Memory Efficient Anonymous Graph Exploration

Leszek Gąsieniec[1] Tomasz Radzik[2]

[1] Department of Computer Science, University of Liverpool, Liverpool L69 3BX, United Kingdom. E-mail: `L.A.Gasieniec@liverpool.ac.uk`
[2] Department of Computer Science, King's College London, Strand, London WC2R 2LS, United Kingdom. E-mail: `Tomasz.Radzik@kcl.ac.uk`.

Abstract. Efficient exploration of unknown or unmapped environments has become one of the fundamental problem domains in algorithm design. Its applications range from robot navigation in hazardous environments to rigorous searching, indexing and analysing digital data available on the Internet. A large number of exploration algorithms has been proposed under various assumptions about the capability of mobile (exploring) entities and various characteristics of the environment which are to be explored. This paper considers the *graph model*, where the environment is represented by a graph of connections in which discrete moves are permitted only along its edges. Designing efficient exploration algorithms in this model has been extensively studied under a diverse set of assumptions, e.g., directed vs undirected graphs, anonymous nodes vs nodes with distinct identities, deterministic vs probabilistic solutions, single vs multiple agent exploration, as well as in the context of different complexity measures including the time complexity, the memory consumption, and the use of other computational resources such as tokens and messages. In this work the emphasis is on memory efficient exploration of anonymous graphs. We discuss in more detail three approaches: *random walk*, *Propp machine* and *basic walk*, reviewing major relevant results, presenting recent developments, and commenting on directions for further research.

1 Introduction

A *graph* is a crucial combinatorial notion used for modeling complex systems in various application domains including communication, transportation and computer networks, manufacturing, scheduling, molecular biology and peer-to-peer networks. Models based on graphs often involve mobile entities which can move throughout the graph from node to node along the edges. We call such entities *agents*. An agent can be a robot servicing a hazardous environment, or a software process navigating the Internet in search for some information. *Graph exploration* refers to problems of designing algorithms (protocols) for an agent, or a group of agents, to traverse a graph in a systematic and efficient way.

In recent years the research on efficient graph exploration gathered a new momentum, generated to large extent by the theory and the applications coming ever closer together. The demand for efficient practical solutions has increased as systems of software agents moving through a large network of computers have become reality. Another example is the relevance of efficient graph exploration algorithms for efficiency of the

Internet search engines. The current applications indicate that the relative importance of various aspects of graph exploration has been changing, and those changes become reflected in the theoretical research.

In the broad context of algorithmic agent design we distinguish two main models: the *geometric model* where the search environment is represented by two- or higher- dimensional space (see, e.g., [11, 28, 59]) and the *graph model*, considered in this paper, where the environment is represented by a finite or infinite graph supported by discrete moves permitted only along its edges. The design of efficient exploration algorithms in the graph model has been extensively studied under many different assumptions, e.g., *directed* vs *undirected graphs*, *anonymous* nodes vs nodes with *distinct identities*, or *deterministic* vs *probabilistic* solutions, as well as with different performance objectives in mind including optimal time complexity, memory consumption, or use of other resources, see [1, 6, 9, 29–31, 34, 41, 56]. Different studies may also consider different aims of exploration. The aim of exploration can be to visit each node in the network, or each edge, and terminate. Alternatively, one may drop the termination requirement and ask only for *perpetual* exploration and a guarantee that each node is visited infinitely many times, or perhaps a stronger guarantee that the nodes are being visited with similar frequencies.

If nodes have distinct identities (given as $O(\log n)$-bit words), the graph stays unchanged (a static graph), and the size of the memory available to the agent (counted in words) is linear in the number of nodes of the graph, then the depth-first search (DFS) procedure gives linear time exploration. However, in many applications one or more of these assumptions might not hold. The nodes may not have unique identities; for example, if the nodes represent very simple devices (*anonymous* graphs). The graph may keep changing; for example, if it models a growing peer-to-peer network (*dynamic* graphs). Finally, the agents may have very limited memory, which may be sufficient for storing only few node identities. For example, software agents moving through a computer network from host to host might not be allowed to carry too much data with them. This survey is mainly concerned with exploration of anonymous graphs by agents equipped with bounded memory.

In graph exploration algorithms, the memory utilisation refers actually not only to the memory of the agents (“carried” by them when they move from node to node), but also to any extra memory required in the graph environment. The latter may store some (pre-computed) additional information about the graph to guide the exploration, or may allow the agent to leave marks at nodes or to move tokens as it traverses. The demand for simple and cost effective agents as well as the desire to design exploration algorithms that are suitable for rigorous mathematical analysis imply the importance of limiting the local memory of agents and their ability to manipulate the explored environment. One of the most challenging problems in the theory of computation is to look at the far ends, *border cases*, of the considered models. In case of algorithmic agent design such a border case may refer to the size of the agent’s memory, where one can limit the memory of an agent to a constant number of bits. This case is very often modeled as graph exploration by a finite state automaton and it has been extensively studied already in the 1970’s [15, 54, 58, 61]. Probably the strongest result in this setting is due to Cook and Rackoff [19]. They proved that a fixed group of finite automata, that can permanently cooperate and that can use "teleportation" to move from their current location to the location of any other automaton, cannot explore all graphs. See [42] for some recent results about limits of graph exploration with finite automata. These results imply that we either have to allow the agents to use larger memory, or to divert to randomization, or to provide the agents with extra structural information that restricts the set of graphs they have to traverse.

It has been known for some time that if we do not place strict restrictions on the local memory, then a single pebble is sufficient to explore an anonymous undirected graph. This result was extended to directed graphs by Bender *et al.* [8]. Note however that a good upper bound on the number of nodes must be known to avoid an exponential-time solution. Moreover, even if such a bound is known, the time complexity, while polynomial, remains impractically high. Another possibility is to drop determinism and to look for randomized solutions. It is known, e.g., that a random walk of length $O(n^3 \log n)$ visits all nodes of an arbitrary n-node graph with high probability [3]. Attempts to regain determinism included research on derandomization of random walks and the main approach was *universal traversal sequences* [3] that provide guidance in deterministic traversal of all graphs in a given class. Several important results have been achieved [4, 7, 46, 60] including Reingold's [60] recent asymptotically optimal $O(\log n)$-space deterministic algorithm for the undirected *st-connectivity* problem based on a novel $O(\log n)-$bit navigation mechanism. However, note that the exploration time given by this algorithm is a polynomial of a rather high degree.

Research closely related to the setting adopted in this paper assumes some structural information about the explored environment. Such additional information allows improvements in the time or memory complexity of graph traversal. The first results, concerning exploration of a labyrinth using a compass, are due to Blum and Kozen [12]. Later, Flocchini *et al.* [39] introduced a more general notion of *sense of direction* and proved that traversal can be performed using $O(n)$ messages/agent moves in this model [38]. Fraigniaud *et al.* [40] have shown that interval routing scheme can be used to achieve the same goal. In fact, given a spanning tree, the graph can be traversed using $O(n)$ moves. Pelc and Panaite [56] studied the impact of having a map of the graph on the efficiency of graph exploration. Finally Cohen *et al.* studied efficient navigation in graphs with nodes marked with a constant number of colors, see [18].

In this paper we focus on three graph traversal methods. We start with the probabilistic *random walk* method and then discuss its recently proposed deterministic counterpart known as the *Propp machine*, which requires some (small) memory at the nodes of the graph. We conclude this survey with presentation of an alternative traversal method based on the *basic walk*, in which a small amount of memory is provided to an agent and a certain type of graph preprocessing is permitted. These three methods have several interesting combinatorial properties and one might say that they have already set certain standards in agent based anonymous graph exploration.

2 The graph model

In this survey we consider environments represented by undirected (symmetric) graphs. We denote by $G = (V, E)$ the graph which is to be explored, and assume that it is connected, unless stated otherwise. It will sometimes be convenient to view G as a digraph $\overleftrightarrow{G}$ obtained by replacing each undirected edge with two arcs pointing in opposite directions. We consider graphs (environments) that are anonymous, i.e., the nodes in the graphs are neither labeled nor marked in any other way. However, the ends of edges incident to each node v are ordered and often labeled by consecutive integers $1, \ldots, d_v$ called *port numbers*, where d_v is the degree of v.

If an agent is in the current step at a node v, then the standard assumptions are that it knows d_v, the label of the port through which it has entered v and any information about the graph that it might have gathered in previous steps and has been able to (has been allowed to) store in its internal limited memory. The agent decides on the basis of this knowledge which port it should take to move in the next step to a neighbour of v. Agents do not have prior knowledge about the topology of the network. The exact details about the resources and abilities of the agents as well as about the objectives of exploration may vary from problem to problem. The number of nodes and the number of edges in G are denoted by n and m, respectively, but note that these parameters might not be known to the exploring agents.

3 The random walk

A *random walk* on an (undirected, connected) graph G starting at a node v_0 is an (infinite) random sequence $(v_0, v_1, v_2, \ldots)$ of nodes in G such that for each $i \geq 1$, node v_i is selected randomly and uniformly from all neighbours of node v_{i-1}. Using the graph exploration terminology, we say that an agent moves in step i from node v_{i-1} to its random neighbour v_i. To implement such random exploration of an arbitrary n-node graph, the agent has to be able to select a random neighbour of the current node, so it needs $O(\log n)$-bit memory and access to $\log n$ random bits per step.

The *(node) cover time of graph G from a node* v is the expected number of steps $C_v(G)$ the random walk starting from node v takes to visit all nodes of the graph. The *cover time $C(G)$ of graph G* is defined as the maximum $C_v(C)$ over all nodes $v \in V$. Thus the cover time is the worst-case (over all starting nodes) expected time of exploring the whole graph. For graphs of some special types, the cover time can be easily estimated, or even calculated exactly. For example, it is easy to show that the cover time of the graph P_n which is an n-node simple path is equal to $C(P_n) = (n-1)^2$, by solving a simple recurrence relation for the number expected number of steps H_i required to reach the end of the path starting from its i-th node. Calculation of the cover time of the n-node clique K_n is the *coupon collector* problem: at step i select randomly and uniformly one of the n coupons/nodes and wait until all coupons have been seen, with a slight modification that the next coupon/node has to be different from the one just selected. If we have already seen exactly i distinct nodes, then the probability that in the next step we will see a new node is equal to $(n-i)/(n-1)$, so the expected number of steps before a new node is encountered is equal to $(n-1)/(n-i)$ and the cover time $C(K_n)$ is equal to $\sum_{i=1}^{n-1}(n-1)/(n-i) = (n-1)\sum_{i=1}^{n-1} 1/i = n(\ln n + O(1))$.

Random walks became an important tool in algorithm design and complexity theory when Aleliunas *et al.* [3] showed in 1979 that the cover time of *every* graph is polynomial, or more specifically, at most $2m(n-1)$. Since then general techniques for bounding the cover times have been developed and bounds for the cover times for various families of graphs have been derived. An agent using a random walk on an n-node graph with the cover time bounded by $T(n) = poly(n)$ should have an $O(\log n)$-bit counter to count $T(n)$ steps to terminate with the knowledge that the whole graph has been explored with constant probability, or to count $T(n)p \log n$ steps to terminate with the knowledge the whole graph has been explored with the high probability of at least $1 - 1/n^p$. Thus a good bound $T(n)$ on the cover time is required not to expand unnecessarily the exploration time, and we review below the main known bounds. Observe that an agent implementing a random walk needs memory for two purposes. It needs $O(\log \Delta)$ bits to implement individual moves from the current node to a random neighbour, where $\Delta \le n$ is a bound on the degree of a node, and $O(\log n)$ bits to count the moves. If we do not require that the agent terminates, than the total of $O(\log \Delta)$ bits suffices. In particular, constant-size memory is sufficient for the randomized *perpetual* exploration of any constant degree graph.

Feige [36, 35] showed the following tight bounds on the range of the cover times of n-node graphs:

$$(1 - o(1))n \ln n \le C(G) \le (1 + o(1))\frac{4}{27}n^3.$$

Thus K_n is an example of a graph with the cover time achieving the lower bound, while it can be shown that the n-node *lollipop* graph given in Figure 1 has the cover time achieving the upper bound. A quadratic $O(n^2)$ upper bound on the cover time of regular graphs was first shown by Kahn, Linial, Nisan and Saks in 1989, and the best know bound of $2n^2$ is due to Feige [37]. This worst-case upper bound for regular graphs should be contrasted with Rubinfeld's [62] $O(n \log n)$ bound on the cover time of regular *expander* graphs, and with Cooper and Frieze's [20] recent result showing that the cover time of a *random* d-regular graph is $(1 + o(1))\frac{d-1}{d-2}\ n \ln n$ with high probability. The cover time of the 2-dimensional $\sqrt{n} \times \sqrt{n}$ grid is $\Theta(n \log^2 n)$, with the upper bound due to Chandra *et al.* [17] and the lower bound due to Zuckerman [65]. Aldous [2] showed that the cover time of the n-node k-dimensional grid is $\Theta(n \log n)$, for $k \ge 3$.

Aleliunas' *et al.* [3] polynomial upper bound on the cover time of an arbitrary graph was an important result for the computational complexity theory as it showed that the undirected s-t connectivity problem can be solved by a randomised log-space algorithm. This started a vast body of research with the goal to settle the conjecture that this problem can be solved by a *deterministic* log-space algorithm, and, ultimately, the more general conjecture that *any* problem solvable by a randomised log-space algorithm can be solved by a deterministic log-space algorithm. A natural approach to settle the first conjecture was to de-randomise random walks. In this framework the objective was to produce an explicit universal traversal sequence (UTS), i.e., a sequence $p_1, \ldots, p_k$ of port labels, such that the path guided by this sequence visits all edges of any graph of a given size. It is known 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) n-node graph [3]. Unfortunately, explicit short UTS are known only for special families of graphs, including 2-regular graphs [7, 13, 16, 50, 53], 3-regular graphs [49], cliques [51], and expanders [46]. Some of these sequences can be constructed in log-space, providing exploration with $O(\log n)$-bit memory. Kouckỳ [52] introduced the notion of a universal *exploration* sequence (UXS), i.e., a sequence $q_1, \ldots, q_k$ such that the agent leaves the current node x via port $p + q_i$ at the ith step, where p is the label of the port through which the agent entered node x. This notion allows to construct simpler and shorter sequences, for example, 1^n is a UXS for n-node cycles, and $(10)^n$ is a UXS for n-node cliques. Reingold [60] has recently showed that a UXS for general graphs is log-space constructible, providing a deterministic log-space algorithm for undirected s-t connectivity and settling the first of the above two conjectures. Note that both UTS and UXS require *a priori* knowledge of the size of the network. If an agent uses an UTS or UXS for exploration, with stopping, of graphs (of some type) of size at most n, then it does not know at the termination whether the graph has at most n nodes, or whether it is larger and possibly not fully explored.

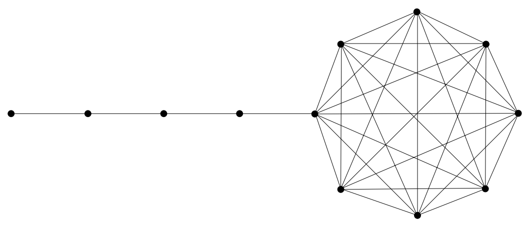

Fig. 1. The n-node lollipop graph: $(n/3)$-node path connected to n-node clique.

A random walk on a graph is an example of a *finite-state Markov chain* and has often been studied within this general context, with its central issue of the *stationary distribution* and the rate of convergence to this distribution. If a random walk starts at a node s ($v_0 = s$), then the i-th node v_i on the walk is the random variable with distribution π_s^i, where $\pi_s^i(v) = \text{Prob}(v_i = v)$. The stationary distribution π is the probability distribution on the set of nodes defined by $\pi(v) = \lim_{i \to \infty} \pi_s^i(v)$, if these limits exist and are independent of the starting node s. The stationary distribution exists for every connected non-bipartite graph, and is equal to $\pi(v) = \deg(v)/(2m)$. (For a bipartite graph, since there is no odd-length cycle, a random walk can visit a given node v either only in even steps, or only in odd steps, so $\lim_{i \to \infty} \pi_s^i(v)$ is not defined. Not to exclude bipartite graphs, the inconvenience of not having an odd length cycle is usually dealt with by allowing the walk to stay at the current node with, say, probability $1/2$, or, equivalently, by adding self-loops to the graph.) The rate of convergence to the stationary distribution is usually measured by the *mixing time*, defined as the first step t when the distribution π_s^t of the random variable v_t is guaranteed to be close to the stationary distribution π. More precisely, the mixing time is the minimum t such that $\max_{s,v \in V}\{|\pi_s^t(v) - \pi(v)|\} \leq 1/n^3$, though other definitions of the distance between two distributions, and degrees of closeness other than $1/n^3$ have also been used. The random walk is *rapidly mixing*, if the mixing time is short, say $O(n^\epsilon)$ for a small constant ϵ. Among the constant degree graphs, expanders have the best possible $O(\log n)$ mixing time.

The advantage of a random walk as a strategy for exploring a graph is its simplicity and low memory requirements. Its main drawback is the time required to complete the exploration, which is given by the cover time and can be as high as $\Omega(n^3)$. Thus a natural question is to reduce the cover time, if possible, for example by considering nonuniform transition probabilities, or by allowing the walking agent to gather, and use, some limited information about the graph. Ikeda *et al.* [47] considered the nonuniform probabilities $p(v, u)$ of moving from a node v to its neighbour u such that $p(v, u')/p(v, u'') = (\deg(u')/\deg(u''))^{1/2}$, for any two neighbours u' and u'' of node v. They showed that these transition probabilities lead to an $O(n^2 \log n)$ bound on the cover time of any graph. This quite remarkable reduction from the $O(n^3)$ bound of the uniform random walk, comes unfortunately with a cost. To implement this non-standard random walk, each node of the graph has to store information about the degrees of its neighbours, or the agent has to visit first all neighbours to gather this information. Ikeda *et al.* [47] showed also that the $O(n^2 \log n)$ bound is close to the best what we can hope for, since for any transition probabilities defined on an n-node path, the cover time is $\Omega(n^2)$.

If we are looking for graph exploration with $k \geq 2$ agents, then we would like to know good bounds on the cover time by k random walks: the expected number of steps until each node has been visited by at least one random walk (assuming all agents move simultaneously in synchronised steps). Observe that it may be, and indeed is, crucial what are the relative positions of the starting nodes of the walks. Broder *et al.* [14] considered k independent random walks starting from the stationary distribution (the starting node of walk i is a node v with probability $\pi(v)$) and showed an $O((m^2 \log^3 n)/k^2)$ bound on the cover time. Thus, for example, for constant degree graphs, the *speed-up* (the ratio of the cover times of a single random walk and k random walks) can be $\Omega(k^2/\log^3 n)$. Recently Alon *et al.* [5] showed bounds on the speed-up of k independent random walks starting from the *same* node, including a $\Theta(\log k)$ speed-up for n-node cycles, if $\log k = O(n)$, and an $\Omega(k)$ speed-up for n-node expanders, if $k \leq n$. Cooper *et al.* [22] considered k independent random walks on random regular graphs, and analysed the cover time and the time required to achieve certain interaction between the agents.

Random walks have been also applied in the context of exploring *dynamic graphs*. Cooper and Frieze [21] considered a random walk on a dynamic graph growing according to some random process, and analysed the expected proportion of visited vertices. Law and Siu [55] and Cooper *et al.* [23] used graph exploration with random walks to create new, random edges in a process of building and maintaining a well connected network.

4 The Propp machine

We consider now a type of graph exploration where the agents have no operational memory and the whole steering mechanism is provided within the environment. Thus the agents may actually be viewed as mere tokens which are being moved around the graph. We discuss the deterministic mechanism of the *rotor-router model*, which was introduced by Priezzhev in [57], further popularised by James Propp, and now known also as the *Propp machine*. In this model each node of the graph G is equipped with a small marker indicating the exit port (the edge) to be taken by an agent on the conclusion of the next visit to this node. After the agent leaves the node, the marker is moved immediately to the next port in the cyclic order. The rotor-router model has been introduced as a deterministic alternative to the random walk method. Its advantages include the balanced usage of exit ports at each node, replacing the "coupon collector" nature of the usage of ports by the random walk method.

The research on this model splits naturally into two directions associated with *finite* and *infinite* graphs. The main question for finite graphs is about properties of the periodic tour that has to be eventually adopted by the agent. For infinite graphs the Propp machine model was mainly investigated in the context of balancing schemes for even distribution of workload in networks.

Finite graphs. Note that if an agent follows the rotor-router mechanism in the Propp machine defined on a finite graph, the agent must eventually lock itself in a tour of limited size. This is a straightforward consequence of the fact that the number of configurations based on positions of markers on exit ports and location of the agent is bounded by $n(d_{max})^n$, where d_{max} is the maximum degree of a node. However, and rather surprisingly, following the rotor-router mechanism leads to a periodic tour that corresponds to an *Euler tour* defined on $\overleftrightarrow{G}$ [10]. Moreover this periodic phenomenon starts occurring very early, namely within $O(|E| \cdot n)$ steps, independently of the original configuration of the port numbers and markers as well as the agent's location. Yanovski *et al.* [64] improved this bound by showing that in fact $2|E| \cdot D$ steps suffice to form an Euler tour, where D is the diameter of G. On the other hand a lower bound $\Omega(|E| \cdot D)$ can be obtained on a lollipop graph (of a general structure as in Figure 1) in which exit ports and markers in the clique with $\Omega(|E|)$ edges are set to form an Euler tour, and markers on the external path of length D are placed on ports leading towards the clique.

Yanovski *et al.* [64] studied also behaviour of a multi-agent system of explorers in the Propp machine where the agents cooperate via shared markers. When l agents want to exit from the same node in the same step, then they all leave this node in this step through the next l consecutive ports (according to an arbitrary assignment of the agents to these l ports, and sending multiple agents through the same ports, if l is greater than the degree of the node). They proved that for a team of k agents, the numbers of edge visits in the networks are balanced up to a factor of two within at most $2(1 + \frac{1}{k})|E|D$ steps.

Infinite graphs. In the context of infinite graphs the rotor-router has been mostly studied as a deterministic analogue of the random walk approach to balancing the workload in a network, and the main question has been how similar these two processes are. The agents are now tokens, which are initially distributed among the nodes in some, possibly uneven, way (for example, they all may be initially piled up on one node). The random walk approach balances the load of tokens among nodes by sending them along independent random walks of the same length t. Let $E_t(v)$ denote the expected number of tokens which end up at a node v. The rotor-router process starting with the same initial distribution of tokens and executing t (deterministic) steps reaches a configuration with $a_t(v)$ tokens at node v. The question of similarity of these two processes is the question of analysing the node discrepancies $|a_t(v) - E_t(v)|$. In particular, Cooper and Spencer [24] studied this question for d-dimensional grids and showed that all node discrepancies are bounded by a constant, which depends on d, but does not depend on the initial configuration, the total time t and the initial rotor settings. For example, they showed a bounding constant for $d = 1$ which is ≈ 2.3. This work was followed by a detail study of the 2-dimensional grid by Doerr and Friedrich [33]. They provided evidence that the exact (constant) value of the maximum node discrepancy depends on the choice of rotor sequences. The situation is different in the case of k-ary trees: for any number D, there exists an initial configuration of tokens, a number t and a node v such that after t steps of the rotor-router process node v has at least D tokens more than it is expected to get in the random walk process [25].

5 The basic walk

The basic walk method is based on an observation that one can cover a graph $G = (V, E)$, or more precisely its symmetric digraph counterpart $\overleftrightarrow{G}$, by a collection of directed cycles. The cycles are formed according to a simple rule. At any node v with the degree d_v, the incoming arc incident via a port i becomes the predecessor of the outgoing arc incident via port $(i \mod d_v) + 1$. Note that since each arc in the digraph has a unique predecessor as well as a unique successor a collection of arc-disjoint cycles containing all arcs in the digraph is formed. Figure 2 shows an example.

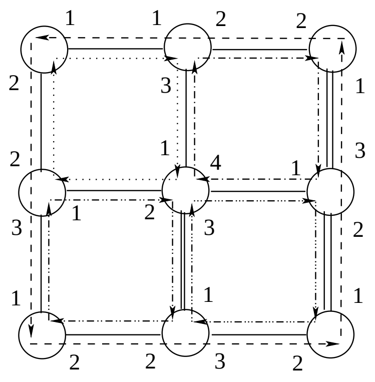

Fig. 2. The formation of cycles

Note also that a certain arrangement of the port numbers may lead to a cycle that visits all nodes in the graph (symmetric digraph), see Figure 3. We call such a cycle a *witness cycle*, see [32]. The presence of a witness cycle is very convenient in the context of graph exploration, since it can be used by a very simple mobile entity to periodically visit all nodes in the graph.

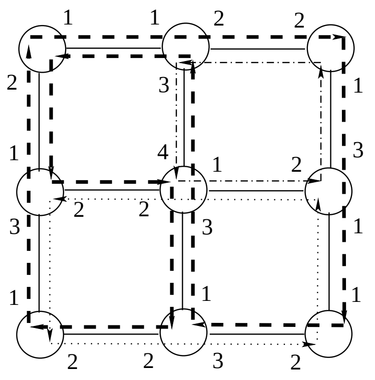

Fig. 3. A witness cycle

We consider two types of graph traversal based on the basic walk method. In the *oblivious model* an agent has no operational memory. It is equipped with a simplest possible mechanism that allows it to follow a single cycle in the collection of cycles covering the input graph. In this model the main task is to provide port labeling to the input graph such that a witness cycle is formed. Moreover one is interested in forming a shortest possible witness cycle. Note that in this model an agent traverses indefinitely along the chosen cycle since its has no memory to make any decisions and no state to stop. In the *adaptive model* an agent is provided with small (constant number of bits) memory that allows the agent to switch between different cycles in order to shorten the route covering all nodes in the graph. Such an agent is often modeled as Mealy automaton, where the output (outgoing port number) depends on the input (incoming port number) and the state of the automaton. More formally the automaton has a transition function f and a finite number of states governing the actions of the agent. When the agent enters a node v of degree d_v through port i, it switches to state s' and exits the node through port i', where $(s', i') = f(s, i, d_v)$, see [44] for more detail description.

Oblivious traversal. The oblivious traversal based on the basic walk method was first proposed by Dobrev *et al.* in [32]. The authors claimed that there exists a port labeling, such that the oblivious agent requires at most $10n$ steps to visit all n nodes of a graph in a periodic manner. While the stated problem and several combinatorial observations, in particular merging and exchanging contents of cycles, attracted attention in the community, the bound of $10n$ on their port labeling turned out to be incorrect. Very recently Czyzowicz *et al.* in [27] proposed a polished version of the previous argument supported by a new combinatorial structure of a *three-layer partition* of graphs. This led to the first provably correct port labeling inducing a linear tour of length $4\frac{1}{3}n$. Moreover, the labeling based on the three-layer partition can be performed in the optimal $O(|E|)-$time. The authors proposed also a non-trivial class of graphs in which one can select a spanning tree such that each node is incident to some edge outside of the spanning tree. For this class of graphs one can construct a labeling that forms a witness cycle of length $\leq 2n-2$. An example of formation of the witness cycle in this class of graphs is presented in Figure 4.

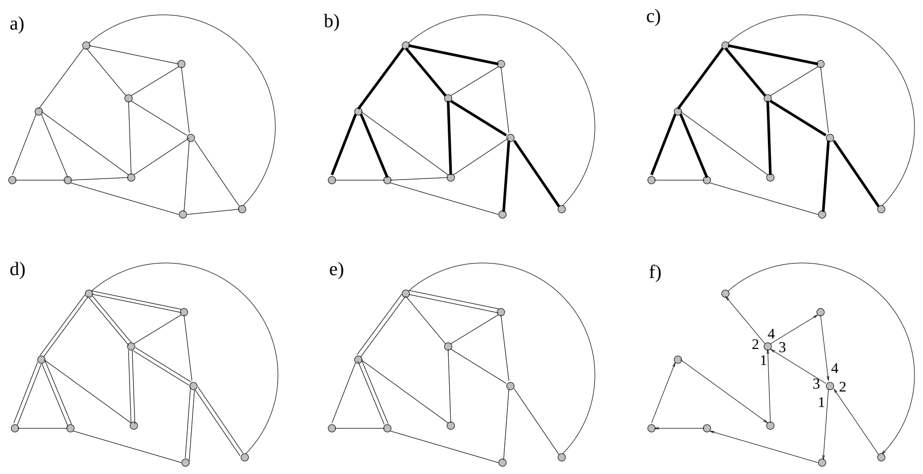

Fig. 4. Formation of a witness cycle: a) an input graph, b) a spanning tree with external edges at each node is formed, c) excessive external edges are dropped, d) the spanning tree edges are doubled, e) the parity of nodes is repaired starting from leaves in the tree, f) remaining double (non-bridging) edges are dropped and a witness cycle is formed.

Unfortunately the problem of deciding whether the input graph has a spanning tree with the required property is *NP-hard* since this problem corresponds to selection of a Hamiltonian path in 3-regular graphs, which is known to be hard. The authors in [27] give also an n-node graph, shown in Figure 5, in which the witness cycle must contain all arcs in $\overleftrightarrow{G}$ and therefore cannot be shorter than $2.8n$.

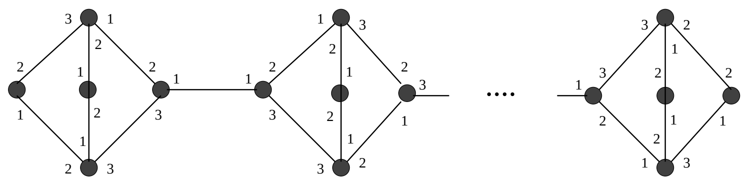

Fig. 5. A feasible solution in which each edge must be used in two directions

It is worth mentioning that the explicit labeling of ports in G is not essential in the oblivious graph traversal. In fact, it is enough to provide a periodic ordering of ports at each node in G. The agent when arrives at some node v via some port must know only its successor in the periodic order to continue the walk along the chosen cycle.

Adaptive traversal. In [48] David Ilcinkas noted that the tour used by an agent to visit all nodes in the graph can be shortened to $4n-2$ if the agent is provided with a 2-bit operational memory. The extra memory translated to a larger number of states allows the agent to perform context sensitive decisions including switching between the cycles available in the basic walk method. More precisely, Ilcinkas noticed that using a certain type of port labeling cycles and the extra memory bits, one can force the agent to follow an Euler tour defined on a chosen spanning tree T in G. The spanning tree contains a unique *root edge* $e_r = (v_r, w_r)$ that bears port number 1 at its both ends. Apart from v_r and w_r every node u in G has a parent which is reachable from u via port 1. In fact one can also interpret that v_r is a parent of w_r and vice versa. All children of any node v in G (including root nodes) are reachable from v via ports 2 through $c_v + 1$, where c_v is the number of children of v in the spanning tree T. When the agent traverses along the Euler tour down in the tree it follows consecutive arcs of some cycle provided by the basic walk method. This process is terminated when for two consecutive nodes on the cycle v_1 and v_2, the node v_2 is entered via port different from 1, meaning that v_2 is not a child of v_1 in T. In this case the agent returns first to v_1 and then to the parent v of v_1, concluding that there are no more children of v_1 to be visited in T. The traversal is then continued along some other cycle starting at the next child (if any) of v. The edge (v_1, v_2) is called a *penalty edge* since it does not belong to T and it contributes an extra two agent moves in the tour that visits all nodes in the graph. The total length of the tour can be bounded by $2n-2$ moves along edges in the spanning tree (each edge has to be visited in two directions) and $2n$ moves along the penalty edges (each node including root nodes may have incident edges not forming a part of the spanning tree T). Thus the total length of the tour is bounded by $4n-2$. Ilcinkas claimed also that $4n-2$ is the exact bound for all agents equipped with a small (constant size) memory.

This claim was later disproved by Gąsieniec *et al.* in [44] where they showed that the length of the tour could be shortened to $3.75n-2$. The improvement was possible due to the observation that visits to penalty edges could be avoided at the fraction $\frac{1}{8}$ of the nodes. They proved that one can construct port labeling in which either a large fraction of nodes is *saturated* (all incident edges to these nodes are present in the spanning tree) or there are large clusters of sibling leaves and extended leaves (paths of length 2) located at the bottom of the spanning tree. Within each cluster the penalty edges are visited only at the first sibling, while visiting all other siblings in the same cluster is penalty free. This result was recently further improved in [27] to $3.5n-2$ with help of the *three-layer partition* and *sham edges* that pretend to be penalty edges while serving as proper edges in the spanning tree.

Note here that while the explicit labeling of ports in G is not essential in the adaptive graph traversal at least one port at each node has to be distinguished in order to form the hierarchical structure of the spanning tree T. More precisely the marked ports play the same role as ports 1 in the explicit labeling setting.

6 Conclusion

The standard random walks are memoryless in the sense that the selection of the next node does not depend on the past. This assumption is important from the point of view of the probabilistic methods used in the theoretical analysis. Simulations show that the performance of random walks may be improved for some types of graphs, if the agents are allowed to remember, and use, some very limited information about the past. It would be very interesting to develop some theoretical analysis of such processes.

Further work on the random walk approach should include further efforts on derandomisation with limited random access memory, ideally considering also the secondary objective of minimizing the time complexity. An interesting aspects of random walks can be considered in the context of efficiency of pseudo-random number generators. A pseudo-random number generator can be seen as a deterministic steering mechanism of an agent and the efficiency of such a generator may be expressed as the efficiency of exploration in arbitrary graphs.

Research on the rotor-router mechanism started only very recently and further results on comparing this approach with random walks should be coming in near future. One possible interesting question is whether the results for infinite graphs can be used to obtain implications for their finite counterparts. Also better understanding of Euler tours formed by the rotor-router mechanism by single and multiple robots would be highly appreciated. Another direction for studies of this model is graph exploration by agents granted dynamic port labeling mechanism. This would refer to creation of various geometrical shapes and surfaces.

Finally, in the context of the basic walk approach, further understanding of witness cycles as well as tours used by agents equipped with a small memory is required. There is also very little known so far about the case when the ports at each node form a random permutation.

7 Acknowledgements

Many thanks go to Colin Cooper, Jurek Czyzowicz, Stefan Dobrev, David Ilcinkas, Jesper Jansson, Ralf Klasing, Adrian Kosowski, Ioannis Lignos, Russell Martin, and Alfredo Navarra for a number of valuable discussions on the main themes of this survey.

References

1. S. Albers and M. R. Henzinger. Exploring unknown environments. *SIAM Journal on Computing*, 29:1164–1188, 2000.
2. D. J. Aldous. On the time taken by random walks on finite groups to visit every state. *Journal Probability Theory and Related Fields*, 62(3):361–374, 1983.
3. R. Aleliunas, R. M. Karp, R. J. Lipton, L. Lovasz, and C. Rackoff. Random walks, universal traversal sequences, and the complexity of maze problems. *FOCS 1979*, pages 218–223, 1979.
4. N. Alon, Y. Azar, and Y. Ravid. Universal sequences for complete graphs. *Discrete Appl. Math.*, 27(1-2):25–28, 1990.

5. N. Alon, C. Avin, M. Koucky, G. Kozma, Z. Lotker, and M. R. Tuttle. Many random walks are faster than one. In *SPAA '08: Proceedings of the twentieth annual symposium on Parallelism in algorithms and architectures*, pages 119–128, New York, NY, USA, 2008. ACM.
6. B. Awerbuch, M. Betke, R.L. Rivest, and M. Singh. Piecemeal graph exploration by a mobile robot. *Information and Computation*, 152(2):155–172, 1999.
7. A. Bar-Noy, A. Borodin, M. Karchmer, N. Linial, and M. Werman. Bounds on universal sequences. *SIAM J. Comput.*, 18:268–277, 1989.
8. 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.
9. M. Bender and D. K. Slonim. The power of team exploration: two robots can learn unlabeled directed graphs. In *Proc. of FOCS 94*, pages 75–85, 1994.
10. S. Bhatt, S. Even, D. Greenberg, and R. Tayar, Traversing Directed Eulerian Mazes, *Journal of Graph Algorithms and Applications*, Vol. 6(2), pp. 157-173 (2002).
11. A. Blum, P. Raghavan, and B. Schieber. Navigating in unfamiliar geometric terrain. *SIAM Journal on Computing*, 26:110–137, 1997.
12. M. Blum and D. Kozen. On the power of the compass (or, why mazes are easier to search than graphs). In *Proc. of FOCS 78*, pages 132–142, 1978.
13. M. F. Bridgland. Universal traversal sequences for paths and cycles. *J. Algorithms*, 8(5):395–404, 1987.
14. A. Z. Broder, A. R. Karlin, P. Raghavan, and E. Upfal. Trading space for time in undirected s-t connectivity. *SIAM J. Comput.*, 23(2):324–334, 1994.
15. L. Budach. Automata and labyrinths. *Math. Nachrichten*, pages 195–282, 1978.
16. J. F. Buss and M. Tompa. Lower bounds on universal traversal sequences based on chains of length five. *Inf. Comput.*, 120(2):326–329, 1995.
17. A. K. Chandra, P. Raghavan, W. L. Ruzzo, and R. Smolensky. The electrical resistance of a graph captures its commute and cover times. In *Proc. STOC '89: Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 574–586, New York, NY, USA, 1989. ACM.
18. R. Cohen, P. Fraigniaud, D. Ilcinkas, A. Korman, and D. Peleg. Label-guided graph exploration by a finite automaton. In *Proc. of the 32nd International Colloquium on Automata, Languages and Programming (ICALP)*, pages 335–346, 2005.
19. S. A. Cook and C. Rackoff. Space lower bounds for maze threadability on restricted machines. *SIAM Journal on Computing*, 9(3):636–652, 1980.
20. C. Cooper and A. Frieze. The cover time of random regular graphs. *SIAM J. Discret. Math.*, 18(4):728–740, 2005.
21. C. Cooper and A. M. Frieze. Crawling on web graphs. In *Proc. STOC '02*, pages 419–427, 2002.
22. C. Cooper, A. M. Frieze, and T. Radzik. Multiple random walks in random regular graphs. Unpublished manuscript, 2008.
23. C. Cooper, R. Klasing, and T. Radzik. A randomized algorithm for the joining protocol in dynamic distributed networks. Technical Report RR-1432-07, LaBRI, Bordeaux, France, June 2007.
24. J.N. Cooper and J. Spencer, Simulating a random walk with constant error, *Combinatorics, Probability and Computing*, Vol. 15, pp. 815-822 (2006).
25. J. Cooper, B. Doerr, T. Friedrich, and J. H. Spencer. Deterministic random walks on regular trees. In *Proc. of SODA 2008*, pages 766–772, 2008.
26. J. Cooper, B. Doerr, J. H. Spencer, and G. Tardos. Deterministic random walks on the integers. *European Journal of Combinatorics*, 28(8):2072–2090, 2007.
27. J. Czyzowicz, S. Dobrev, L. Gąsieniec, D. Ilcinkas, J. Jansson, R. Klasing, I. Lignos, R. Martin, K. Sadakane, and W.-K. Sung. More efficient periodic traversal in anonymous undirected graphs. Unpublished manuscript, 2008.

28. X. Deng, T. Kameda, and C. H. Papadimitriou. How to learn an unknown environment: The rectilinear case. *Journal of the ACM*, 45:215–245, 1998.
29. X. Deng and C. H. Papadimitriou. Exploring an unknown graph. *Journal of Graph Theory*, 32(3):265–297, 1999.
30. A. Dessmark and A. Pelc. Optimal graph exploration without good maps. *Theoretical Computer Science*, 326(1-3):343–362, 2004.
31. K. Diks, P. Fraigniaud, E. Kranakis, and A. Pelc. Tree exploration with little memory. *Journal of Algorithms*, 51:38–63, 2004.
32. S. Dobrev, J. Jansson, K. Sadakane, and W.-K. Sung. Finding Short Right-Hand-on-the-Wall Walks in Graphs. In *12th Colloquium on Structural Information and Communication Complexity SIROCCO*, volume LNCS 3499, pages 127 – 139, 2005.
33. B. Doerr and T. Friedrich, Deterministic Random Walks on the Two-Dimensional Grid, *17th International Symposium on Algorithms and Computation*, ISAAC'06, pp. 474-483.
34. C. A. Duncan, S. G. Kobourov, and V.S.A. Kumar. Optimal constrained graph exploration. In *ACM Transaction on Algorithms*, volume 2(3), pages 380–402, 2006.
35. U. Feige. A tight upper bound on the cover time for random walks on graphs. *Random Structures and Algorithms*, 6(1):51–54, 1995.
36. U. Feige. A tight lower bound on the cover time for random walks on graphs. *Random Struct. Algorithms*, 6(4):433–438, 1995.
37. U. Feige. Collecting coupons on trees, and the cover time of random walks. *Computational Complexity*, 6(4):341–356, 1996.
38. P. Flocchini, B. Mans, and N. Santoro. On the impact of sense of direction on message complexity. *Information Processing Letters*, 63(1):23–31, 1997.
39. P. Flocchini, B. Mans, and N. Santoro. Sense of direction: definition, properties and classes. *Networks*, 32(3):165–180, 1998.
40. P. Fraigniaud, C. Gavoille, and B. Mans. Interval routing schemes allow broadcasting with linear message-complexity. *Distributed Computing*, 14(4):217–229, 2001.
41. P. Fraigniaud and D. Ilcinkas. Digraph exploration with little memory. In *21st Symp. on Theoretical Aspects of Computer Science (STACS'04)*, pages 246–257, 2004.
42. P. Fraigniaud, D. Ilcinkas, G. Peer, A. Pelc, and D. Peleg. Graph exploration by a finite automaton. In *Theoretical Computer Science*, volume 345(2-3), pages 331–344, 2005.
43. M. R. Garey, D. S. Johnson, and R. E. Tarjan. The planar hamiltonian circuit problem is np-complete. *SIAM J. Comput.*, 5(4):704–714, 1976.
44. L. Gąsieniec, R. Klasing, R. Martin, A. Navarra, and X. Zhang. Fast Periodic Graph Exploration with Constant Memory. *J. Comput. Syst. Sci.*, 74(5):808 – 822, 2007.
45. L. Gąsieniec, A. Pelc, T. Radzik, and X. Zhang. Tree exploration with logarithmic memory. In *SODA*, pages 585 – 594, 2007.
46. S. Hoory and A. Wigderson. Universal traversal sequences for expander graphs. *Inf. Process. Lett.*, 46(2):67–69, 1993.
47. S. Ikeda, I. Kubo, N. Okumoto, and M. Yamashita. Impact of local topological information on random walks on finite graphs. In *Automata, Languages and Programming, the 30th International Colloquium, ICALP*, volume LNCS 2719, pages 1054–1067, 2003.
48. D. Ilcinkas. Setting Port Numbers for Fast Graph Exploration. In *13th Colloquium on Structural Information and Communication Complexity SIROCCO*, volume LNCS 4056, pages 59 – 69, 2006.
49. R. Impagliazzo, N. Nisan, and A. Wigderson. Pseudorandomness for network algorithms. In *Proc. STOC '94: Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pages 356–364, New York, NY, USA, 1994. ACM.
50. S. Istrail. Polynomial universal traversing sequences for cycles are constructible. In *Proc. STOC '88: Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 491–503, New York, NY, USA, 1988. ACM.

51. H. J. Karloff, R. Paturi, and J. Simon. Universal traversal sequences of length $n^{O(\log n)}$ for cliques. *Inf. Process. Lett.*, 28(5):241–243, 1988.
52. M. Koucký. Universal traversal sequences with backtracking. *J. Comput. Syst. Sci.*, 65(4):717–726, 2002.
53. M. Koucký. Log-space constructible universal traversal sequences for cycles of length o(n4.03). *Theor. Comput. Sci.*, 296(1):117–144, 2003.
54. D. Kozen. Automata and planar graphs. In *Proc. of Fundations Computatial Theory (FCT 79)*, pages 243–254, 1979.
55. C. Law and K.-Y. Siu. Distributed construction of random expander graphs. In *Proc. 22-nd Annual Joint Conference of the IEEE Computer and Communications Societies*, April 2003.
56. P. Panaite and A. Pelc. Impact of topographic information on graph exploration efficiency. *Networks*, 36:96–103, 2000.
57. V.B. Priezzhev, D. Dhar, A. Dhar, and S. Krishnamurthy, Eulerian walkers as a model of selforganized criticality, *Physics Review Letters*, Vol. 77, pp. 5079–5082, 1996.
58. M. O. Rabin. Maze threading automata. Technical Report Seminar Talk, University of California at Berkeley, October 1967.
59. N. Rao, S. Kareti, W. Shi, and S. Iyengar. Robot navigation in unknown terrains: Introductory survey of length,non-heuristic algorithms. Technical Report ORNL/TM12410, Oak Ridge National Lab., 1993.
60. O. Reingold. Undirected st-connectivity in log-space. In *Proc. STOC'05*, pages 376–385, 2005.
61. H. A. Rollik. Automaten in planaren graphen. *Acta Informatica*, 13:287–298, 1980.
62. R. Rubinfeld. The cover time of a regular expander is $O(nlogn)$. *Information Processing Letters*, 35:49–51, 1990.
63. P. Winkler and D. Zuckerman. Multiple cover time. *Random Structures and Algorithms*, 9:403–411, 1996.
64. V. Yanovski, I.A. Wagner, and A.M. Bruckstein, A Distributed Ant Algorithm for Efficiently Patrolling a Network, *Algorithmica*, Vol. 37, pp. 165–186 (2003).
65. D. Zuckerman. A technique for lower bounding the cover time. *SIAM J. Discret. Math.*, 5(1):81–87, 1992.