Impact of memory size on graph exploration capability

Pierre Fraigniaud*
CNRS
LIAFA, Univ. Denis Diderot, Paris, France
pierre.fraigniaud@liafa.jussieu.fr

David Ilcinkas*
CNRS
LaBRI, Univ. Bordeaux I, France
david.ilcinkas@labri.fr

Andrzej Pelc†
Dép. d'informatique
Univ. du Québec en Outaouais, Canada
pelc@uqo.ca

October 24, 2007

Abstract

A mobile agent (robot), modeled as a finite automaton, has to visit all nodes of a regular graph. How does the memory size of the agent (the number of states of the automaton) influence its exploration capability? In particular, does every increase of the memory size enable an agent to explore more graphs? We give a partial answer to this problem by showing that a strict gain of the exploration power can be obtained by a polynomial increase of the number of states. We also show that, for automata with few states, the increase of memory by even one state results in the capability of exploring more graphs.

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

*This work was done while the authors were with University Paris-Sud (LRI). Additional support by the project "PairAPair" of the ACI Masses de Données, and by the project "Grand Large" of INRIA.

†Supported in part by NSERC discovery grant and by the Research Chair in Distributed Computing of the Université du Québec en Outaouais.

1 Introduction

1.1 The problem

A mobile agent (robot), has to *explore* an undirected graph by visiting all its nodes, without any a priori knowledge of the topology of the graph or of its size. This task is fundamental, e.g., in searching for data stored at unknown nodes of a network. More precisely, we consider the task of "perpetual" exploration in which the robot has to visit all nodes of the graph but is not required to stop. That is, the robot moves from node to node, traversing edges, so that eventually all nodes have been visited.

If nodes and edges have unique labels, exploration can be easily achieved, e.g., by depth-first search. However, if the robot operates in an unknown environment, such unique labeling may not be available. Hence it is important to be able to program the robot to explore *anonymous* graphs, i.e., graphs without unique labeling of nodes or edges. Clearly, the robot has to be able to locally distinguish ports at a node: otherwise it is impossible to explore even the star with 3 leaves (after visiting the second leaf, the robot cannot distinguish the port leading to the first visited leaf from that leading to the unvisited one). Hence we make a natural assumption that all ports at a node are locally labeled $0, \ldots, d-1$, where d is the degree of the node. No consistency between those local labelings is assumed.

In many applications the memory size of the robot has to be kept small, due to cost and size constraints. Hence it is important to determine the exploration capabilities of a robot with restricted memory size. We model a robot as a finite automaton and measure its memory by the number of states. A given robot is capable of exploring a graph G if it can visit all its nodes regardless of the port labeling and of the starting node. (Note that if the port labeling can be chosen, even a memoryless robot, i.e., an automaton with one state, can visit all nodes by always exiting by port $i+1$ after entering by port i: the labeling simply has to guide the robot along an Eulerian tour of the respective bidirectional oriented graph.)

In this paper we restrict attention to exploration of regular graphs. These graphs are usually harder to explore by an automaton with small memory because decisions where to go next cannot be made on the basis of the degrees of visited nodes, and must be based only on entry port numbers. For example, it is easy to see that *every* graph can be augmented by adding a few high-degree nodes on some edges, so that the resulting (non-regular) graph can be explored by a memoryless robot (i.e., an automaton with one state). These high degree nodes are used as flags guiding the robot in DFS traversal of a spanning tree. The non-tree edges are marked by inserting middle nodes of high degree. By contrast we show that the only *regular* graphs that can be explored by a memoryless robot are cycles and 1- and 2-node graphs.

The central question of this paper is:

> How does the memory size of the robot (the number of states of the automaton) influence its exploration capability? In particular, does every increase of the memory size enable the robot to explore more graphs?

We conjecture that the answer to the latter question is yes and our results give a partial support to this claim. In order to state the conjecture and our results precisely, we need to introduce appropriate terminology and fix the model.

1.2 Terminology and model

We consider only undirected regular connected graphs without multiple edges and self-loops. An anonymous graph with locally labeled ports is a regular graph of degree d whose nodes are unlabeled and where the edges incident to a node v have distinct labels $0, \ldots, d-1$. Thus every undirected edge $\{u, v\}$ has two labels which are called its *port numbers* at u and at v. Port numbering is *local*, i.e., there is no relation between port numbers at u and at v. All arithmetic operations on port numbers are done modulo d.

We are given a mobile entity (robot) traveling in a regular anonymous graph of degree d with locally labeled ports. The graph and its size are a priori unknown to the robot, which only knows the degree d. Since the graph is regular and anonymous, the only input obtained by the robot at every stage of the exploration is the port number by which it enters a node. More precisely, a k-state robot is a finite Mealy automaton $\mathcal{R} = (X, Y, \mathcal{S}, \delta, \lambda, S_0)$ where $X = Y = \{0, \ldots, d-1\}$ (X and Y are the input and the output alphabets, respectively), $\mathcal{S}$ is a set of k states among which there is a specified state S_0 called the *initial* state, $\delta : \mathcal{S} \times X \to \mathcal{S}$ is the state transition function, and $\lambda : \mathcal{S} \times X \to Y$ is the output function. Initially the robot comes to some node u_0 by a port i_0 in the initial state $S_0 \in \mathcal{S}$. The pair (S_0, i_0) determines a local port number $i_1 = \lambda(S_0, i_0) \in Y$, by which the robot leaves u_0, and a new state $S_1 = \delta(S_0, i_0)$. When incoming to a node v in some state S, the behavior of the robot is as follows. It reads the number $i \in X$ of the port through which it entered v. The pair (S, i) causes the transition from state S to the state $S' = \delta(S, i)$ and yields a local port number $j = \lambda(S, i)$, by which the robot leaves v. The robot continues moving in this way, possibly infinitely.

Further in the paper we will use the following observation.

Observation 1.1 *If there exist a state S and port number i such that $\delta(S, i) = S$ and $\lambda(S, i) = i$ then a robot entering an edge whose both port numbers are i, in state S, must move along this edge back and forth forever.*

As mentioned before, we consider the task of "perpetual" exploration in which the robot has to visit all nodes of the graph but is not required to stop. That is, it is not required that a final state be in $\mathcal{S}$. A robot is said to perform an *exploration* of a graph G with fixed port labeling starting at (u_0, i_0) if, when entering an initial node u_0 of G by the initial port i_0 in the initial state S_0, it visits all nodes of G in finitely many steps.

A regular graph G of degree d is *edge-colored* if every edge of G is given a color, every two incident edges have different colors, and there are d colors used in total. There is a clear correspondence between edge-colored graphs and regular graphs in which the labels at the two extremities of each edge are identical. Notice that in an edge-colored graph the automaton does not learn any new information when coming to a new node because the port number by which it enters is the same as the port number by which it left the previous node. Hence in edge-colored graphs both the trajectory of the robot and its sequence of states depend exclusively on the pair (S_0, i_0).

Since we are interested in exploring graphs regardless of their port labeling and starting node and port number, we introduce the following notion.

Definition 1.1 *Given any positive integer k, the* class of k-explorable graphs *is the set $\mathcal{G}_k$ of regular graphs G such that there exists a k-state automaton $A(G)$ which visits all nodes of G for any port labeling of G, any starting node u_0, and any initial port number i_0.*

Clearly, the sequence of sets $\mathcal{G}_k$ is non-decreasing, i.e., $\mathcal{G}_k \subseteq \mathcal{G}_{k+1}$ for every $k > 0$.

1.3 The conjecture and our results

Using the notion of classes $\mathcal{G}_k$ of k-explorable graphs, our conjecture that every increase of the memory size enables the robot to explore more graphs can be concisely and formally stated as follows.

Conjecture. For every $k > 0$, $\mathcal{G}_k$ is strictly included in $\mathcal{G}_{k+1}$.

Our results give a partial support to this conjecture. We prove that a polynomial increase of the number of states increases exploration capabilities of a robot. More precisely, we show that there is a polynomial function $f : \mathbb{N} \to \mathbb{N}$, such that $\mathcal{G}_k$ is strictly included in $\mathcal{G}_{f(k)}$, for any $k > 0$.

The previous knowledge on this subject was the following. It follows from [31] that the hierarchy $\mathcal{G}_k$ is unbounded, in the sense that there are graphs outside of any $\mathcal{G}_k$. On the other hand, any graph can be explored by a sufficiently large automaton, hence [31] implies that there exists a function $g : \mathbb{N} \to \mathbb{N}$, such that $\mathcal{G}_k$ is strictly included in $\mathcal{G}_{g(k)}$. However, this function g obtained from [31] is exponential. Hence the result from [31] guarantees only that an exponential increase of the number of states allows a robot to explore more graphs, while we show that a polynomial increase is sufficient.

On the other hand, we show that for a small number of states the increase of memory by even one state results in the capability of exploring new graphs, i.e., that the beginning of the hierarchy $\mathcal{G}_k$ is indeed strictly increasing. More precisely, we show that $\mathcal{G}_k$ is strictly included in $\mathcal{G}_{k+1}$, for $k = 1, 2$. Moreover we provide an exact characterization of classes $\mathcal{G}_1$ and $\mathcal{G}_2$. The class $\mathcal{G}_1$ of regular graphs that can be explored by a memoryless automaton (i.e., an automaton with 1 state) consists of cycles and of the 1- and 2-node cliques, while the class $\mathcal{G}_2$ of regular graphs that can be explored by a 2-state automaton consists of all cycles and all cliques.

1.4 Related work

Exploration and navigation problems for robots in an unknown environment have been extensively studied in the literature (cf. [20, 30]). There are two groups of models for these problems. In one of them a particular geometric setting is assumed. Another approach is to model the environment as a graph, assuming that the robot may only move along its edges. The graph setting can be further specified in two different ways. In [1, 6, 7, 14, 19] the robot explores strongly connected directed graphs and it can move only in the head-to-tail direction of an edge, not vice-versa. In [3, 10, 12, 15, 16, 17, 29, 31] the explored graph is undirected and the robot can traverse edges in both directions. Graph exploration scenarios considered in the literature differ in an important way: it is either assumed that nodes of the graph have unique labels which the robot can recognize (as in, e.g., [14, 17, 29]), or it is assumed that nodes are anonymous (as in, e.g., [6, 7, 12, 31]). We are concerned with the latter context. The efficiency measure adopted in papers dealing with graph exploration is either the completion time of this task, measured by the number of edge traversals, (cf., e.g., [29]), or the memory size of the robot, measured either in bits or by the number of states of the finite automaton modeling the robot (cf., e.g., [15, 18, 19]). Time is not an issue in our approach, and we address the latter efficiency measure, i.e., memory space. Three versions of the exploration problem have been addressed in the literature: exploration with return (in which the robot has to perform exploration and return to its starting position), exploration with stop (in

which the robot has to complete exploration and eventually stop), and perpetual exploration (the type of exploration considered in this paper). For instance, it is shown in [15] that exploration with stop of n-node trees requires a robot with memory size $\Omega(\log\log\log n)$ bits, and that exploration with return of n-node trees can be achieved by a robot with $O(\log^2 n)$ memory bits.

The capability of a robot to explore anonymous undirected graphs has been addressed in, e.g., [8, 12, 15, 18, 27, 31]. In particular, it was shown in [31] that no finite automaton can explore all cubic planar graphs (in fact no finite set of finite automata can cooperatively perform this task). The size of port-labeled graphs which cannot be explored by a given robot was investigated in [18].

Our work has connections with derandomized random walks. There, the objective is to produce an explicit universal traversal sequence (UTS), i.e., a sequence of port labels, such that the path guided by this sequence visits all edges of any graph. It has been proved in [24] that, with high probability, a sequence of length $O(n^3 d^2 \log n)$, chosen uniformly at random, guides a walk in any d-regular (connected) graph of n nodes. Explicit UTS constructions are known for 2-regular graphs (cf. [5, 11, 13, 23, 26]), for 3-regular graphs (cf. [4, 22, 28]), for cliques (cf. [2, 25]), and for expanders (cf. [21]). Some of these sequences can be constructed in log-space, and hence can produce perpetual exploration with compact memory. However, even if bounds on the length of these sequences have been derived, they provide little knowledge on the minimum number of states for graph exploration by a robot. For instance, sequences of length $\Omega(n^{1.43})$ are required to traverse all cycles with n nodes [13], although a 1-state robot can accomplish this task.

2 Polynomial memory increase

In this section we prove that a polynomial increase of the number of states of a robot results in augmenting the capability to explore graphs. More precisely, the main result of this section is the following.

Theorem 2.1 *There exists a polynomial function $f : \mathbb{N} \to \mathbb{N}$, such that $\mathcal{G}_k$ is strictly included in $\mathcal{G}_{f(k)}$, for any $k > 0$.*

We will use the notion of pseudo-palindrome, which was defined in [18]. We recall here the main definitions and properties.

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

A sequence L' is a *reduction* of L if $L' = A \circ B$ and $L = A \circ L'' \circ B$ where L'' is a nonempty pseudo-palindrome, and A and B are two arbitrary sequences (possibly empty). A sequence is said *pp-free* if it has no reduction. A sequence L' is the *pp-reduction* of a sequence L if L' is pp-free and obtained from L by successive reductions. One can easily check that the pp-reduction of a finite sequence is unique (cf., e.g., Section 1.7 in [12]). For instance the pp-reduction of 11221211213223311313323112 21 is 1231. Given any regular edge-colored graph $G = (V, E)$ and any node $u \in V$, each sequence L of edge labels defines a path P from u in G. By induction of the length of L, the following observation holds.

Observation 2.1 *If L is a pseudo-palindrome then P starts and ends at u.*

An infinite sequence L is called *periodic* if $L = A \circ B^*$, where A and B are finite, B non-empty. The length of the shortest B for which L can be represented in this way is called the *period* of L. By induction of the period of L we have the following observation.

Observation 2.2 *If L is an infinite sequence with period p, then the pp-reduction L' of L is either finite or is also periodic with period $p' \leq p$.*

The main tool to prove Theorem 2.1 is the following lemma, which may be of independent interest.

Lemma 2.1 *For any $k > 0$ there exists a regular planar graph G of degree 3, with $O(k^3)$ nodes, such that $G \notin \mathcal{G}_k$.*

Proof. For $k = 1$ the lemma follows from Propositions 3.1 and 3.2 in the next section. Fix an integer $k > 1$.

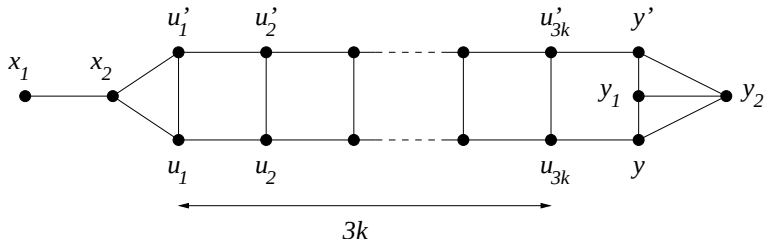

Figure 1: The subgraph T

We first define a graph T that is used as a subgraph of G (cf. Figure 1). The graph T has $6k+6$ nodes $x_1, x_2, y, y', y_1, y_2, u_1, \ldots, u_{3k}, u'_1, \ldots, u'_{3k}$ and $9k+7$ edges $\{x_1, x_2\}$, $\{x_2, u_1\}$, $\{x_2, u'_1\}$, $\{y_1, y_2\}$, $\{y, y_1\}$, $\{y, y_2\}$, $\{y', y_1\}$, $\{y', y_2\}$, $\{y', u_{3k}\}$, $\{y', u'_{3k}\}$, $\{u_{3k}, u'_{3k}\}$ and $\{u_i, u'_i\}$, $\{u_i, u_{i+1}\}$, $\{u'_i, u'_{i+1}\}$ for all $1 \leq i < 3k$. Thus all nodes have degree 3 except x_1 that has degree 1. The subgraph of T induced by the nodes y, y', y_1, y_2 is called the *tail* of the graph T.

Let $\widehat{G}$ be a chain of disjoint cycles of all lengths from 3 to $3k$. More precisely, let v_n and v'_n be two different nodes of the cycle of length n, for $3 \leq n \leq 3k$. Connect the cycles by the edges $\{v'_n, v_{n+1}\}$, for $3 \leq n < 3k$. Now attach a graph T to each node w of $\widehat{G}$ of degree 2 by identifying w of $\widehat{G}$ and x_1 of T. Let G denote the resulting graph. It is regular of degree 3. Since T has $O(k)$ nodes, the graph G has $O(k^3)$ nodes.

It remains to prove that $G \notin \mathcal{G}_k$. Fix an automaton A with $k > 1$ states. Our goal is to find a port-labeling of G, and a starting node and port number, such that the automaton A does not visit all nodes of G.

If an automaton traverses only colored edges in a regular graph (even if ports of non traversed edges are different at both extremities of the edge), then both the trajectory of the automaton (the sequence of port numbers) and its sequence of states depend exclusively on the pair (S_0, i_0), where S_0 is the initial state and i_0 the initial port number. In particular, after at most $3k+1$ steps, the automaton will enter a node in a state S by a port i, such that the pair (S, i) already occurred at some previous node. Thus, after this step, both the sequence of states and the sequence of edge colors $L = \{a_n\}_{n \geq 0}$ are periodic. There exist q and p, with $q + p \leq 3k$, such that for all $i \geq q$, $a_i = a_{i+p}$. Let $L' = b_1, b_2, \ldots$ be the pp-reduction of the sequence L. We have $i_1 < i_2 < \ldots$

such that $b_r = a_{i_r}$. We use a specific port-labeling which is partially an edge-coloring. The given automaton will only traverse colored edges, even though the graph is not completely edge-colored.

By Observation 2.2, we consider two cases.

First assume that the sequence L' is finite. Then the pp-reduction of the sequence $a_{q+1}, \ldots, a_{q+p}$ is a palindrome and hence the sequence $a_{q+1}, \ldots, a_{q+2p}$ is a pseudo-palindrome. Choose some node w in the cycle of size 3 of $\widehat{G}$. Choose an arbitrary proper 3-coloring of the three edges of the cycle. Extend the proper 3-coloring to the rest of the graph except for the tail of each T. (This can be easily done, while proper edge 3-coloring of the entire graph, even of the subgraph T, is impossible). Label arbitrarily the remaining ports of the graph (those in the tails of copies of T) by integers 0,1,2 at each node. It remains to compute the starting node of the automaton. From node w, traverse edges colored $a_q, a_{q-1}, \ldots, a_1$. This traversal leads to a node u_0. We claim that the automaton, starting at u_0 with the initial port i_0 never visits any tail of a copy of T and thus fails to explore G. During the first q steps, the automaton is at distance at most $3k$ from w because $q \leq 3k$. At step $q + 2rp$, for $r \geq 0$, the automaton is always at w by Observation 2.1, because $a_q, a_{q+1}, \ldots, a_{q+2p}$ is a pseudo-palindrome. Since $p \leq 3k$, the automaton is again at distance at most $3k$ from w. Since w is on one of the $3k-2$ cycles, the automaton never visits any tail of a copy of T, thus it never traverses non-colored edges. Hence A fails to explore G.

Assume from now on that L' is infinite. By Observation 2.2 L' inherits the property of periodicity: there exist $q' \leq q$ and $p' \leq 3k$ such that $b_r = b_{r+p'}$ for all $r \geq q'$. If the period is 2, we choose $p' = 4$ to avoid double edges (a cycle of length 2). Starting at a node u, label the edges of the cycle of size p' in G with colors $b_{q'+1}, b_{q'+2}, \ldots, b_{q'+p'}$. Extend the coloring to the rest of the graph except for the tail of each copy of T. Label arbitrarily the remaining ports of the graph by integers 0,1,2 at each node. It remains to compute the starting node of the automaton. From node u, traverse edges colored $b_{q'}, b_{q'-1}, \ldots, b_1$. This traversal leads to a node u_0. We claim that the automaton, starting in u_0 with the initial port i_0 never visits any tail of a copy of T and thus fails to explore G. During the first q steps, the automaton is at distance at most $3k$ from u because $q \leq 3k$. At step $i_{q'}$ the automaton is in u and then it comes back to u every $2p$ steps because u is on the cycle of length p', which is labeled to trap the automaton. Since $p \leq 3k$, the automaton is again at distance at most $3k$ from u. Since u is on one of the $3k-2$ cycles, the automaton never explores any tail of a copy T, thus it fails to explore G.

For every automaton with k states we showed a port labeling of G, an initial node u_0 and a port i_0 such that this automaton fails to explore G. Hence $G \notin \mathcal{G}_k$. □

Proof of Theorem 2.1. Now the proof of the theorem can be concluded as follows. For any k take the graph G with $n \in O(k^3)$ nodes whose existence is guaranteed by Lemma 2.1. Hence $G \notin \mathcal{G}_k$. Take a Universal Traversal Sequence of length $O(n^3 \log n)$ for all regular graphs of degree 3 and of size n. This sequence can be coded by an automaton with $O(n^3 \log n)$ states. Hence there exists a polynomial $f(k)$, such that an automaton with $f(k)$ states explores graph G regardless of the port labeling and of the starting node and port. This implies $G \in \mathcal{G}_{f(k)}$. □

3 Exploration power of small automata

In this section we show that our conjecture concerning the hierarchy $\mathcal{G}_k$ holds for $k = 1, 2$, i.e., $\mathcal{G}_1$ is strictly included in $\mathcal{G}_2$, which is strictly included in $\mathcal{G}_3$. To this end we determine exactly elements

of $\mathcal{G}_1$ and of $\mathcal{G}_2$.

Proposition 3.1 *The class $\mathcal{G}_1$ of regular graphs that can be explored by a memoryless automaton (an automaton with one state) consists of the single-node graph, the two-node clique, and all cycles.*

Proof. The fact that the one- and two-node cliques and all cycles can be explored by a 1-state automaton is straightforward (for cycles the output function is $\lambda(S_0, i) = i+1$). Suppose that some other regular graph G belongs to $\mathcal{G}_1$. Since the only regular trees are the single-node graph and the two-node clique, the graph G must contain a simple cycle C. Since G itself is not a cycle, there are two cases: either there are nodes of G not in C or G contains a chord of C. In the latter case G contains a shorter cycle C' containing this chord. It follows that in the second case there are nodes of G not in C'. Hence, in any case, G contains a cycle C^* and a node v not belonging to it. Let $\lambda : \{S_0\} \times X \to Y$ be the output function of an automaton with a single state S_0 exploring G. If $\lambda(S_0, i) = i$, for all i, then, by Observation 1.1, it does not explore G. Otherwise, let i be such that $\lambda(S_0, i) \neq i$. Label all ports on the cycle C^* such that every edge of the cycle C^* is labeled i at one of its extremity and $\lambda(S_0, i)$ at the other extremity. For this labeling, the automaton entering any node of the cycle by the initial port i perpetually explores only C^* without ever visiting v. Hence it does not explore G. This contradiction implies that G cannot belong to $\mathcal{G}_1$. □

In order to characterize the class $\mathcal{G}_2$ we will need some graph theoretical facts. By a simple cycle we mean a cycle with non-repeating nodes. The following result is well known.

Lemma 3.1 *Every regular graph of degree larger than 2 contains a simple cycle of even length.*

Lemma 3.2 *Every regular graph of degree larger than 2 and different from the 4-node clique contains a non-hamiltonian simple cycle of even length.*

Proof. Take the simple cycle C of even length whose existence is guaranteed by Lemma 3.1. If it is not hamiltonian, we are done, so suppose that C is a hamiltonian cycle. First suppose that the graph is of degree at least 4. Take any node v of C. There must exist at least two chords of C with endpoint v, say $\{v, x\}$ and $\{v, y\}$. Consider the cycle C' composed of the chord $\{v, y\}$ and the part of C containing x (cf. Fig. 2 (a)). If the length of C' is even, this is a non-hamiltonian cycle of even length. Otherwise, one of the following cycles must have this property: either the cycle composed of $\{v, x\}$ and of the part of C not containing y, or the cycle containing $\{v, x\}$ and $\{v, y\}$ and the part of C between x and y not containing v (the sum of lengths of these cycles is of the same parity as the length of C').

It remains to consider the case when the degree of G is 3. Consider two adjacent nodes v and w on C (cf. Fig. 2 (b) and (c)). Each of these nodes is an endpoint of a chord, $\{v, x\}$ and $\{w, y\}$, respectively, with $x \neq y$. If these chords are not intersecting (when the cycle C is represented as a circle in the plane, cf. Fig. 2 (b)), there are two cases. If part a or part c of the cycle is of even size then the cycle (a, v, x) (resp. (c, y, w)) is as required. If both a and c are of odd size then part b must be of even size (because C is of even length) and the cycle (v, w, y, b, x) is as required. Hence suppose that the chords $\{v, x\}$ and $\{v, y\}$ intersect (cf. Fig. 2 (c)). Consider two cases.

Case 1. Part b is of even size. If either a or c is nonempty then the cycle (v, x, b, y, w) is non-hamiltonian. It is of even size, hence we are done. Otherwise b must be nonempty because G is not the 4-node clique. Hence the cycle (v, x, w, y) is a 4-node non-hamiltonian cycle.

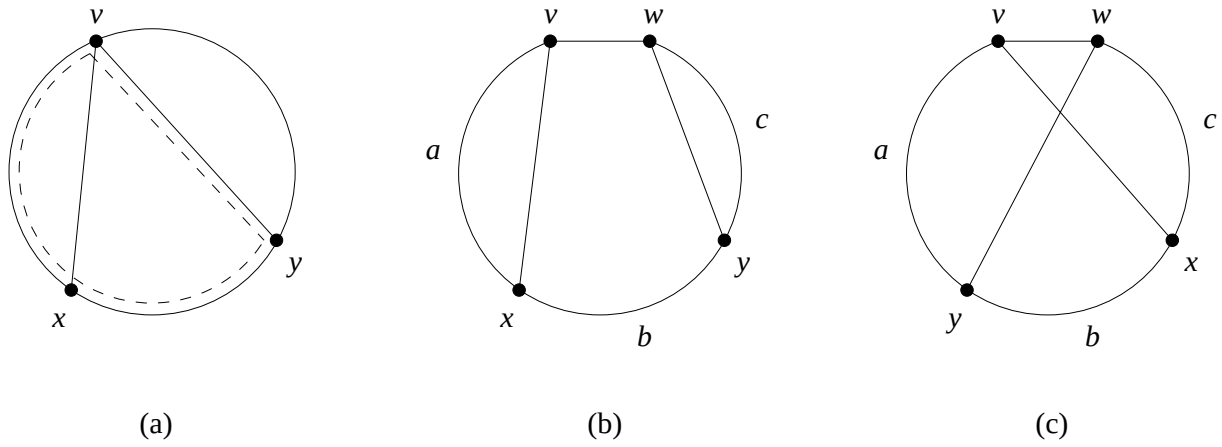

Figure 2: (a) degree at least 4, (b) and (c) Chords in the cycle C

Case 2. Part b is of odd size. Either part a or part c must be of even size because C is of even length. Without loss of generality suppose that a is of even size. Then the cycle (v, a, y, b, x) is a non-hamiltonian cycle of even length. □

Lemma 3.3 *Every n-node regular graph of degree larger than 2 and different from the 4-node clique contains a simple cycle of length at most $n-2$.*

Proof. Consider an n-node regular graph of degree larger than 2 and different from the 4-node clique. By Lemma 3.2 it contains a simple non-hamiltonian cycle C of even length. If there are at least two nodes outside of C, we are done. Hence assume that there exists exactly one node v outside of C. If C has a chord in G then both shorter cycles containing this chord have the required property. Hence we assume that C does not have chords. Since the degree of G is at least 3, all nodes of C must be adjacent to v. Hence v is of degree $n-1$. Since G is regular, it must be a clique. Since it is not the 4-node clique, it must be a clique with at least 5 nodes. Hence it contains a cycle of length at most $n-2$. □

Our next result characterizes the class $\mathcal{G}_2$.

Proposition 3.2 *The class $\mathcal{G}_2$ of regular graphs that can be explored by a 2-state automaton consists of all cliques and of all cycles.*

Proof. Since all cycles belong to class $\mathcal{G}_1$, they also belong to class $\mathcal{G}_2$. Consider any clique and the 2-state automaton with states S_0 and S_1 and the following transition and output functions: for all i,

$$\delta(S_0, i) = S_1$$
$$\delta(S_1, i) = S_0$$
$$\lambda(S_0, i) = i + 1$$
$$\lambda(S_1, i) = i$$

It is easy to verify that the above automaton, starting at state S_0 at any node v of a clique, with any initial port, traverses the star centered at v and hence explores the clique.

It remains to prove that no regular graphs other than cycles and cliques belong to class $\mathcal{G}_2$. Suppose that such a graph G exists and let A be a 2-state automaton that explores G. Let S_0 and S_1 be the states of A, where S_0 is the initial state. Define, for every i,

$$\delta_0(i) = \delta(S_0, i)$$
$$\delta_1(i) = \delta(S_1, i)$$
$$\lambda_0(i) = \lambda(S_0, i)$$
$$\lambda_1(i) = \lambda(S_1, i)$$

where δ and λ are the transition and output functions of A.

First suppose that $\delta_0(i) = S_0$ for some port number i. If $\lambda_0(i) = i$ then, by Observation 1.1, the automaton does not explore G. If $\lambda_0(i) \neq i$ then let C be a non-hamiltonian cycle in G (cf. Lemma 3.1). Label all ports on the cycle C by the sequence $(i, \lambda_0(i), i, \lambda_0(i), \ldots)$. For this labeling, the automaton entering any node of the cycle by the initial port i in state S_0 perpetually explores only C and thus fails to explore G.

Hence from now on we assume that $\delta_0(i) = S_1$ for all i.

Next suppose that $\lambda_0(i) = i$, for all i.

If $\delta_1(j) = S_0$, for all port numbers j, then let u be an arbitrary node in G. Let $v_0, v_1, \ldots, v_{d-1}$ be the d neighbors of u. If the automaton enters a node v_k by port i in state S_0, then it backtracks to u in state S_1. If the automaton enters node u in state S_1, then it leaves u in state S_0 moving to some v_s. Thus, for this labeling, the automaton entering node v_0 by the initial port i leading to u explores only u and its neighbors. Since G is not a clique, the automaton fails to explore G.

Otherwise, let j be the port number for which $\delta_1(j) = S_1$. If $\lambda_1(j) = j$ then, by Observation 1.1, the automaton does not explore G. If $\lambda_1(j) \neq j$ then let C be a non-hamiltonian cycle in G. Label all ports on the cycle C by the sequence $(j, \lambda_1(j), j, \lambda_1(j), \ldots)$. For this labeling, the automaton entering any node of the cycle by the initial port $\lambda_1(j)$ in state S_0 perpetually explores only C and thus fails to explore G.

Hence in the rest of the proof we assume that $\delta_0(i) = S_1$ for all i, and that λ_0 is not the identity function. We consider three cases.

Case 1: $\delta_1(j) = S_0$ and $\lambda_1(j) = j$ for some port number j. Let u be an arbitrary node in G. Let $v_0, v_1, \ldots, v_{d-1}$ be the d neighbors of u. At every v_k, give the port number j to the edge leading to u. If the automaton enters a node v_k by port j in state S_1, then it backtracks to u in state S_0. If the automaton enters node u in state S_0, then it leaves u in state S_1 moving to some v_s. Thus, for this labeling, the automaton entering node u by any initial port explores only u and its neighbors. Since G is not a clique, the automaton fails to explore G.

Case 2: $\delta_1(j) = S_0$ and $\lambda_1(j) \neq j$ for some port number j. Let i be a port number such that $\lambda_0(i) \neq i$. Let C be a non-hamiltonian cycle of even length in G, existing in view of Lemma 3.2. Label all ports on the cycle C by the sequence $(i, \lambda_0(i), j, \lambda_1(j), i, \lambda_0(i), j, \lambda_1(j), \ldots)$. More precisely, if the cycle C is $v_1, v_2, \ldots, v_{2k}, v_1$, then an edge $\{v_{2l}, v_{2l+1}\}$ is labeled $\lambda_0(j)$ at v_{2l} and i at v_{2l+1}. The edge $\{v_{2l+1}, v_{2l+2}\}$ is labeled $\lambda_0(i)$ at v_{2l+1} and j at v_{2l+2}. For this labeling, the automaton entering node v_1 in state S_0 by the initial port i perpetually explores only C and thus fails to explore G.

Case 3: $\delta_1(j) = S_1$ for all port numbers j. There are two subcases.

Case 3.a: $\lambda_0(\{0, \ldots, d-1\}) \cap \lambda_1(\{0, \ldots, d-1\}) = \emptyset$
Hence λ_1 is not bijective and therefore not injective. Thus there are two different port numbers

j and j' such that $\lambda_1(j) = \lambda_1(j') = k$. At least one of j and j' is different from k. Without loss of generality assume $j' \neq k$. If $j \neq k$, let C be a cycle of size at most $n-2$ in G, existing in view of Lemma 3.3. Label all ports on the cycle C by the sequence $(j, k, j, k, \ldots)$. If $j = k$, let C be an edge of G. Label both port numbers of this edge by k. Let v be a node of C adjacent to a node u outside C. Such a node exists since G is connected. Label ports of edge $e = \{u, v\}$ by j' at v and by $\lambda_0(i)$ at u, for some port number i. For this labeling, the automaton entering node u by the initial port i in state S_0 moves to v and perpetually explores C. Hence the automaton visits u and all nodes of C. Since the size of C is at most $n-2$, the automaton fails to explore G.

Case 3.b: $\lambda_0(\{0, \ldots, d-1\}) \cap \lambda_1(\{0, \ldots, d-1\}) \neq \emptyset$
Hence there exist some port numbers i and j such that $\lambda_0(i) = \lambda_1(j) = k$. If $k = j$ then consider an edge $e = \{u, v\}$ with both ports numbered j. By Observation 1.1, the automaton entering u by the initial port i in state S_0 visits only u and v. If $k \neq j$ then let C be a non-hamiltonian cycle in G. Label all ports in the cycle C by the sequence $(j, k, j, k, \ldots)$. For this labeling, the automaton entering any node of the cycle by the initial port i perpetually explores only C and thus fails to explore G.

Hence for every regular graph G which is not a cycle or a clique and for any 2-state automaton A we showed a port labeling of G for which A does not visit all nodes of G. Thus $G \notin \mathcal{G}_2$. □

Propositions 3.1 and 3.2 imply that the class $\mathcal{G}_1$ is strictly included in $\mathcal{G}_2$. In order to prove the strict inclusion of $\mathcal{G}_2$ in $\mathcal{G}_3$, it is enough to describe a graph in $\mathcal{G}_3$ which is neither a cycle nor a clique.

Proposition 3.3 *The class $\mathcal{G}_3$ contains a graph which is neither a cycle nor a clique.*

Proof. Consider the 6-node graph G consisting of two disjoint cycles of length 3 whose nodes are connected by edges forming a perfect matching (cf. Fig. 3). We will show that G can be explored by a 3-state automaton. More precisely we show that this can be done by the following automaton A:

- Set of states $\{U, D, B\}$, where D is the initial state;
- Transition function $\delta(D, i) = B$, $\delta(B, i) = U$, $\delta(U, i) = D$, for any i;
- Output function $\lambda(D, i) = i + 1$, $\lambda(B, i) = i$, $\lambda(U, i) = i + 1$, for any i.

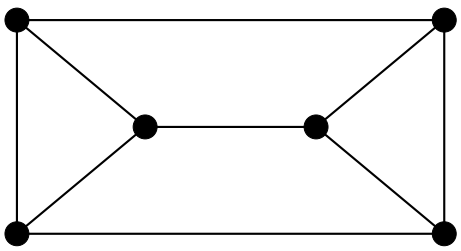

Figure 3: A graph in $\mathcal{G}_3 \setminus \mathcal{G}_2$

First notice that if A enters a node u in state D during exploration (excluding the first step), then the automaton visits node u, and each of its three neighbors, at least once, for any port assignment. One of u's neighbor is visited just before entering u in state D, and the two other neighbors are visited during the three next steps.

Let us consider the behavior of A in G. At step 4, A enters some node u in state D. After three more steps, A enters another node v in state D. After again three more steps, A enters a node w in state D. Node w is different from v. It is also different from u because edges $\{u, v\}$ and $\{v, w\}$ are different, and there are no multiple edges in G. Therefore, three different nodes of G are entered by A in state D during exploration. These three nodes and all of their neighbors are thus visited at least once by the automaton. Since any set of three nodes of G is a dominating set, all nodes of G are visited by A, for any port labeling and for any starting position. □

Propositions 3.1, 3.2 and 3.3 imply the following corollary.

Corollary 3.1 *Both inclusions $\mathcal{G}_1 \subset \mathcal{G}_2 \subset \mathcal{G}_3$ are strict.*

4 Conclusion

We showed that a polynomial increase of the number of states of a robot results in augmenting the capability to explore graphs. The main open problem is to prove or disprove our conjecture which says that *any* increase of the number of states of a robot enables to explore more graphs, i.e., that the hierarchy $\mathcal{G}_k$ is strict. Another interesting question is to show how complex it is to decide whether a given graph G can be explored by a k-state automaton. More precisely, can the decision problem "$G \in \mathcal{G}_k$?", for a given input (G, k), be decided in polynomial time?

References

[1] S. Albers and M. R. Henzinger. Exploring unknown environments. SIAM J. Computing 29:1164-1188, 2000.

[2] N. Alon, Y. Azar, and Y. Ravid. Universal sequences for complete graphs. Discrete Applied Mathematics 27:25-28, 1990.

[3] B. Awerbuch, M. Betke, R. Rivest and M. Singh. Piecemeal graph learning by a mobile robot. Information and Computation 152(2):155-172, 1999.

[4] L. Babai, N. Nisan, and M.Szegedy. Multiparty protocols and logspace-hard pseudorandom sequences. J. Computer and System Sciences 45(2):204-232, 1992.

[5] A. Bar-Noy, A. Borodin, M. Karchmer, N. Linial, and M.Werman. Bounds on universal sequences. SIAM J. Computing 18(2):268-277, 1989.

[6] 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):1-21, 2002.

[7] 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.

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

[9] M. Blum and W. Sakoda. On the capability of finite automata in 2 and 3 dimensional space. In 18th Ann. Symp. on Foundations of Computer Science (FOCS), pages 147-161, 1977.

[10] M. Betke, R. Rivest and M. Singh. Piecemeal learning of an unknown environment. Machine Learning 18:231-254, 1995.

[11] M. Bridgland. Universal traversal sequences for paths and cycles. J. Algorithms 8(3):395-404, 1987.

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

[13] J. Buss, and M. Tompa. Lower bounds on universal traversal sequences based on chains of length five. Information and Computation 120(2):326-329, 1995.

[14] X. Deng and C. H. Papadimitriou. Exploring an unknown graph. J. Graph Theory 32(3):265-297, 1999.

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

[16] 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.

[17] C. Duncan, S. Kobourov and V. Kumar. Optimal constrained graph exploration. In 12th Ann. ACM-SIAM Symp. on Discrete Algorithms (SODA), pages 807-814, 2001.

[18] P. Fraigniaud, D. Ilcinkas, G. Peer, A. Pelc, D. Peleg, Graph exploration by a finite automaton, Proc. 29th International Symposium on Mathematical Foundations of Computer Science (MFCS), LNCS 3153, 451-462, 2004.

[19] P. Fraigniaud, and D. Ilcinkas. Directed Graphs Exploration with Little Memory. In 21st Symposium on Theoretical Aspects of Computer Science (STACS), LNCS 1996, pages 246-257, 2004.

[20] A. Hemmerling. Labyrinth Problems: Labyrinth-Searching Abilities of Automata. Volume 114 of Teubner-Texte zur Mathematik. B. G. Teubner Verlagsgesellschaft, Leipzig, 1989.

[21] S. Hoory, and A. Widgerson. Universal traversal sequences for expander graphs. Information Processing Letters 46(2):67-69, 1993.

[22] R. Impagliazzo, N. Nisan, A. Widgerson. Pseudorandomness for network algorithms. In 26th Ann. ACM Symposium on Theory of Computing (STOC), pages 356-364, 1994.

[23] S. Istrail. Polynomial universal traversing sequences for cycles are constructible. In 20th Annual ACM Symposium on Theory of Computing (STOC), pages 491-503, 1988.

[24] J.D. Kahn, N. Linial, N. Nisan, M.E. Saks, On the cover time of random walks on graphs, J. Teoret. Probab. 2 (1989), 121-128.

[25] H. Karloff, R. Paturi and J. Simon. Universal traversal sequences of length $n^{\log n}$ for cliques. Information Processing Letters 28(5):241-243, 1988.

[26] M. Koucký. Log-space constructible universal traversal sequences for cycles of length $O(n^{4.03})$. Theoretical Computer Science 296(1); 117-144, 2003.

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

[28] N. Nisan. Pseudorandom generators for space-bounded computation. Combinatorica 12(4):449-461, 1992.

[29] P. Panaite and A. Pelc, Exploring unknown undirected graphs, J. Algorithms 33(2):281-295, 1999.

[30] N. Rao, S. Kareti, W. Shi, and S. Iyengar. Robot navigation in unknown terrains: Introductory survey of non-heuristic algorithms. Tech. Report ORNL/TM-12410, Oak Ridge National Lab., 1993.

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

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