Algorithmica (2012) 63:26–38
DOI 10.1007/s00453-011-9518-1

Graph Decomposition for Memoryless Periodic Exploration

Adrian Kosowski · Alfredo Navarra

Received: 6 April 2010 / Accepted: 19 April 2011 / Published online: 3 May 2011

Abstract We consider a general framework in which a memoryless robot periodically explores all the nodes of a connected anonymous graph by following local information available at each vertex. For each vertex v, the endpoints of all edges adjacent to v are assigned unique labels within the range 1 to $\deg(v)$ (the degree of v). The generic exploration strategy is implemented using a right-hand-rule transition function: after entering vertex v via the edge labeled i, the robot proceeds with its exploration, leaving via the edge having label $[i \bmod \deg(v)] + 1$ at v.

A lot of attention has been given to the problem of labeling the graph so as to achieve a periodic exploration having the minimum possible length π. It has recently been proved (Czyzowicz et al., Proc. SIROCCO'09, 2009) that $\pi \leq 4\frac{1}{3}n$ holds for all graphs of n vertices. Herein, we provide a new labeling scheme which leads to shorter exploration cycles, improving the general bound to $\pi \leq 4n - 2$. This main result is shown to be tight with respect to the class of labellings admitting certain connectivity

The research was partially funded by the State Committee for Scientific Research (Poland) Grant 4 T11C 047 25, by the ANR-project "ALADDIN" (France), and by the project "CEPAGE" of INRIA (France).
An extended abstract of this paper appeared in the proceedings of the 34th International Symposium on Mathematical Foundations of Computer Science (MFCS) [12].

A. Kosowski
LaBRI - Université Bordeaux 1, 351 cours de la Libération, 33405 Talence, France
e-mail: adrian@kaims.pl

A. Kosowski
Department of Algorithms and System Modeling, Gdańsk University of Technology, Narutowicza 11/12, 80233 Gdańsk, Poland

A. Navarra (✉)
Dipartimento di Matematica e Informatica, Università degli Studi di Perugia, Via Vanvitelli 1, 06123 Perugia, Italy
e-mail: navarra@dmi.unipg.it

properties. The labeling scheme is based on a new graph decomposition which may be of independent interest.

Keywords Anonymous graph · Labeling · Exploration algorithm · Oblivious robots

1 Introduction

The problem of finding Eulerian cycles and Hamiltonian cycles is at the heart of graph theory. In fact, many applications arise in different contexts. One of these can be found in the field of graph exploration by means of a mobile entity. The entity might be represented, for instance, by a software agent or a robot. The task of periodic visiting of all vertices of a network is particularly useful in network maintenance, where the status of every vertex has to be checked regularly. According to the imposed model as well as to the robot capabilities, the problem may vary in its complexity.

1.1 Model Assumptions

We assume that the explored graph $G = (V, E)$ is simple, undirected, connected, and anonymous, i.e., the vertices in the graph are neither labeled nor colored. However, while visiting a vertex the robot can distinguish between its adjacent edges. This is achieved by using a predefined local ordering of edges known as a *local orientation*, which can be given in one of the following two ways:

- An implicit *cyclic ordering*, given at each node $v \in V$ as a local cyclic ordering of adjacent edges. This is encoded through a *NextPort* function which naturally defines a transition operation known as the *right-hand-rule* (or *basic walk* [10]): a robot entering vertex v by an adjacent edge e leaves this vertex by the next edge adjacent to v in the specified cyclic order, $\mathit{NextPort}(e)$. The local ordering of edges is cyclic in the sense that successive applications of *NextPort* iterate through all of the edges adjacent to v.
- An explicit *port labeling*, in which, for each vertex $v \in V$, there exist consecutive integer labels (starting from 1), also called *port numbers*, preassigned to all the edges adjacent to v, next to the endpoint v of each edge. Thus, each edge of the graph is assigned two port numbers, one for each endpoint. The labels at node v are always distinct and form the discrete interval $[1, \deg(v)]$, where $\deg(v)$ is the degree of v in G. In such a setting, the natural definition of *NextPort* for port number l at vertex v is $\mathit{NextPort}(l) := [l \bmod \deg(v)] + 1$.

A robot, initially located at a generic vertex v, starts the exploration of G by traversing the edge having label 1 at endpoint v. Once it has reached the other endpoint of this edge, say u, it reads the associated label, say l, and enters to the neighboring vertex u of v. In order to keep on with the exploration, the robot now follows the right-hand rule, leaving vertex u by the edge labeled $\mathit{NextPort}(u)$. In doing so, eventually the robot will re-enter port 1 at vertex v, and the traversal will proceed periodically from then on. We will say that the robot *explores* the graph if its route goes through each vertex of the graph at least once; from now on, we will only consider port labellings leading to valid explorations. It is known that all graphs admit a port labeling leading to an exploration [6].

1.2 Studied Parameter and Its Motivation

For a given port labeling, the *exploration period* π is defined as the total number of steps made by the robot before returning to the initial port (or equivalently, as the total number of arcs of the form (u, v), for $\{u, v\} \in E$, used during the exploration). In this paper, we focus on finding labellings which lead to valid explorations of minimum possible period. This immediately leads to the natural definition of the graph parameter $\pi(G)$ known as the *minimum exploration period* of the graph.

The parameter $\pi(G)$ obviously characterizes the best-case behavior of the basic walk on G, but its studies have in fact some much stronger motivation:

- $\pi(G)$ is the minimum possible exploration period for any oblivious robot (i.e., a robot which is not equipped with any state information which survives when traversing an edge), even if the labeling of the graph is given using explicit port numbers, and the choice of the next edge is not necessarily governed by the right-hand-rule [5]. In other words, in the context of these studies, the basic walk is fully representative of all oblivious exploration strategies.
- the value of $\pi(G)$, expressed in relation to the number of nodes n, exposes certain interesting structural properties of the graph. For example, we have that $\pi(G) = n$ if and only if G is Hamiltonian, and for a Hamiltonian graph an appropriate labeling can be defined so as to direct ports 1 and 2 of all nodes along the edges of the Hamiltonian cycle (Fig. 1a). Moreover, it is also known that $\pi(G) < 2n$ for all graphs admitting a spanning tree T such that $G \setminus T$ has no isolated vertices [5].

Periodic graph exploration requires that the robot has to visit every vertex infinitely many times in a periodic manner. In practice, we do not require the cycle to touch each vertex exactly once but only that all the vertices are touched. In this setting, every connected graph admits an exploration cycle. In fact, one may consider, for instance, the cycle determined by the visit of a spanning tree of the input graph. In this case, the period would be of length $2n$ as each edge of the spanning tree is traversed twice in one cycle. However, the difficulty resides in encoding somehow the cycle only by means of an appropriate labeling scheme (see for instance [3] for recent studies on this matter). Encoding the spanning tree is not always possible. Also in the example shown in Fig. 1 it is not clear whether there exists a labeling scheme to allow a robot to follow the cycle determined by the spanning tree by means of a fixed rule. Actually, in [5] a lower bound of $2.8n$ for the length of an exploration cycle

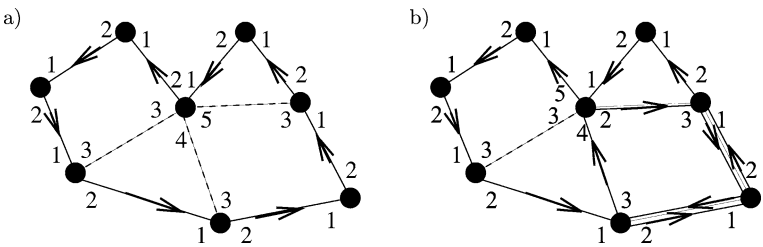

Fig. 1 Exploration cycles obtained for different labellings: (**a**) a labeling leading to a Hamiltonian cycle, (**b**) another exemplary labeling

within an additive factor has been provided when the robot is oblivious, i.e. it has no capabilities rather than to follow one specific rule based on the local orientation such as the right-hand-rule.

1.3 Related Work

Periodic graph exploration problems have been studied in a wide variety of contexts; we confine ourselves to a brief survey of directly related results in the model of anonymous undirected graphs with local port labels, and robots are deterministic and equipped with no memory (or a very small number of bits of memory).

When no assumptions are made about the port labeling, it is a well-established fact [1] that no oblivious robot can explore all graphs. In [14], the impossibility result was extended to a finite team of robots, showing that they cannot explore all planar cubic graphs. This result is improved in [4], where the authors introduce a powerful tool, called the Jumping Automaton for Graphs (JAG). A JAG is a finite team of finite automata that permanently cooperate and that can use *teleportation* to move from their current location to the location of any other automaton. However, even JAGs cannot explore all graphs. The proof that a robot requires at least n states (and thus $\Omega(\log n)$ bits of state memory) to explore all graphs of order n can be found in [7]. On the other hand, by a seminal result of Reingold [13], a robot equipped with $\Theta(\log n)$ memory can perform a deterministic exploration of any graph, and the resulting exploration period is thus polynomial with respect to n. In [8], the authors investigate the graph exploration capability with respect to the memory size provided to a robot. Another way of assisting exploration is described in [2], where it is shown that an appropriate pre-coloring of the graph using 3 colors always allows a robot with constant memory to successfully complete the exploration.

A natural problem consists in manually setting up the local orientation of the port numbers in order to allow an oblivious robot to efficiently explore the input graph. This line of study, which is also pursued herein, was introduced in [6]. That paper provided the first constructions of port labellings leading to short exploration periods for an oblivious robot, showing that for any graph on n nodes, we have $\pi(G) \leq 10n$. Recently, by applying a clever graph decomposition technique in order to build an appropriate exploration cycle, [5] have improved this bound to $\pi(G) \leq 4\frac{1}{3}n$. They have also shown a strong worst-case lower bound: for arbitrarily large values of n, there exist n-node graphs G_n such that $\pi(G_n) \geq 2.8n - O(1)$.

An interesting variation to this problem was proposed in [11], where the robot is equipped with few extra memory bits; we will denote the exploration periods in such a model by π_c. In [11] it is shown how to obtain an exploration period $\pi_c(G) \leq 4n - 2$, regardless of the starting vertex of the robot. The obtained bound has since been improved in [9] to $\pi_c(G) < 3.75n - 2$ by exploiting some particular graph properties, still allowing only constant memory. The constant memory model was also addressed in [5] and the bound was further improved to $\pi_c(G) < 3.5n - 2$ by using a combination of the properties from [9] and the new decomposition technique also used in [9] for the oblivious case. Interestingly enough, apart from the relation $\pi_c(G_n) \geq 2n - 2$ which clearly holds whenever G_n is a tree on n nodes, there are to date no known non-trivial lower bounds on the worst case value of parameter π_c.

1.4 Our Results

The central result of this paper is to establish an improved bound on the minimum exploration period of any graph G on n nodes, namely, $\pi(G) \leq 4n - 2$. The proof is constructive, and we also show that a port labeling which guarantees an exploration period of at most $4n - 2$ for the basic walk can be computed in polynomial time in a centralized setting. The result is obtained by applying a new graph decomposition technique which may be of interest in its own right, and a completely new analysis. Our labeling preserves a structural property first introduced in [5]: the set of edges which are traversed twice (in opposite directions) during one exploration period is a connected spanning subgraph of G. We present an example showing that the worst-case analysis of our approach is tight, and that our approach is the best possible when restricted to the class of explorations having the stated structural property.

1.5 Outline

The next section introduces some notation. Section 3 describes the structural properties defined in [5] which we use in order to construct a suitable graph decomposition. The main proofs are given in Sect. 4. Section 4.1 provides lemmata which characterize the decomposition that we introduce, while Sect. 4.2 formally describes the corresponding algorithmic procedure. Further properties of the labellings are discussed in Sect. 5. Section 5.1 describes a modification of our construction, leading to a polynomial time algorithm for computing an appropriate labeling. Section 5.2 shows that for some graphs, the proposed construction is the best possible with respect to the applied graph decomposition. Final remarks are given in Sect. 6.

2 Preliminaries and Notation

Before describing how the labeling scheme is set up for a graph exploration, we introduce some notation. For a graph or multigraph H, we will denote by $V(H)$ its vertex set, by $E(H)$ its edge multiset, and by $|E(H)|$ the number of its edges (including multiple edges). The number of edges adjacent in H to a vertex $v \in V(H)$ is denoted by $\deg_H(v)$. The notation $2e$ denotes 2 copies of an edge e; the notation $2H$ denotes a multigraph with vertex set $V(H)$ and each edge $e \in E(H)$ replaced by $2e$. An edge $e \in H$ is called *double* if $2e$ belongs to H, and *single* otherwise. Throughout the paper we will never consider multigraphs with more than two parallel edges.

Let $G = (V, E)$ be a connected simple graph, with $|V| = n$. Any labeling scheme for G uniquely determines an exploration cycle, understood as a sequence of directed edges traversed in a single period of the exploration, i.e., a sequence in which the directed edge (u, v) corresponds to a transition of the robot from some node u to another node v, where $\{u, v\} \in E$. The corresponding *exploration multigraph* H is defined as the undirected submultigraph of $2G$ given by the edges of G traversed by the robot during one exploration cycle (each edge is included as it is traversed, possibly twice if it is traversed in both directions). Let H_2 be the spanning subgraph

of H consisting of its double edges only, and let $H_1 = H \setminus H_2$. A vertex $v \in V$ is called *saturated* in H with respect to G if $\deg_H(v) = \deg_{2G}(v)$.

In the next section, we show how to build the exploration multigraph H by keeping H_2 as small as possible, so as not to allow the robot to traverse too many edges twice.

3 The Graph Decomposition

In order to provide the new construction of the exploration multigraph H, we need to briefly discuss some of the structural properties of exploration multigraphs. The following simple observation holds (cf. e.g. [5] for a more detailed discussion).

Proposition 1 [5] *Any exploration multigraph $H \subseteq 2G$ has the following properties*:

A. *For each vertex $v \in V$, $\deg_H(v)$ is even.*
B. *Each vertex $v \in V$ having $\deg_{H_1}(v) = 0$ is saturated in H with respect to G.*

The converse of the above proposition does not hold in general, but one more additional property can also be formulated.

C. *H_2 is connected.*

Then, the following structural theorem has recently been shown.

Theorem 1 [5] *Any multigraph $H \subseteq 2G$ fulfilling properties A, B, and C is a valid exploration multigraph, i.e. induces an exploration cycle on G of length at most $|E(H)|$.*

We can thus concentrate on defining a multigraph which satisfies properties A, B, and C. To achieve this, in graph G we select an arbitrary spanning tree T_0. Let $G' = G \setminus T_0$. Then, in multigraph $2G'$ we find a spanning (not necessarily connected) submultigraph H' satisfying properties analogous to A and B:

A′. *For each vertex $v \in V$, $\deg_{H'}(v)$ is even.*
B′. *Each vertex $v \in V$ having $\deg_{H'_1}(v) = 0$ is saturated in H' with respect to G'.*

The final multigraph H is given as $H = H' \cup 2T_0$, thus $2T_0 \subseteq H_2$. It is clear that H satisfies properties A, B, C, and that $|E(H)| = |E(H')| + 2(n-1)$. Note that the construction of H' can be performed independently for each connected component of G'; throughout the rest of the discussion, w.l.o.g. we assume that G' is connected. Hence, in order to obtain an exploration cycle with period $\pi(G) \leq 4n-2$, we confine ourselves to constructing an appropriate submultigraph $H' \subseteq 2G'$ with $|E(H')| \leq 2n$. So, it remains to show the following theorem, which constitutes the main result of our paper.

Theorem 2 *For any connected graph G' with vertex set V, $|V| = n$, there exists a multigraph $H' \subseteq 2G'$ such that $|E(H')| \leq 2n$, and H' satisfies properties A′ and B′.*

4 Proof of Theorem 2

Let T be a rooted spanning tree in G' with root r. We will call a vertex $v \in V$ *tree-saturated* in T if $\deg_T(v) = \deg_{G'}(v)$. For a tree T, let $s(T)$ denote the number of tree-saturated vertices in T, and let $s_h(T)$, for $0 \leq h < n$, be the number of tree-saturated vertices in T at height (i.e. distance in tree T from root r to the vertex) not greater than h. The vertex adjacent to v on the path in T leading from v to root r will be called the *parent* $p(v)$, while the edge connecting these two vertices will be called the *parent edge* $pe(v) = \{p(v), v\}$. For convenience of notation, we will occasionally augment the edge set of tree T by the fictional parent edge of the root $pe(r)$.

Consider the following partial order on rooted spanning trees in graph G'. We will say that $T_a < T_b$ if one of the following conditions is fulfilled.

1. $s(T_a) < s(T_b)$,
2. $s(T_a) = s(T_b)$, and for some h, $0 \leq h < n$, we have $\forall_{0 \leq l < h}\ s_l(T_a) = s_l(T_b)$ and $s_h(T_a) > s_h(T_b)$.

Now, multigraph H' can be determined by the following algorithm:

Algorithm 1: Computing multigraph H'

1. Let T be a minimal spanning tree in G' with respect to order $(<)$.
2. Let $\mathcal{S}$ be a subgraph in $G' \setminus T$, whose connected components are stars, such that for each $v \in V$ either v is tree-saturated in T or $\deg_{\mathcal{S}}(v) > 0$.
3. Find a submultigraph $H' \subseteq \mathcal{S} \cup 2T$ fulfilling properties A′ and B′, such that $|E(H')| \leq 2n$, and return it as output.

We have to show that all the steps of the above algorithm are well defined. Step (1) requires no comment. For step (2), notice that graph $\mathcal{S}$ is well defined because any graph admits a subgraph which is a set of stars, touching all non-isolated vertices; for graph $G' \setminus T$, the only isolated vertices are those which were tree-saturated in T. Before we proceed to describe the procedure to be used in step (3), we first need to show some structural lemmata.

4.1 Properties of Graph $\mathcal{S}$

Let S be an arbitrary star which is a connected component of $\mathcal{S}$ with vertex set $\{v_1, \ldots, v_k\}$. Whenever $k \geq 3$ (i.e. when S is not a single edge) we assume that v_1 is the center of the star. Note that none of the vertices v_i can be tree-saturated in T.

Lemma 1 *Assume that $k \geq 3$. Then for any v_i, $1 < i \leq k$, if $p(v_i)$ is tree-saturated in T, then v_i lies on the path in T from v_1 to root r.*

Proof Suppose that, to the contrary, $p(v_i)$ is tree-saturated in T and, that v_i does not lie on the path from v_1 to root r. We have two cases.

1. Center v_1 lies on the path from v_i to root r (Fig. 2a). Then, consider the tree $T^* = T \cup \{v_1, v_i\} \setminus pe(v_i)$. T^* is a valid spanning tree in G'; moreover, $s(T^*) < s(T)$ (since vertex $p(v_i)$ is no longer tree-saturated in T^* and no new vertices are tree-saturated), a contradiction with the minimality of T.

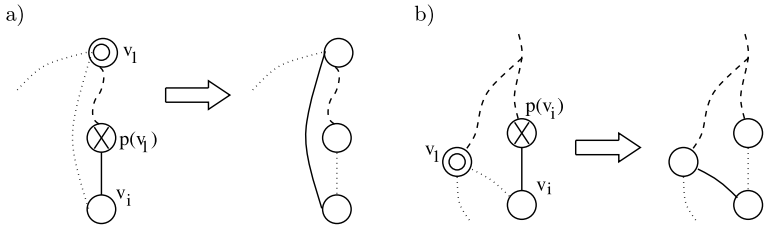

Fig. 2 Arrangements of a star center with respect to saturated vertices (*solid edges* belong to the tree, *dashed edges* belong to a star; a *cross* denotes a saturated vertex, a *double circle* denotes the star center)

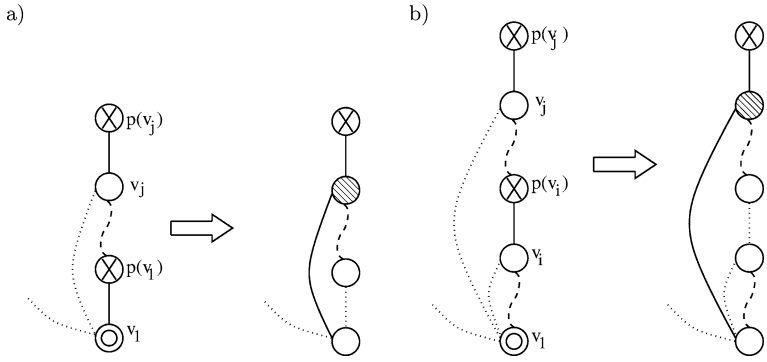

Fig. 3 Arrangements of two saturated parents for a star (*solid edges* belong to the tree, *dashed edges* belong to a star; a *cross* denotes a saturated vertex, a *shaded circle* denotes a possible saturated vertex, while a *double circle* denotes the star center)

2. Vertices v_1 and v_i lie on paths to root r, neither of which is contained in the other (Fig. 2b). Then we obtain a contradiction by defining tree T^* in the same way as in the previously considered case. □

Lemma 2 *Assume that $k \geq 3$. Then there can exist at most one v_i, $1 \leq i \leq k$, such that $p(v_i)$ is tree-saturated in T.*

Proof Suppose that, to the contrary, there are two vertices v_i and v_j, $i \neq j$, such that $p(v_i)$ and $p(v_j)$ are tree-saturated (note that we do not necessarily assume that $p(v_i) \neq p(v_j)$). Once again, we need to consider two cases.

1. The parent $p(v_1)$ of the center of the star is tree-saturated (i.e. we can put $i = 1$; Fig. 3a). Then by Lemma 1, v_j lies on the path in T leading from v_1 to root r. Then, consider the tree $T^* = T \cup \{v_1, v_j\} \setminus pe(v_1)$. T^* is a valid spanning tree in G'. We now have $s(T^*) \leq s(T)$, since vertex $p(v_1)$ is no longer tree-saturated in T^*, vertex v_j may have become tree-saturated, while the tree-saturation of the other vertices does not change. However, if $s(T^*) = s(T)$, then v_j must be tree-saturated in T^*, so denoting by h the height of v_j in T (equivalently T^*) we have $\forall_{0 \leq l < h}\ s_l(T^*) = s_l(T)$ and $s_h(T^*) > s_h(T)$. Thus, we obtain that $T^* < T$, a contradiction with the minimality of T.

2. The parents $p(v_i)$ and $p(v_j)$, for some $i, j > 1$, are tree-saturated (Fig. 3b). Then by Lemma 1, both v_i and v_j lie on the path in T leading from v_1 to root r; without loss of generality we can assume that v_j is closer than v_i to root r. Then, consider the tree $T^* = T \cup \{v_1, v_j\} \setminus pe(v_i)$. T^* is a valid spanning tree in G', and using the same arguments as in the previous case we obtain that $T^* < T$, a contradiction with the minimality of T. □

Lemma 3 *Let S be a two-vertex star component consisting of vertices v_1, v_2, at heights h_1 and h_2 in tree T, respectively. If $p(v_1)$ and $p(v_2)$ are both tree-saturated, then $|h_1 - h_2| \leq 1$.*

Proof Without loss of generality let $h_1 \leq h_2$. Suppose that $p(v_1)$ and $p(v_2)$ are both tree-saturated, and that $h_1 < h_2 - 1$. Similarly to the proof of Lemma 1, let $T^* = T \cup \{v_1, v_2\} \setminus pe(v_2)$. It is clear that in T^* vertex $p(v_2)$ (at level $h_2 - 1$) is no longer tree-saturated, vertex v_1 (at level $h_1 < h_2 - 1$) may possibly become tree-saturated, while the tree-saturation of the remaining vertices does not change. Hence $T^* < T$, a contradiction with the minimality of T. □

4.2 Construction of Multigraph H'

We now describe the routine used in Step (3) of Algorithm 1 to construct submultigraph $H' \subseteq \mathcal{S} \cup 2T$ by iteratively adding edges. As it will be proven by the next lemma, the construction of H' preserves Properties A′ and B′. Taking into account Lemma 3, we can write $E(\mathcal{S}) = E(\mathcal{S})_+ \cup \bigcup_{h=1}^{n} E(\mathcal{S})_{h,h} \cup \bigcup_{h=1}^{n} E(\mathcal{S})_{h,h-1}$, where $E(\mathcal{S})_+$ denotes the set of edges belonging to stars of more than 2 vertices, $E(\mathcal{S})_{h,h}$ is the set of edges of two-vertex stars with both vertices at height h in tree T, and finally $E(\mathcal{S})_{h,h-1}$ is the set of edges of two-vertex stars with one vertex at height h and the other at height $h-1$ in tree T. Likewise, since T is a tree, we can write $E(2T) = \bigcup_{h=1}^{n} E(2T)_{h,h-1}$, where $E(2T)_{h,h-1}$ contains edges connecting a vertex at height h with a vertex at height $h-1$ in tree T.

We start by putting $E(H') = E(\mathcal{S})_+$. Then, for all levels $h \in (n, n-1, \ldots, 1)$, considered in decreasing order, we choose which edges from $E(\mathcal{S})_{h,h}$, $E(\mathcal{S})_{h,h-1}$ and $E(2T)_{h,h-1}$ to add to $E(H')$:

(a) Add all edges from $E(\mathcal{S})_{h,h} \cup E(\mathcal{S})_{h,h-1}$ to $E(H')$.
(b) For each vertex v at level h, add to $E(H')$ a subset of the two copies of edge $pe(v)$ from $E(2T)_{h,h-1}$ so that the degree of v is even in H'. When an even number of edges is required, 2 edges should be used if v is tree-saturated or $p(v)$ is tree-saturated, and 0 edges should be used otherwise.
(c) Successively consider all edges $\{v_1, v_2\} \in E(\mathcal{S})_{h,h} \cup E(\mathcal{S})_{h,h-1}$ which were added in step (a). If both vertex v_1 and vertex v_2 are currently of even degree in H', and both $p(v_1)$ and $p(v_2)$ are tree-saturated, then remove from $E(H')$ edge $\{v_1, v_2\}$.
(d) For each vertex v at level h affected by changes in step (c), remove from $E(H')$ one of the two copies of edge $pe(v)$ so that the degree of v is even in H'.

To show that graph H' fulfills properties A′ and B′ it suffices to prove the following lemma.

Lemma 4 *For each vertex $v \in V$, $\deg_{H'}(v)$ is even. Moreover, if $\deg_{H'_1}(v) = 0$, then v is tree-saturated and $\deg_{H'_2}(v) = \deg_{2T}(v)$.*

Proof For each height h, the degree of all vertices at height h in H' is always even after completion of steps (b) and (d) for height h, and is never changed afterwards. (For the root r at height $h = 0$, the degree must be even by the Handshaking lemma). Consequently, the degree of all vertices in H' is even.

If $\deg_{H'_1}(v) = 0$, then since all edges of stars with more than two vertices appear in $E(H'_1)$, v can only be adjacent to a 2-vertex star or tree-saturated in T. The former case is impossible, since by definition of steps (c) and (d) either the edge of the 2-vertex star, or $pe(v)$ belongs to $E(H'_1)$. In the latter case, the definition of steps (a)-(d) is such that each edge of tree T adjacent to a tree-saturated vertex appears either in 1 or 2 copies in $E(H') = E(H'_1) \cup E(H'_2)$, hence if $\deg_{H'_1}(v) = 0$, then clearly $\deg_{H'_2}(v) = \deg_{2T}(v)$. □

The relation $|E(H')| \leq 2n$ will now be shown using a local cost-based argument. First, we will assign costs c to vertices and edges of T as follows.

1. For each edge $e \in E(T)$, cost $c(e) \in \{0, 1, 2\}$ is set as the number of times edge e appears in multigraph H'.
2. For each vertex $v \in V$, we set cost $c(v) \in \{0, 1\}$ by considering the following cases:
   (a) if v is tree-saturated in T, then $c(v) = 0$,
   (b) if v belongs to a 2-vertex star from $\mathcal{S}$, then $c(v) = 1$ if $pe(v)$ appears in H' at most once, and $c(v) = 0$ otherwise,
   (c) if v belongs to a star from $\mathcal{S}$ having at least 3 vertices, then $c(v) = 0$ if $p(v)$ is tree-saturated, and $c(v) = 1$ otherwise.

Lemma 5 *For the cost assignment c we have $\sum_{x \in E(T) \cup V} c(x) \geq |E(H')| = |E(H' \cap 2T)| + |E(H' \cap \mathcal{S})|$.*

Proof Indeed, we have $\sum_{x \in E(T)} c(x) = |E(H' \cap 2T)|$ by definition of the costs of edges. Next, notice that $|E(H' \cap \mathcal{S})|$ can be redistributed over the costs of vertices as follows (note that the stars are vertex-disjoint). For a k-vertex star, with $k > 2$, we pay for $k - 1$ edges by assigning a cost of 1 to all vertices of the star, except perhaps one vertex v for which $p(v)$ is tree-saturated (by Lemma 2 there can be at most one such vertex for each star). For an edge of a 2-vertex star which appears in H', we pay for the edge using the cost assigned to at least one of its end-vertices having cost 1; an end-vertex with such cost must exist by the construction of steps (c)–(d) of the algorithm. □

Lemma 6 *For each vertex $v \in V$, $c(v) + c(pe(v)) \leq 2$.*

Proof Since $c(v) \leq 1$, the claim is clearly true if $c(pe(v)) < 2$. If $c(pe(v)) = 2$, then edge $pe(v)$ appears twice in H', hence by the construction of steps (a)–(d) of the algorithm it is clear that vertex v or vertex $p(v)$ is tree-saturated. In either case, we have $c(v) = 0$ by the definition of costs, which completes the proof. □

Combining Lemmata 5 and 6, summing over all vertices, immediately gives the sought bound, $|E(H')| \leq 2n$, which completes the proof of Theorem 2.

5 Construction of the Labeling

Recalling from Sect. 3 that $|E(H)| = |E(H')| + 2(n-1)$, we can rephrase Theorem 2 in the original language of graph exploration.

Theorem 3 *For any graph G of size n there exists a port labeling leading to an exploration period $\pi \leq 4n - 2$.*

It is natural to ask about the runtime of the procedure required to obtain a labeling with such an exploration period, and about the tightness of the obtained bound; we address these questions in the following subsections.

5.1 Runtime of the Labeling Procedure

Whereas the construction of the appropriate cycle can always be performed using Algorithm 1 (in finite time), this does not necessarily mean that a solution can be found in polynomial time. The problem consists in computing an appropriate spanning tree, minimal in the sense of order $(<)$, in step (1). In general, finding a spanning tree having a minimum number of saturated vertices is already *NP*-hard. (The proof of this observation proceeds by reduction from the problem of finding a Hamiltonian path in a 3-regular graph: a 3-regular graph has a spanning tree without saturated vertices if and only if it admits a Hamiltonian path.)

In order to obtain polynomial time complexity, we apply a slight modification of step (1): instead of requiring tree T to be minimal, we only require that Lemmata 1, 2 and 3 hold for this tree. If there appears a contradiction in the proof of either lemmata, we replace tree T by tree T^*, using the modifications shown in Fig. 2 for Lemma 1. For Lemma 2, we use the replacement from Fig. 3a, while instead of the replacement from Fig. 3b, define tree T^* by putting $T^* = T \cup \{v_1, v_j\} \cup \{v_1, v_i\} \setminus (pe(v_1) \cup pe(v_i))$ as shown in Fig. 4. For Lemma 3, we simply perform the operations described in the proof. We make the following observations:

- The modification for Lemma 1 decreases the value of parameter $s(T)$.
- The modifications for Lemma 2 and Lemma 3 either decrease the value of parameter $s(T)$, or decrease the value of parameter $\sum_{0 \leq l < n} (l \cdot s_l(T))$.

The initial value of parameter $s(T)$ is at most n; clearly, the modification process stops when $s(T) = 0$. Next, since we have $\sum_{0 \leq l < n} (l \cdot s_l(T)) < n^2$, then this parameter may only be decreased less than n^2 times for a given value of $s(T)$. Overall, the total number of tree modifications is less than n^3. This guarantees polynomial runtime of the whole algorithm.

Theorem 4 *There exists a polynomial time algorithm which, given a graph G, determines a port labeling leading to an exploration period $\pi \leq 4n - 2$.*

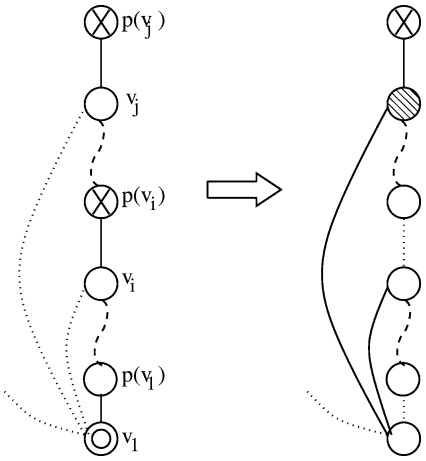

Fig. 4 Alternative replacement for the case from Fig. 3b, used to guarantee polynomial runtime. Tree T^* is defined as $T^* = T \cup \{v_1, v_j\} \cup \{v_1, v_i\} \setminus (pe(v_1) \cup pe(v_i))$

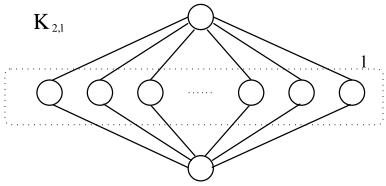

Fig. 5 Examples of graphs $G = K_{2,l}$

5.2 Tightness of the Bound

In this section, we show that there actually exist instances for which the length of the computed exploration cycle by means of our approach is $4n - O(1)$. Consider the following property of multigraph H_2, obtained by a slight modification of property C, which is also satisfied by our construction.

D. *H_2 spans vertex set V, i.e. for all $v \in V$, $\deg_{H_2}(v) > 0$.*

We make the following claim, which is partially complementary to Theorem 1.

Theorem 5 *For all values of $n \geq 3$, there exists a graph G of order n, such that any exploration multigraph $H \subseteq 2G$ fulfilling properties A, B, and D, has $|E(H)| \geq 4n - 8$ edges.*

Proof Consider the complete bipartite graph $G = K_{2,l}$ on $n = l + 2$ vertices, for any $l \geq 1$ (see Fig. 5). By property D, for each of the l vertices $v \in V$ such that $\deg_G(v) = 2$, we have $\deg_{H_2}(v) > 0$, hence $\deg_{H_1}(v) \leq 1$. Taking into account property A we obtain $\deg_{H_1}(v) = 0$, and so by property B, v is saturated in H_2. This means that $H = 2G$, and so $|E(H)| = 2|E(G)| = 4n - 8$. □

6 Conclusion

We have considered the problem of periodic graph exploration by means of an oblivious robot. The robot must explore all the nodes of a connected anonymous graph by

following local information available at each vertex, properly initialized. The problem requires the robot to visit every vertex infinitely many times in a periodic manner. This can be considered as a relaxation of the Hamiltonian cycle problem since it is not required that the exploration cycle touches each vertex exactly once but only that all the vertices are touched.

We have shown that by applying a special construction of a spanning tree, dedicated to the considered exploration problem, it is possible to find in polynomial time an exploration cycle of length at most $4n - 2$. Moreover, we have shown that the obtained bound on the exploration period is sometimes tight up to an additive factor. The best known existing lower bound on the length of an exploration cycle is within an additive factor of $2.8n$ [5]. However, obtaining cycles significantly shorter than $4n$ would require some completely new insight; in particular, the construction would need to avoid the condition imposed on the double-edge subgraph of multigraph H (property D in Theorem 5).

References

1. Budach, L.: Automata and labyrinths. Math. Nachr. **86**, 195–282 (1978)
2. Cohen, R., Fraigniaud, P., Ilcinkas, D., Korman, A., Peleg, D.: Label-guided graph exploration by a finite automaton. ACM Trans. Algorithms **4**(4), 1–18 (2008)
3. Cohen, R., Fraigniaud, P., Ilcinkas, D., Korman, A., Peleg, D.: Labeling schemes for tree representation. Algorithmica **53**(1), 1–15 (2009)
4. Cook, S., Rackoff, C.: Space lower bounds for maze threadability on restricted machines. SIAM J. Comput. **9**(3), 636–652 (1980)
5. Czyzowicz, J., Dobrev, S., Gąsieniec, L., Ilcinkas, D., Jansson, J., Klasing, R., Lignos, Y., Martin, R.A., Sadakane, K., Sung, W.K.: More efficient periodic traversal in anonymous undirected graphs. In: Proceedings of the 16th Colloquium on Structural Information and Communication Complexity (SIROCCO). LNCS, vol. 5869, pp. 167–181 (2009)
6. Dobrev, S., Jansson, J., Sadakane, K., Sung, W.K.: Finding short right-hand-on-the-wall walks in graphs. In: Proceedings of the 12th Colloquium on Structural Information and Communication Complexity (SIROCCO). LNCS, vol. 3499, pp. 127–139 (2005)
7. Fraigniaud, P., Ilcinkas, D., Peer, G., Pelc, A., Peleg, D.: Graph exploration by a finite automaton. Theor. Comput. Sci. **345**(2–3), 331–344 (2005)
8. Fraigniaud, P., Ilcinkas, D., Pelc, A.: Impact of memory size on graph exploration capability. Discrete Appl. Math. **156**(12), 2310–2319 (2008)
9. Gąsieniec, L., Klasing, R., Martin, R.A., Navarra, A., Zhang, X.: Fast periodic graph exploration with constant memory. J. Comput. Syst. Sci. **74**(5), 802–822 (2008)
10. Gąsieniec, L., Radzik, T.: Memory efficient anonymous graph exploration. In: Proceedings of the 34th International Workshop on Graph-Theoretic Concepts in Computer Science (WG). LNCS, vol. 5344, pp. 14–29 (2008)
11. Ilcinkas, D.: Setting port numbers for fast graph exploration. Theor. Comput. Sci. **401**(1–3), 236–242 (2008)
12. Kosowski, A., Navarra, A.: Graph decomposition for improving memoryless periodic exploration. In: Proceedings of the 34th International Symposium on Mathematical Foundations of Computer Science (MFCS). LNCS, vol. 5734, pp. 501–512 (2009)
13. Reingold, O.: Undirected st-connectivity in log-space. In: Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC), pp. 376–385 (2005)
14. Rollik, H.: Automaten in planaren graphen. Acta Inform. **13**, 287–298 (1980)