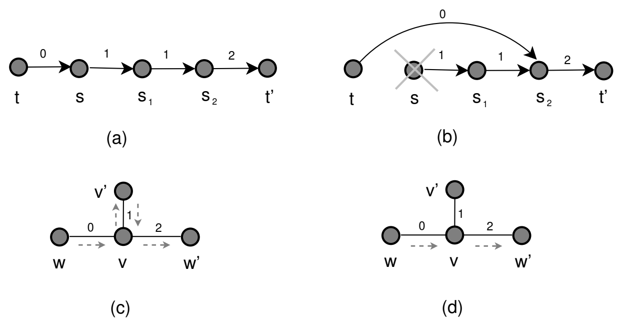

Fig. 3. A type-i reduction

trated in Figure 3. In (a) there is a segment of $G(A)$ with the two consecutive arcs labeled 1, and in (b) there is the corresponding segment of $G(A')$ after the reduction. In this example s has only one in-neighbor, t, and hence s becomes unreachable. This is the basic idea behind the type-i reduction, but in the formal definition below we need to consider several special cases depending on the number of in-neighbors of s, and on where is the initial state $\hat{s}$.

The properties of a type-i reduction that we need are illustrated in Figure 3(c) and (d), where the path traversed by A and A' resp. is depicted in dotted arrows. If A is in node w of G in state t, it moves to node v in state s, and then it moves to v', change to state s_1, and move back to v, in state s_2 (since $f_\ell(s) = f_\ell(s_1) = i$, where $\{v, v'\}$ is colored i; in the figure $i = 1$). 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 distance at most 1 from the nodes explored by the robot after the reduction.

Formally, a *type-i reduction* transforms A into A' by doing the following changes to f and by defining $\hat{s}'$ ($f'(\cdot) = f(\cdot)$ and $\hat{s}' = \hat{s}$ unless specified otherwise below). We consider four cases:

Case $s = s_2$: In this case the cycle is of length 2 with the same labels. Assume w.l.o.g. that s has no other in-neighbor besides s_1 (it is impossible that both s and s_1 have 2 in-neighbors). Let $f'(s_1) = (s_1, i)$, where $i = f_\ell(s_1)$. If $s = \hat{s}$ then $\hat{s}' = s_1$.

Otherwise, if $s \neq s_2$, it is possible that s has $0, 1$, or 2 in-neighbors.

Case $s \neq s_2$, s has 0 in-neighbors: In this case $s = \hat{s}$. Let $\hat{s}' = s_2$.

Case $s \neq s_2$, s has 1 in-neighbor: Let t be the in-neighbor ($t \neq s_1$), with $f(t) = (s, i)$. Then let $f'(t) = (s_2, i)$. If $s = \hat{s}$ then let $\hat{s}' = s_2$.

Case $s \neq s_2$, s has 2 in-neighbors: Assume they are t_1, t_2, with $f(t_1) = (s, i)$, $f(t_2) = (s, j)$. Then $s \neq \hat{s}$. Let $f'(t_1) = (s_2, i)$ and $f'(t_2) = (s_2, j)$.

After doing these modifications, A' is obtained by removing any unreachable states. Notice that for each one of the previous four 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.

Lemma 2. *Let A' be the robot obtained from $A = (\Delta, \mathcal{S}, f, \hat{s})$ by applying a type-i reduction on arcs $s \to s_1 \to s_2$ with $f_\ell(s) = f_\ell(s_1)$. Then*

1. *The node s together with $s \to s_1$ does not appear in A'.*
2. *If A and A' start at the same node u of a graph in the same state s that belongs to their cycle, when A and A' are back in state s, they are placed in the same node v and A has traversed at most one edge more than A'.*

Proof. The first part of the lemma holds because state s becomes unreachable in A'. We now prove the second part of the lemma. We thus consider that A and A' are both started from a node x_0 in a state t_0 that belongs to their cycle.

Assume a type-i reduction is applied to $G(A)$ on the arcs $s \to s_1 \to s_2$ with $f_\ell(s) = f_\ell(s_1)$, to obtain A'. When s has one in-neighbor t, with $f(t) = (s, i)$, and $s \neq \hat{s}$, A' is equal to A except that $f'(t) = (s_2, i)$ (and hence s becomes unreachable).

Consider the sequence of configurations of $G(A)$ when starting in a node x_0,

$$(x_0, t_0) \to (x_1, t_1) \to \cdots$$

where $t_0 = \hat{s}$. Then the sequence of configurations of $G(A')$ is the same, except that each time A gets to state t, say in the i-th step

$$\cdots \to (x_i, t_i) \to (x_{i+1}, t_{i+1}) \to (x_{i+2}, t_{i+2}) \to \cdots$$

where $t_i = t$, and hence $t_{i+3} = s_2$ with $x_{i+1} = x_{i+3}$ (since $f_\ell(t_{i+1}) = f_\ell(t_{i+2})$), then the sequence of $G(A')$ is

$$\cdots \to (x_i, t_i) \to (x_{i+3}, t_{i+3}) \to \cdots$$

Therefore, the original path in the graph

$$x_0, x_1, \ldots, x_i, x_{i+1}, x_{i+2}, x_{i+3}, \ldots$$

becomes

$$x_0, x_1, \ldots, x_i, x_{i+3}, \ldots$$

and the loop $x_{i+1}, x_{i+2}, x_{i+3}$ ($x_{i+1} = x_{i+3}$) traversing the edge $\{x_{i+1}, x_{i+2}\}$ back and forth is eliminated from the path. □

Type-ii Reduction. Once a type-i reduction step is not applicable in $G(A)$, a single type-ii reduction can be used. A type-ii reduction step is *applicable* if $G(A)$ has 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 $f_\ell(s) = f_\ell(s_1)$. See Figure 4 where $f_\ell(s) = f_\ell(s_1) = 1$; in part (a) there is $G(A)$, and in part (b) there is $G(A')$ after the reduction. A *type-ii reduction* transforms A into A' by doing the following

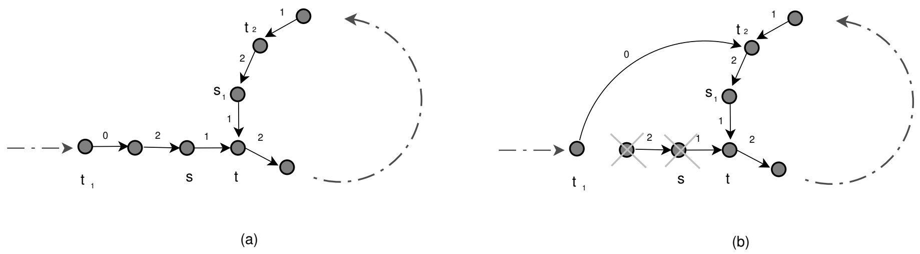

Fig. 4. A type-ii reduction

changes to f and by defining $\hat{s}'$ ($f'(\cdot) = f(\cdot)$ and $\hat{s}' = \hat{s}$ unless specified otherwise below). 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. This path is of length at least 1, 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 $fp(A) = W_0W^*$. Let W' be the longest common postfix of W_0 and W^*; $|W'| > 0$ by the type-ii assumption. Let t_2 be the node just before W' starts in the cycle of $G(A)$. In Figure 4, $W' = 21$. We consider two cases, in both cases A' is obtained from A by the following modifications, and removing any unreachable states:

Case $|W_0| > |W'|$**:** That is, W' is a strict postfix of W_0; let t_1 be the in-neighbor of the node just before W' starts in the simple path of $G(A)$. Thus $f_\ell(f_{st}(t_1)) = f_\ell(t_2)$ is the first letter of W'. Let $f'(t_1) = (t_2, f_\ell(t_1))$.
Case $|W_0| = |W'|$**:** Let $\hat{s}' = t_2$.

The following lemma is straightforward.

Lemma 3. *Let $A' = (\Delta, \mathcal{S}', f', \hat{s}')$ be the robot obtained from $A = (\Delta, \mathcal{S}, f, \hat{s})$ by applying a type-ii reduction on arcs $s \to t$, $s_1 \to t$, with $f_\ell(s) = f_\ell(s_1)$, and s_1 in the cycle of $G(A)$. Then*

1. *The node s together with $s \to t$ does not appear in A'. Moreover, a type-ii reduction is not applicable to $G(A')$.*
2. *If A and A' start at the same node of a graph, they both traverse the same path.*
3. *If a type-i reduction is not applicable to $G(A)$ then it is not applicable to $G(A')$.*

Using Lemma 2 and Lemma 3 it is easy to prove the following, summarizing the procedure to obtain an irreducible robot.

Lemma 4. *Let $A' = (\Delta, \mathcal{S}', f', \hat{s}')$ be the robot obtained from $A = (\Delta, \mathcal{S}, f, \hat{s})$ through the longest possible sequence of type-i reductions followed by a type-ii reduction (if applicable). Let k be the number of reduction steps in this sequence.*

1. *A' is irreducible.*
2. *$|\mathcal{S}'| + k \leq |\mathcal{S}|$.*
3. *If A and A' start at the same node u of a graph in the same state s that belongs to their cycle, when A and A' are back in state s, they are placed in the same node v and A has traversed at most k edge more than A'.*

Proof. The first part of the lemma follows from Lemma 2(1) and from Lemma 3(1,3): if there are two arcs violating property (i), then a type-i reduction can be applied, and at least one state is removed in the process. Also, if there are two arcs violating property (ii) after all arcs satisfy property (i), then a type-ii reduction will eliminate the situation, without creating arcs that violate property (i).

The second part of the lemma follows because each type-i and type-ii reduction eliminates at least one state, as observed in Lemma 2(1) and Lemma 3(1).

The third part of the lemma follows from Lemma 2(2) and Lemma 3(2), by induction on k. □

3 A Trap for a Single Robot

In this section, we focus on graph exploration by a single robot. We present a trap for a K-state robot of size $O(K)$. As explained in the Introduction, a similar result was presented in [12]. We consider a robot and an irreducible version of it. First we show how to construct a trap core for the irreducible robot, and then how to extend it to a trap for the original robot.

3.1 A Trap for an Irreducible Robot

Let $\widehat{A} = (\Delta, \mathcal{S}, f, \hat{s})$ be an irreducible robot with footprint $fp(\widehat{A}) = W_0W^*$, $|W_0W| = K$. Recall that its graph of state transitions $G(\widehat{A})$ consists of a directed path starting in the initial state $\hat{s}$, followed by a directed cycle. Thus, the corresponding undirected graph, $\bar{G}(\widehat{A})$, consists of a path P connected to a cycle C; let $\hat{x}$ be the initial node of P. If C is of length at least 3 then $\bar{G}(\widehat{A})$ is a simple edge-colored graph (no parallel edges and no self-loops), and it serves as a trap core (Definition 1) for $\widehat{A}$. If C is of length less than 3 we modify it a little to make it a simple edge-colored graph that is also a trap for $\widehat{A}$, denoted $\bar{G}_1(\widehat{A})$.

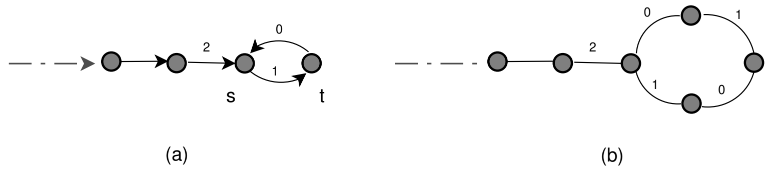

Fig. 5. Eliminating parallel edges

We construct the simple, edge-colored graph $\bar{G}_1(\widehat{A})$ from $\bar{G}(\widehat{A})$ as follows:

- Assume the directed cycle of $G(\widehat{A})$ is of length 2, with states s and t (Figure 5(a) illustrates this case with $W = 10$). Then the undirected cycle in $\bar{G}_1(\widehat{A})$ will have 4 edges, labeled WW, adding two new nodes as in Figure 5(b). The path is P, as in $\bar{G}(\widehat{A})$.
- Assume the directed cycle of $G(\widehat{A})$ is of length 1 with state s (Figure 6(a) illustrates this case with $W = 1$). Then the undirected cycle in $\bar{G}_1(\widehat{A})$ will have 4 edges, labeled $abab$, where a is equal to the single letter of W and b is different from a and from the last letter of W_0 (if any), as in Figure 6(b), where $abab = 1010$. The path is P, as in $\bar{G}(\widehat{A})$.

Notice that the only node of $\bar{G}_1(\widehat{A})$ of degree 3 is the node where the path and the cycle are joined. Thus, it is not homogeneous, and if we place a robot in one of its nodes, it could try to take an edge that does not exist in the graph. Clearly, this does not happen if we place $\widehat{A}$ at $\hat{x}$. Namely, starting at $\hat{x}$, $\bar{G}_1(\widehat{A})$ with any edge added is a trap core for $\widehat{A}$, with at most 3 nodes more than $\bar{G}(\widehat{A})$. We have the following straightforward lemma.

Lemma 5. *The graph $\bar{G}_1(\widehat{A})$ is simple and edge-colored, with at most $|\mathcal{S}| + 3$ nodes. Moreover, $\bar{G}_1(\widehat{A})$ is a trap core for $\widehat{A}$ when starting at $\hat{x}$.*

Fig. 6. Eliminating a self-loop

3.2 A Trap for the Original Robot

We present two different constructions of a trap for A. In both cases we use the graph $\bar{G}_1(\widehat{A})$ of Lemma 5, where $\widehat{A}$ is an irreducible robot obtained from A. The first method, described in Theorem 1, produces a smaller trap than the second, described in Theorem 2, but the second method will be useful in the following section.

Theorem 1. *For any robot $A = (\Delta, \mathcal{S}, f, s_0)$ there exist a trap of at most $|\mathcal{S}|+2$ nodes, and an homogeneous trap of at most $|\mathcal{S}| + 13$ nodes.*

Proof. Let $\widehat{A}$ be an irreducible robot obtained from A, and consider its undirected graph $\bar{G}(\widehat{A})$. By Lemma 5 the modified graph $\bar{G}_1(\widehat{A})$ is simple and edge-colored. Also, $\widehat{A}$ can be placed in the first node $\hat{x}$ of the path P of $\bar{G}_1(\widehat{A})$ in its initial state, and it never tries to take an edge not in the graph. Now, place A in $\hat{x}$ in its initial state. Each time A wants to take an edge with some label not in the graph, we add the edge (with the label) to the graph. By Lemma 2 the paths traversed by A (and not by $\widehat{A}$) are trees where A stays in states eliminated by the series of type-i reductions. Thus, the added edges form trees, and the nodes added correspond to the eliminated states, so we get back a graph with $|\mathcal{S}|$ nodes. Now, A never tries to take an edge not in the graph. For this graph, consider the simple extension of Definition 2, and the homogeneous extension as in Definition 3. The first is a trap $(G, \hat{x})$ for A with at most 2 additional nodes, while the second is a homogeneous trap $(G, \hat{x})$ for A with at most 13 additional nodes. $\square$

Remark. Using extensions for arbitrary degree allows us to obtain a similar result as the one in [12]. In fact, the extension used in [12] outputs a graph which is neither simple nor homogeneous. It is however of smaller size: $|S| + 1$ nodes, independently from the considered degree.

The second way of constructing a trap uses the *K-tower method.* Assume an homogeneous graph H is given, together with one of its edges, say $\{v, v'\}$. Cut the edge to produce two pending half-edges e, e'. Add a "tower" of height $K + 1$ connected to e, e', and a gadget closing the tower as in Figure 7. The two internal nodes of the gadget at the top of the tower are denoted by v_1 and v_1'. Add labels to the tower and the gadget to make the whole graph edge-colored, and denote it G. Thus, G is homogeneous.

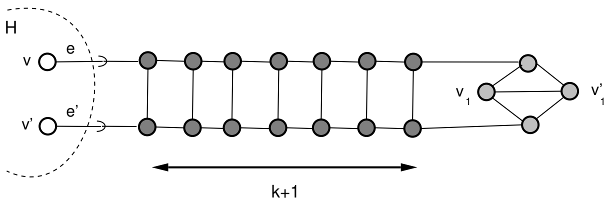

Fig. 7. The tower method

Theorem 2. *For any robot $A = (\Delta, \mathcal{S}, f, s_0)$ there exist an homogeneous trap (G, U) of at most $3|\mathcal{S}| + 22$ nodes.*

Proof. Let $\widehat{A}$ be an irreducible robot obtained from A, and consider its undirected graph $\bar{G}(\widehat{A})$. By Lemma 5 the modified graph $\bar{G}_1(\widehat{A})$ is simple and edge-colored. Also, $\widehat{A}$ can be placed in the first node $\hat{x}$ of the path P of $\bar{G}_1(\widehat{A})$ in its initial state, and it never tries to take an edge not in the graph. Consider the homogeneous extension H of $\bar{G}_1(\widehat{A})$, as in Definition 3, with at most 13 additional nodes. Pick any of the new edges added to H, say $\{v, v'\}$, and add a tower of height $K + 1$, $K = |\mathcal{S}|$, as described above, to obtain G. Now, place A in x_0 in its initial state. Notice that as G is homogeneous, A never tries to take an edge not in the graph. Finally, the edge $\{v_1, v_1'\}$ is not traversed by A. This is because $\widehat{A}$ does not traverse the edge $\{v, v'\}$, and hence it does not enter the tower. By Lemma 4, the trajectory of A is never at distance greater than K (where $K = |\mathcal{S}|$) 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 $\{v_1, v_1'\}$, and $(G, \hat{x})$ is a trap for A.

It remains to count the number of nodes of G. The graph $\bar{G}(\widehat{A})$ has at most K nodes, $\bar{G}_1(\widehat{A})$ has at most $K + 3$ nodes. Then, H has at most $K + 16$ nodes. The tower has $2K + 6$ nodes, so the total is at most $3K + 22$ nodes. □

Corollary 1. *A robot that explores all graphs of size n requires at least $\Omega(\log n)$ memory bits.*

Using a different proof argument, [12] also proves a lower bound that depends on the diameter and the maximum degree of the graph, rather than just the number of nodes. It is, nevertheless, possible to use the trap core proof method to obtain a similar result. We recall the theorem from [12]:

Theorem 3. *A robot that explores all graphs of diameter D and maximum degree δ requires exactly $\Theta(D \log \delta)$ memory bits.*

4 A Trap for a Team of Non-Cooperative Robots

In this section, we focus on graph exploration by a team of non-cooperative robots. (The independent robots may have different transition functions, hence they will follow different paths in the explored graph.) The main result of the section is the construction of a trap of size $O(qK)$ for any set of q non-cooperative K-state robots. This result is stated in Theorem 4. To prove this result, we first need an auxiliary lemma (Lemma 6) that shows how, given any automaton, any homogeneous graph can be transformed into a trap for this automaton. This result is used at every induction step of the proof of Theorem 4.

4.1 Trapping an Irreducible Robot

In this section we prove an auxiliary result for Theorem 4. Assume an irreducible K-state robot $\widehat{A} = (\Delta, \mathcal{S}, f, \hat{s})$ is placed at a node x_0 of a graph G in its initial state $\hat{s}$, and we want to create a trap core for $\widehat{A}$ by extending G at a given edge $\{v, v'\}$, and moreover, the extension should be of size $O(K)$. (if $\widehat{A}$ does not traverse the edge then there is nothing to be done.) See Figure 8. The extension

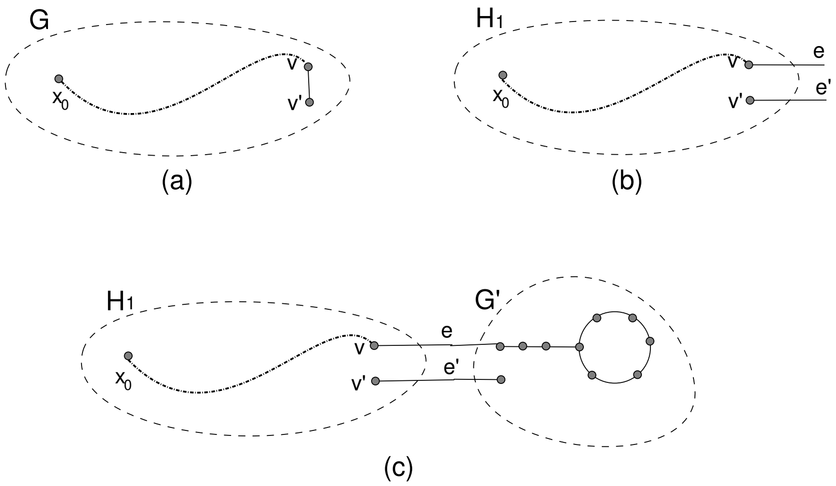

Fig. 8. Extending a graph at a single edge to trap a robot

is by cutting $\{v, v'\}$ to create two pending half-edges (Figure 8(b)); the node v is connected to a pending half-edge e and v' is connected to a pending half-edge e'. The resulting graph is H_1. A graph G' of $O(K)$ nodes is glued to e and e' (Figure 8(c)), such that $\widehat{A}$ does not traverse at least one of its edges. The resulting graph is called H. Actually, it turns out that the extension added to G is pretty simple: either adding a path connected to a cycle (based on $\bar{G}(\widehat{A})$) as illustrated in Figure 8(c), or connecting e and e' by (an appropriately labeled) path.

Consider the footprint of $\widehat{A}$, $fp(\widehat{A}) = W_0W^*$, $|W_0W| = K$, where p_i is the i-th letter in $fp(\widehat{A})$. Consider the sequence of configurations of $\widehat{A}$

$$(x_0, s_0) \to (x_1, s_1) \to \cdots$$

where $s_0 = \hat{s}$. Let $p_{i+1} = f_\ell(s_i)$. Assume $\widehat{A}$ traverses $\{v, v'\}$ for the first time at step i, $i \geq 1$; *i.e.*, when going from (x_{i-1}, s_{i-1}) to (x_i, s_i); assume w.l.o.g. it traverses it at this time from v to v', *i.e.*, $x_{i-1} = v$ and $x_i = v'$. Thus, p_i is the label of $\{v, v'\}$. In other words, if we cut the edge to obtain the two pending half-edges e, e', then $\widehat{A}$ traverses e.

We consider two cases depending on when $\widehat{A}$ traverses e, during the simple path of $G(\widehat{A})$ or during the cycle of $G(\widehat{A})$.

Case 1. Assume $\widehat{A}$ traverses e at step $i \leq |W_0|$. Thus, p_i belongs to W_0. In this case we can use the undirected graph of $\widehat{A}$, $\bar{G}(\widehat{A})$, and construct the version with no parallel edges and self-loops, $\bar{G}^s(\widehat{A})$, as in Lemma 5, adding at most 3 new nodes. We glue the part of $\bar{G}^s(\widehat{A})$ that starts after p_i to e. Namely, we connect to e a path of length $|W_0| - i$ whose extremity is denoted by w. The edges of this path are labeled $p_{i+1}, \ldots, p_{|W_0|}$. At w, we add the ring of $\bar{G}^s(\widehat{A})$. The other half-edge e' is completed into an edge by adding to it one new node, and the graph obtained is H. Notice that we added at most $K + 4$ new nodes.

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. □

By Claim 4.1 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), $|W''| > 2$. 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. The other half-edge e' is completed into an edge by adding to it one new node, and the graph obtained is H. Notice that we added at most $2K+1$ new nodes.

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, to obtain H. Thus, when $\widehat{A}$ 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. Notice that we added at most $2K$ new nodes.

Lemma 6. *The graph H is simple and edge-colored. Also, H is a trap core for $\widehat{A}$ when starting in x_0, with at most $2K+3$ nodes more than G.*

Proof. It follows directly from Lemma 5 that H is simple and edge-colored. The number of nodes of H is counted in the previous three cases.

The proof of Case 1 is as follows; the other cases are similar. Assume $\widehat{A}$ traverses e at step $i \leq |W_0|$. In this case $\widehat{A}$ does not traverse the edge e' of H. Observe that $\widehat{A}$ is trapped in the segment of $\bar{G}^s(\widehat{A})$ added to e. This follows because $\widehat{A}$ is in (x_{i-1}, s_{i-1}) before traversing e, and in (x_i, s_i) after traversing it. At this moment it is at the beginning of the segment of $\bar{G}^s(\widehat{A})$ added, so it will continue traversing this graph without trying to take an edge not defined, as in Lemma 5. □

4.2 Trapping a Team of Robots

With the results of the previous subsection we are ready to prove the main result of this section.

Theorem 4. *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 $O(qK)$ 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 9.

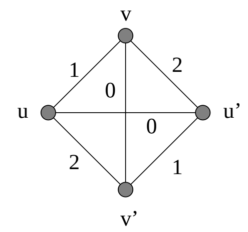

Fig. 9. Basic step of the induction

For the induction step, assume that Theorem 4 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 4 for $\mathcal{A}$.

Let $\widehat{A}$ be an irreducible robot obtained from A as in Lemma 4. 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 edge-colored. Let us place $\widehat{A}$ at node u of G_q, and perform the construction of the previous subsection, Lemma 6, to obtain a graph H. Then, as in Theorem 2, construct an homogeneous graph H_1 from Definition 3, and add the tower at any of the newly added edges, say $\{v, v'\}$ of height $K+1$ (see Figure 7), to obtain a graph H_2. The edge $\{v_1, v_1'\}$ in the gadget of the tower is not traversed by A when starting from u.

We repeat the same construction by considering the robot $\widehat{A}$ launched from u' in H_2. More precisely, we construct G_{q+1} from H_2 in the same way H_2 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. Also, any robot in $\mathcal{A}$ fails to traverse the edge $\{v_2, v_2'\}$ of G_{q+1} when starting from u or u'. This is because 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}. Starting from u, A fails to traverse the edge $\{v_1, v_1'\}$ of H_2. This edge being the one that is "opened" to construct G_{q+1} from H_2, 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_2, 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_2 when starting from u.

To complete the proof, it just remains to compute the size of G_{q+1}. We claim $|G_{q+1}| \leq |G_q| + 10K + O(1)$. We give simple upper bounds on the size of the intermediate graphs. First, we have $|H| \leq |G_q| + 2K + 3$ by Lemma 6. Then, we have $|H_1|$ has 13 more nodes at the most, as in Definition 3, so $|H_1| \leq |G_q| + 2K + 16$. The tower has $2K + 6$ nodes (proof of Theorem 2), so $|H_2| \leq |G_q| + 4K + 22$. The same procedure for the starting node u' contributes to another $4K + 22$ additional nodes. The result follows. Therefore, $|G_{q+1}| \leq 8qK + O(q)$, which completes the proof of the theorem. □

By simply rewriting Theorem 4, 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 [16]:

Corollary 2. *For any set of q non-cooperative K-state robots, there exists a trap of size $O(qK)$.*

Corollary 3. *A team of q non cooperative robots that explores all graphs of size n requires at least $\Omega(\log \frac{n}{q})$ memory bits per robot.*

By simply plugging this latter bound in the construction by Rollik [16] for team of locally-cooperative robots, we get:

Corollary 4. *For any set of q locally-cooperative K-state robots, there exists a trap of size $\tilde{O}(K^{K^{\cdot^{\cdot^{\cdot^{K}}}}})$, with $q+1$ levels of exponential.*

5 Bounds for Terminating Exploration

In this section, we consider the *terminating exploration* 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 [16]. On the other hand, it is known that a robot with unbounded memory can explore all graphs, using only one pebble [8]. 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 $A = (\Delta, \mathcal{S}, f, s_0)$, with a finite set of states $\mathcal{S}$, $s_0 \in \mathcal{S}$, and

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

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, $p(s) = 0$. Only if $p(s) = 1$ we allow f to be undefined; in such case we say s is a *stop* state. For the initial state s_0, $p(s_0) = 0$. 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.

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 performs terminating exploration of 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 succeed terminating exploration is called a *trap* for A.

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

Theorem 5. *For any K-state p-robot there exists a trap of size $O(K^3)$.*

Proof. Let $A = (\Delta, \mathcal{S}, f, s_0)$ 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 4 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 10).

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 11. The resulting graph is denoted by T.

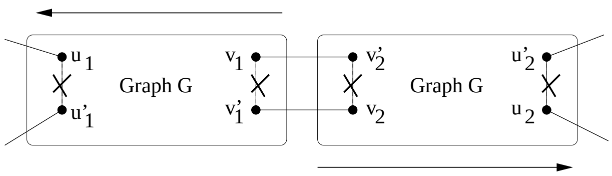

Fig. 10. The graph G_1

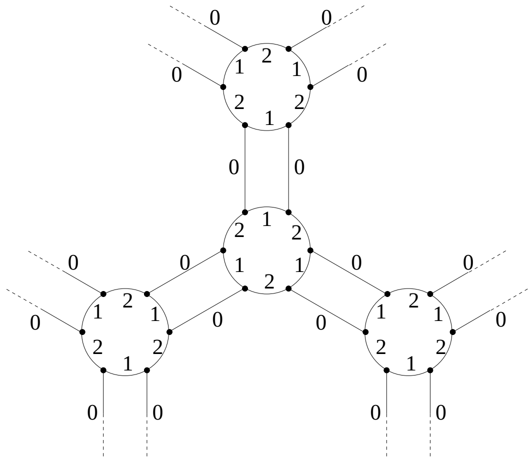

Fig. 11. The modified infinite tree T

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[6]. 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 [12] and in [16]. 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 [12], we identify the nodes w and w'. This leads to the

[6] Since the $\{u,u'\}$ edges are "open," the proof requires to consider the last time the p-robot is in a u node.

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. □

Corollary 5. *A robot that performs terminating exploration of all graphs of size n requires at least $\Omega(\log n)$ memory bits.*

Remark. This latter bound is tight, as proved in [13].

6 Conclusions

On the one hand, we have proved that terminating exploration (using one pebble) requires $\Omega(\log n)$ bits for the family of graphs with at most n nodes. On the other hand, we proved in [13] that there exists an terminating exploration algorithm using a robot with $O(D \log \Delta)$ bits of memory for the terminating exploration of all graphs of diameter at most D and degree at most Δ. The design of a tight bound for terminating exploration is still an open problem.

References

1. R. Aleliunas, R. Karp, R. Lipton, L. Lovasz and C. Rackoff Random walks, universal traversal sequences, and the time complexity of maze problems In Proc. of the 20th Annual Symposium on Foundations of Computer Science (FOCS), pages 218–223, 1979.
2. S. N. Bhatt, S. Even, D. S. Greenberg, R. Tayar. Traversing Directed Eulerian Mazes. *J. Graph Algorithms Appl.* **6**(2): 157–173, 2002.
3. 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.)
4. 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.
5. L. Budach. Automata and labyrinths. *Math. Nachrichten,* pages 195–282, 1978.
6. S. Cook and C. Rackoff. Space lower bounds for maze threadability on restricted machines. SIAM J. on Computing 9(3):636–652, 1980.
7. 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. (Full version to appear in J. of Algorithms.)
8. 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.
9. S. Even, G. Itkis, S. Rajsbaum. On Mixed Connectivity Certificates. *Theor. Comput. Sci.* **203**(2): 253–269, 1998.
10. S. Even, A. Litman, P. Winkler. Computing with Snakes in Directed Networks of Automata. *J. Algorithms* **24**(1): 158–170, 1997.
11. S. Even and R.E. Tarjan. Computing an st -Numbering. *Theor. Comput. Sci.* **2**(3): 339–344, 1976.

12. 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. (Full version to appear in Theoretical Computer Science.)
13. P. Fraigniaud, D. Ilcinkas, S. Rajsbaum, S. Tixeuil. Space Lower Bounds for Graph Exploration via Reduced Automata. In 12th International Colloquium on Structural Information and Communication Complexity (SIROCCO), Lecture Notes in Computer Science #3499, Springer, pp. 140–154, 2005.
14. M. Kouckỳ, Universal Traversal Sequences with Backtracking, Proc. 16th IEEE Conference on Computational Complexity (2001), 21-26. (Also, to appear in J. Computer and System Sciences.)
15. O. Reingold. Undirected ST-Connectivity in Log-Space. To appear in 37th ACM Symp. on Theory of Computing (STOC), 2005.
16. H.-A. Rollik. Automaten in planaren graphen. *Acta Informatica* **13**: 287–298, 1980.
17. C.-E. Shannon. Presentation of a Maze-Solving Machine. In 8th Conf. of the Josiah Macy Jr. Found. (Cybernetics), pages 173–180, 1951.