Network Verification via Routing Table Queries[*]

Evangelos Bampas[1], Davide Bilò[2], Guido Drovandi[3], Luciano Gualà[4], Ralf Klasing[1], and Guido Proietti[3,5]

[1] LaBRI, CNRS / University of Bordeaux, Bordeaux, France[**]
[2] Dipartimento di Teorie e Ricerche dei Sistemi Culturali, University of Sassari, Italy
[3] Istituto di Analisi dei Sistemi ed Informatica, CNR, 00185 Rome, Italy
[4] Dipartimento di Matematica, University of Tor Vergata, Rome, Italy
[5] Dipartimento di Ingegneria e Scienze dell'Informazione e Matematica, University of L'Aquila, Italy

Abstract. We address the problem of *verifying a network*, namely that of establishing the accuracy of a high-level description of its physical topology, by making as few measurements as possible on its nodes. This task can be formalized as an optimization problem that, given a graph $G = (V, E)$, and a *query model* specifying the information returned by a query at a node, asks for finding a *minimum-size* subset of nodes of G to be queried so as to univocally identify G. This problem has been studied w.r.t. different query models assuming that a node had some *global* knowledge about the network. Here, we propose a new query model based on the *local* knowledge a node instead usually has. Quite naturally, we assume that a query at a given node returns the associated *routing table*, i.e., a set of entries which provides, for each destination node, a corresponding (set of) first-hop node(s) along an underlying shortest path. First, we show that any network of n nodes needs $\Omega(\log \log n)$ queries to be verified, and, even worse, that if the network has a constant diameter, then $\Omega(\log n)$ queries are needed. Then, we prove that there is no $o(\log n)$-approximation algorithm for the problem in the class of all networks of diameter 2, unless $\mathsf{P} = \mathsf{NP}$, and we show this is tight. On the positive side, we give exact linear-time algorithms for the basic network topologies of paths, trees, and cycles of even length. Interestingly, for a path and a cycle of even length, the size of the optimal query set is about

[*] A preliminary version of this work was presented at the *18th International Colloquium on Structural Information and Communication Complexity (SIROCCO'11)*, June 26-29, 2011, Gdansk, Poland, Lecture Notes in Computer Science, vol. 6796, Springer 2011, pp. 270–281. This work was partially supported by research projects PRIN 2008 "COGENT" (COmputational and GamE-theoretic aspects of uncoordinated NeTworks) and PRIN 2010 "ARS TechnoMedia" (Algorithms for Techno-Mediated Social Networks), funded by the Italian Ministry of Education, University, and Research. Part of this work was done while the second author was visiting LaBRI-Bordeaux.

[**] Additional support by the ANR projects ALADDIN, DISPLEXITY and IDEA and the INRIA project CEPAGE.

a half and a third of n, respectively, while for a tree we show that such a size eventually depends on its structure, but is at least $\lceil \frac{n}{3} \rceil$.

Keywords: Network verification, Graph/Network topology, Computational complexity, Approximation algorithms.

1 Introduction

There is a growing interest in networks which are built and maintained by decentralized processes. In such a setting, it is natural to consider the problem of discovering the map of an unknown (in terms of edges) network, or to verify whether a given map is accurate, i.e., to check whether the edges of the map are exactly those of the underlying network. A common approach to discover or to verify a map is to make some local measurement on a selected subset of nodes that – once collected – can be used to derive information about the whole network (see for instance [6, 9]). A measurement on a node is usually costly, so it is natural to try to make as few measurements as possible.

These two tasks – that of *discovering* a map and that of *verifying* a given map – have been formalized as optimization problems and have been studied in several papers. The idea is to model the network as a graph $G = (V, E)$, while a measurement at a given node can be seen as a unit-cost *query* returning some piece of information about G. In the *network discovery* problem, we know V but not E, and so we want to design an on-line algorithm that selects a minimum-size subset of nodes $Q \subseteq V$ to be queried that allows to precisely map the entire graph, i.e., to settle all the edges and all the non-edges of G. The quality of the algorithm is measured by its competitive ratio, i.e., the ratio between the number of queries made by the algorithm and the minimum number of queries which would be sufficient to discover the graph. On the other hand, the *network verification* problem, which is of interest for our paper, is the off-line version of the problem, and so we are given a graph $G = (V, E)$, and we want to compute a *minimum* number of queries sufficient to discover G (if E was unknown). This latter problem has an interesting application counterpart, since it models the activity of verifying the accuracy of a given map associated with an underlying real network (on which the queries are actually done).

In the literature, two main query models have been studied. In the *all-shortest-paths* query model, a query of a node q returns the subgraph of G consisting of the union of all shortest paths between q and every other node $v \in V$. A weaker notion of query is used in the *all-distances* query model, in which a query to a node q returns all the distances in G from q to every other node $v \in V$. Notice that both models inherently require global knowledge/information about the network, hence a central problem for these query models is whether/how the information can be obtained locally (without preprocessing of the network). In this paper, we propose a query model that uses only *local* knowledge/information about the network. Quite naturally, we assume that a query at a given node q returns the associated *routing table*, namely a set of entries which provides, for each destination node, a corresponding (set of) first-hop node(s) along an underlying shortest path. In the rest of the paper, this will be referred to as the *routing-table query model*.

Previous work. It turns out that the verification problem with the *all-shortest-paths* query model is equivalent to the problem of placing landmarks on a graph [17]. In this problem, we want to place landmarks on a subset of the nodes in such a way that every node is uniquely identified by the distance vector to the landmarks, and the minimum number of landmarks to be placed is called the *metric dimension* of a graph [14]. The problem has been shown to be NP-hard in [8]. An explicit reduction from 3-SAT is given in [17] which also provides an $O(\log n)$-approximation algorithm (n is the number of nodes) and an exact polynomial-time algorithm for trees. Subsequently, in [1], the authors prove that the problem is not $o(\log n)$ approximable, unless P=NP, showing thus that the algorithm in [17] is the best possible in an asymptotic sense. We finally mention that in [2] the authors studied the related problem of monitoring link failures (i.e., verifying only the graph edges) in a very similar query model.

As far as the *all-distances* query model is concerned, the verification problem has been studied in [1], where the NP-hardness is proved and an algorithm with $O(\log n)$-approximation guarantee is provided. Other results in [1] include exact polynomial-time algorithms for trees, cycles and hypercubes.

Concerning the network discovery problem, this received attention in several papers and in both query models. More precisely, for the *all-distances* query model, in [1] the authors have shown an $\Omega(\log n)$ lower bound on the competitive ratio of any *randomized* algorithm, and an $\Omega(\sqrt{n})$ lower bound on the competitive ratio of any *deterministic* online algorithm. Moreover, they also provided a randomized $O(\sqrt{n \log n})$ competitive algorithm. On the other hand, in the *all-shortest-paths* query model, in [1] the authors have shown a $3-\epsilon$ lower bound on the competitive ratio of any *deterministic* online algorithm, for any $\epsilon > 0$, and they provided a randomized $O(\sqrt{n \log n})$ competitive algorithm. This was then improved in [20], where the authors provided an $O(\log^2 n)$-competitive Monte Carlo randomized algorithm. Still for the same query model, we finally mention that in [3] the authors studied how to discover several graph properties, while in [7] the authors focused on the *approximate* discovery of Erdös-Rényi random graphs.

Besides the two traditional query models we widely discussed above, it is worth mentioning that the network discovery and verification problems were also analyzed in the *edge-counting* query model, where a query at a set of vertices returns the number of edges in the corresponding induced subgraph. In this model, the information theoretic lower bound for the query complexity of discovering a graph is $\Omega\left(\frac{m \log n^2/m}{\log m}\right)$, and in [4, 18] a polynomial time algorithm was provided with a query complexity matching such a lower bound (which improved a previous result in [5]). Finally, concerning the graph verification, in [19] the authors gave a Monte Carlo randomized algorithm with error ϵ for using $O(\log 1/\epsilon)$ queries.

Our results. Throughout the paper, we focus on the verification problem w.r.t the routing-table query model. We first show a lower bound of $\Omega(\log\log n)$ on the minimum number of queries needed to verify any graph with n nodes. This is in contrast to the previous two query models for which certain classes of graphs can be verified with a constant number of queries, like paths and cycles. Our proof also implies a lower bound of $\Omega(n)$ on the number of queries needed to verify a path or a cycle. So, one may wonder whether every graph needs a linear number of queries to be verified. We provide a negative answer to this question by exhibiting a class of graphs (of diameter 2) that can be verified with $O(\log n)$ queries. On the other hand, we show that this number of queries is asymptotically optimal, since we prove that for graphs of constant diameter $\Omega(\log n)$ queries are actually needed.

We then analyze the computational complexity of the problem. In this respect, although it remains open for general input graphs to establish whether the problem is in NPO, we are able to provide an $O(\log n)$-approximation algorithm to verify graphs of diameter 2. Moreover, we also show that this bound is asymptotically tight, unless $\mathsf{P} = \mathsf{NP}$. On the positive side, we provide exact linear-time algorithms to verify paths, trees and cycles of even length. In terms of number of queries, these algorithms show that for paths and cycles of even length, the size of the optimal query set is about a half and a third of n, respectively. On the other hand, for trees such a size eventually depends on the tree structure, but we show that it is at least $\lceil \frac{n}{3} \rceil$ (and we also show this is tight). Our result for trees is based on a characterization of a solution that can be used to reduce the problem to that of computing a minimum vertex cover of a certain class of graphs (for which a vertex cover can be found in polynomial time). The algorithm for cycles of even length shows a counterintuitive fact about the routing-table query model. Indeed, while a query in our model seems to obtain only local information about the graph, we show in the case of the cycle that the symmetry can be used to infer some knowledge about edges and non-edges that are far from queried nodes.

The paper is organized as follows. After giving some basic definitions in Section 2, we formally introduce our query model in Section 3. Section 4 is devoted to the lower bound of $\Omega(\log\log n)$ for any graph with n nodes, and of $\Omega(\log n)$ for graphs of constant diameter. The verification of graphs of diameter 2 is presented in Section 5, while in Section 6, we describe the exact linear-time algorithms for certain classical topologies (paths, trees, and cycles of even length), and we characterize the size of the corresponding optimal query sets. Finally, Section 7 concludes the paper, by providing a discussion about an extension of our query model in which the routing tables also contain information about distances to all the nodes, and by proposing open problems and directions for future research.

2 Basic Definitions

Let $G = (V, E)$ be an undirected (simple, connected) graph with n vertices. We assume that vertices are distinguishable, i.e., they have different identifiers. Given two distinct vertices $u, v \in V$, we say that (u, v) is an *edge* (resp., a *non-edge*) of G if $(u, v) \in E$ (resp., $(u, v) \notin E$).[1] For a graph G, we will also denote by $V(G)$ and $E(G)$ its set of vertices and its set of edges, respectively.

For every vertex $v \in V$, let $N_G(v) = \{u \mid u \in V \setminus \{v\}, (u, v) \in E\}$ and let $N_G[v] = N_G(v) \cup \{v\}$. The *maximum degree* of G is equal to $\max_{v\in V} |N_G(v)|$. Let $U \subseteq V$ be a set of vertices. We denote by $G[U]$ the graph with $V(G[U]) = U$ and $E(G[U]) = \{(u, v) \mid u, v \in U, (u, v) \in E\}$. Let $F \subseteq \{(u, v) \mid u, v \in V, u \neq v\}$. We denote by $G + F$ (resp., $G - F$) the graph on V with edge set $E \cup F$ (resp., $E \setminus F$). When $F = \{e\}$ we will denote $G + \{e\}$ (resp., $G - \{e\}$) by $G + e$ (resp., $G - e$). For two graphs G_1 and G_2, we denote by $G_1 \cup G_2$ the graph with $V(G_1 \cup G_2) = V(G_1) \cup V(G_2)$ and $E(G_1 \cup G_2) = E(G_1) \cup E(G_2)$. We denote by $d_G(u, v)$ the *distance* in G from u to v. The *diameter* of G is equal to $\max_{u,v\in V} d_G(u, v)$.

Let $\texttt{query}_G$ be a *query model*, that is, a function which associates with each vertex of G a set of information about G. For a set $Q \subseteq V$, we denote by $\texttt{query}_G(Q) = \{\texttt{query}_G(q) \mid q \in Q\}$. We say that Q *verifies* edge (resp., non-edge) (u, v) of G iff for every graph $G' = (V, E')$ with $\texttt{query}_G(Q) = \texttt{query}_{G'}(Q)$, we have that $(u, v) \in E'$ (resp., $(u, v) \notin E'$). Finally, we say that Q *verifies* G iff for every $G' = (V, E')$ with $E \neq E'$, we have that $\texttt{query}_G(Q) \neq \texttt{query}_{G'}(Q)$. This implies that Q verifies G iff Q verifies every edge and every non-edge of G. Clearly, we have that if $Q \subseteq V$ verifies G, then for every $q \in V$, $Q \cup \{q\}$ verifies G. As a consequence of the definitions, we therefore have two ways of actually proving that a set of queries Q verifies a graph G: On the one hand, we can prove it directly by applying the definition (i.e., by showing that for any graph $G' = (V, E')$ with $E' \neq E$, we have that $\texttt{query}_G(Q) \neq \texttt{query}_{G'}(Q)$). On the other hand, we can prove it by showing that for any pair of vertices $u, v \in V$, the set of information returned by querying Q in G is sufficient to logically infer the existence (resp., non existence) of (u, v) in G (this is equivalent to saying that by querying Q on any graph $G' = (V, E')$ with $E' \neq E$, we would get a different set of information, and thus Q verifies G).

Given an undirected graph $G = (V, E)$, the *Network Verification Problem w.r.t. query model* $\texttt{query}_G$ is the optimization problem of finding a *minimum-size* subset $Q \subseteq V$ that verifies G w.r.t. query model $\texttt{query}_G$.

3 The Routing-Table Query Model

For a given vertex $q \in V$, we denote by $\texttt{table}_G(q)$ the *routing table* of q in G, i.e.,

$$\texttt{table}_G(q) = \Big\{\langle v, u\rangle \mid v, u \in V \setminus \{q\} \wedge (q, u) \in E \wedge d_G(q, v) = d_G(v, u) + 1\Big\}.$$

[1] Notice that, in contrast to standard notation, (u, v) is used to denote an undirected edge since $\{u, v\}$ will instead be used to denote the set containing nodes u and v.

A pair $\langle v, u\rangle \in \texttt{table}_G(q)$ means that there exists a shortest path from q to v whose first hop is vertex u. The *routing-table query model* is the model in which $\texttt{query}_G(q) = \texttt{table}_G(q)$, for every $q \in V$. In the rest of the paper, we will denote by $\texttt{T}^q_G(u) = \big\{v \in V \mid \langle v, u\rangle \in \texttt{table}_G(q)\big\}$. Clearly, for every $v \in V$ we have that $Q = V \setminus \{v\}$ verifies G w.r.t. the routing table query model, as any $q \in Q$ verifies all edges and non-edges of G of the form (q, u), for any $u \in V \setminus \{q\}$. Notice also that if G is a clique, then this is optimal.

The following fact is easy to prove:

Proposition 1. *Let q, u, v be such that $v \in \texttt{T}^q_G(u)$ and for every other $u' \neq u$, $v \notin \texttt{T}^q_G(u')$. Then, $(q, u) \in E$ and the shortest paths between q and v in G all pass through edge (q, u) and use only vertices in $\texttt{T}^q_G(u)$.*

As a consequence of the above fact, we are now able to give some easy-to-check conditions which are sufficient to verify a given edge (respectively, non-edge) w.r.t. the routing-table query model. Unfortunately, these conditions are not necessary, and so it remains open to establish whether the problem is in NPO.

For any distinct vertices $u, v \in V$, the following hold:

Proposition 2. *Let q be such that $\texttt{T}^q_G(u) = \{u, v\}$. Then, $(u, v) \in E$ and $\{q\}$ verifies the edge (u, v).*

Proposition 3. *Let q be a neighbor of u and let q' be a neighbor of v, respectively. If $\texttt{T}^{q'}_G(v) \cap \texttt{T}^q_G(u) = \{u, v\}$, then $(u, v) \in E$ and $\{q, q'\}$ verifies the edge (u, v).*

Proposition 4. *Let $q \in V \setminus \{u, v\}$ be such that $(q, u) \in E$, $(q, v) \notin E$. If $v \notin \texttt{T}^q_G(u)$, then $(u, v) \notin E$ and $\{q\}$ verifies the non-edge (u, v).*

Proposition 5. *Let $q, q' \in V \setminus \{u, v\}$ be two distinct vertices such that $(q, q') \in E$. If there exists $w \in V$ such that $v \notin \texttt{T}^q_G(w)$, $u \in \texttt{T}^q_G(w)$, $v \in \texttt{T}^q_G(q')$, and $u \in \texttt{T}^{q'}_G(q)$, then $(u, v) \notin E$ and $\{q, q'\}$ verifies the non-edge (u, v).*

Proof. For the sake of contradiction, assume there exists a graph $G' = (V, E')$ satisfying the hypothesis of the claim such that $(u, v) \in E'$. This implies that $|d_{G'}(z, u) - d_{G'}(z, v)| \leq 1$ for every vertex $z \in V$. As $v \notin \texttt{T}^q_{G'}(w)$ and $u \in \texttt{T}^q_{G'}(w)$, we have that $d_{G'}(q, v) \leq d_{G'}(q, u)$. Moreover, as $u \in \texttt{T}^{q'}_{G'}(q)$ and $v \in \texttt{T}^q_{G'}(q')$, we have that $d_{G'}(q', u) = d_{G'}(q, u) + 1$ and $d_{G'}(q', v) = d_{G'}(q, v) - 1$. As a consequence, $d_{G'}(q', v) = d_{G'}(q, v) - 1 \leq d_{G'}(q, u) - 1 = d_{G'}(q', u) - 2$, which implies $|d_{G'}(q', u) - d_{G'}(q', v)| \geq 2$, contradicting the fact that $|d_{G'}(q', u) - d_{G'}(q', v)| \leq 1$ and thus the assumption. This completes the proof. $\square$

Before ending this section, we provide some connections between the routing-table query model and the all-shortest-paths query model, which will be useful in the following sections. First, we recall the formal definition of the all-shortest-paths query model. For two given vertices $u, v \in V$, let $\Pi_G(u, v)$ denote the graph obtained by the union of all shortest paths in G between u and v. For a given vertex $q \in V$, we denote by $\texttt{asp}_G(q) = \bigcup_{u \in V} \Pi_G(q, u)$. The *all-shortest-paths query model* is the model in which $\texttt{query}_G(q) = \texttt{asp}_G(q)$, for every $q \in V$.

Lemma 1 ([1]). *A set $Q \subseteq V$ verifies a graph $G = (V, E)$ w.r.t. the all-shortest-paths query model iff, for every $u, v \in V$, with $u \neq v$, there exists a vertex $q \in Q$ such that $|d_G(q, u) - d_G(q, v)| \geq 1$.*

As from $\texttt{asp}_G(q)$ we can easily construct $\texttt{table}_G(q)$, the routing-table query model is weaker than the all-shortest-paths query model. More formally:

Proposition 6. *If $Q \subseteq V$ verifies $G = (V, E)$ w.r.t. the routing-table query model, then Q verifies G w.r.t. the all-shortest-paths query model.*

In Section 6 we will prove that on several basic network topologies, the routing-table query model is actually *much* weaker than the all-shortest-paths query model (for instance, on paths it will require a query set of linear size, while in the all-shortest-paths query model one query is clearly enough). One may then wonder whether the routing-table query model always incurs a considerably larger cost in terms of query set size. The following proposition, which we will use also in the rest of the paper, shows that this is not the case for some class of graphs.

Proposition 7. *Let $G = (V, E)$ be a graph containing a vertex s which is adjacent to all other vertices of G. Let $Q \subseteq V$. If Q verifies G w.r.t. the all-shortest-paths query model, then $Q \cup \{s\}$ verifies G w.r.t. the routing-table query model.*

Proof. Let $u, v \in V, u \neq v$ and $u, v \notin Q$. As Q verifies G w.r.t. the all-shortest-paths query model, then Lemma 1 implies that $|d_G(q, u) - d_G(q, v)| \geq 1$, for some $q \in Q$. W.l.o.g., let $d_G(q, u) < d_G(q, v)$. Now, consider the routing-table query model.

After making the query at s, we *discover* that every vertex of G is at distance 1 from s (i.e., we are able to certify that the distance in G between any vertex $v \neq s$ and s is equal to 1 by using the information returned by the query at s, namely its adjacency to all vertices of G). Thus, the distance between any pair of distinct vertices of G can be either 1 or 2. Since $\langle u, u \rangle \in \texttt{table}_G(q)$ whilst $\langle v, v \rangle \notin \texttt{table}_G(q)$, we discover that $d_G(q, u) = 1$ whilst $d_G(q, v) = 2$. Therefore, (u, v) is an edge of G iff $\langle v, u \rangle \in \texttt{table}_G(q)$ (and thus, (u, v) is a non-edge of G iff $\langle v, u \rangle \notin \texttt{table}_G(q)$). Hence, $Q \cup \{s\}$ discovers (i.e., verifies) G w.r.t. the routing-table query model. □

4 Lower Bounds on the Size of Feasible Solutions

In this section, we show lower bounds on the minimum number of queries needed to verify any graph G of n vertices w.r.t. the routing-table query model, as well as improved (linear) lower bounds for paths and cycles. We also exhibit a class of graphs (of diameter 2) that can be verified with $O(\log n)$ queries, and show that this number of queries is asymptotically optimal in the class of all graphs of constant diameter.

We begin by showing that $\Omega(\log \log n)$ queries are necessary to verify a graph G of n vertices. First of all, notice that in [1], the authors proved that $\log_3 \Delta$

queries are necessary to verify a graph G of maximum degree equal to Δ w.r.t. the all-shortest-paths query model. Therefore, by Proposition 6, we have the following:

Corollary 1. *Let $G = (V, E)$ be a graph of maximum degree equal to Δ and let Q be a query set that verifies G w.r.t. the routing-table query model. Then $|Q| \geq \log_3 \Delta$.*

Besides that, we now show a lower bound of $\Omega\bigl(\frac{\log n}{\Delta}\bigr)$ on the minimum number of queries needed to verify a graph G with n vertices and maximum degree equal to Δ w.r.t. the routing-table query model, thus obtaining a lower bound of $\max\{\log_3 \Delta, \Omega\bigl(\frac{\log n}{\Delta}\bigr)\} = \Omega(\log \log n)$. For any $q, v \in V$, let

$$\texttt{group}^q_G(v) := \begin{cases} \bigl\{w \in V \mid v \in \texttt{T}^q_G(w)\bigr\} & \text{if } v \notin N_G[q]; \\ \emptyset & \text{otherwise.} \end{cases}$$

The lower bound of $\Omega\bigl(\frac{\log n}{\Delta}\bigr)$ hinges on the following necessary condition:

Proposition 8. *If $Q \subseteq V$ verifies $G = (V, E)$, then, for every $u, v \in V, u \neq v$, one of the following conditions is satisfied:*

(i) $u \in N_G[q]$ or $v \in N_G[q]$, for some $q \in Q$;
(ii) $\exists\, q \in Q$ such that $\texttt{group}^q_G(u) \neq \texttt{group}^q_G(v)$.

Proof. We prove the claim by contraposition. We assume that there exists a pair of vertices u, v which satisfies neither (i) nor (ii). We divide the proof into two cases.

In the first case, we assume that u and v are *twin* vertices, i.e., $N_G(u) = N_G(v)$. Then it follows from the fact that (i) is not satisfied that $u, v \notin Q$. Therefore, $d_G(q, u) = d_G(q, v)$ for every $q \in Q$. Thus, Proposition 6 and Lemma 1 imply that Q cannot verify G.

In the second case, we assume that u and v are not twin vertices. Consider the graph G' obtained from G by swapping the labels of u and v. Clearly, for every $q \in Q$, and for every vertex $x \in V, x \neq u, v$, $\langle x, w\rangle \in \texttt{table}_G(q)$ iff $\langle x, w\rangle \in \texttt{table}_{G'}(q)$, since $w \neq u, v$ as condition (i) does not hold. Moreover, by definition of G' and because (i) does not hold, $u \in \texttt{T}^q_G(w)$ iff $v \in \texttt{T}^q_{G'}(w)$ and $v \in \texttt{T}^q_G(w)$ iff $u \in \texttt{T}^q_{G'}(w)$, for every $q \in Q$. Since (ii) does not hold, then for every $q \in Q$, $u \in \texttt{T}^q_G(w)$ iff $v \in \texttt{T}^q_G(w)$. As a consequence, for every $q \in Q$, $u \in \texttt{T}^q_G(w)$ iff $u \in \texttt{T}^q_{G'}(w)$ and $v \in \texttt{T}^q_G(w)$ iff $v \in \texttt{T}^q_{G'}(w)$. Therefore, $\texttt{table}_G(q) = \texttt{table}_{G'}(q)$ for every $q \in Q$. Thus, Q cannot verify G. □

Moreover, we can prove the following:

Lemma 2. *Let $G = (V, E)$ be a graph with n vertices of maximum degree equal to Δ, and let Q be a query set that verifies G w.r.t. the routing-table query model. Then $|Q| \geq \Omega\bigl(\frac{\log n}{\Delta}\bigr)$.*

Proof. Let $Q = \{q_1, \ldots, q_h\}$ be a minimum cardinality set of queries that verifies G w.r.t. the routing-table query model and let $V' = V \setminus \bigcup_{q \in Q} N_G[q]$. Since G has maximum degree equal to Δ, we have that $|V'| \geq n - |Q|(\Delta + 1)$. Moreover, as $\texttt{group}^q_G(v) \subseteq N_G(q)$, for every $v \in V'$ and for every $q \in Q$, we have that $\texttt{group}^q_G(v)$ is an element of the power set $2^{N_G(q)}$. As a consequence, $\langle \texttt{group}^{q_1}_G(v), \ldots, \texttt{group}^{q_h}_G(v) \rangle$ is an element of the power set $2^{N_G(q_1)} \times \ldots \times 2^{N_G(q_h)}$. As the size of $2^{N_G(q_i)}$ is upper bounded by 2^Δ we have that the size of $2^{N_G(q_1)} \times \ldots \times 2^{N_G(q_h)}$ is upper bounded by $2^{|Q|\Delta}$. Since condition (ii) of Proposition 8 implies that $\langle \texttt{group}^{q_1}_G(v), \ldots, \texttt{group}^{q_h}_G(v) \rangle \neq \langle \texttt{group}^{q_1}_G(u), \ldots, \texttt{group}^{q_h}_G(u) \rangle$ for every two distinct vertices $u, v \in V'$, we have that

$$2^{|Q|\Delta} \geq |V'| \geq n - |Q|(\Delta + 1)$$

holds. Hence, $|Q| = \Omega\left(\frac{\log n}{\Delta}\right)$. □

By combining the lower bound in Corollary 1 with the one in Lemma 2 we obtain:

Theorem 1. *Let $G = (V, E)$ be a graph of n vertices and let Q be a query set that verifies G w.r.t. the routing-table query model. Then $|Q| = \Omega(\log \log n)$.*

We point out that a direct application of Proposition 8 implies linear lower bounds for paths and cycles (unlike in the all-shortest-paths query model, where a constant number of queries suffices). More formally:

Corollary 2. *Let $G = (V, E)$ be a graph of n vertices and let Q be a minimum cardinality set of queries that verifies G w.r.t. the routing-table query model. We have that*

1. $|Q| \geq \left\lfloor \frac{n}{4} \right\rfloor$ *if G is a path;*
2. $|Q| \geq \left\lfloor \frac{n}{8} \right\rfloor$ *if G is a cycle.*

Proof. For paths, by Proposition 8, at least one vertex of every subpath of four consecutive vertices has to be contained in Q. The proof for cycles can be found in Section 6 (see Proposition 13 for cycles of even length and Proposition 15 for cycles of odd length, respectively). □

In Section 6, we actually provide an improved (tight) lower bound for paths and cycles of even length.

In view of Corollary 2, one may wonder whether every graph needs a linear number of queries to be verified. We provide a negative answer to this question by exhibiting a class of graphs that can be verified with $O(\log n)$ queries. Consider any graph G' of n' vertices $u_0, \ldots, u_{n'-1}$. We build G as follows: G contains a copy of G' plus $1 + \lceil \log n' \rceil$ vertices $s, q_1, \ldots, q_{\lceil \log n' \rceil}$, with u_i adjacent to q_j iff the j-th bit of the binary representation of i is equal to 1, and with s adjacent to all vertices of G (but s). Let $Q = \{s, q_1, \ldots, q_{\lceil \log n' \rceil}\}$. We now argue that Q verifies G w.r.t. the all-shortest-path query model. Indeed, Q verifies all edges and non-edges incident to the vertices in Q. Moreover, for every $u_i, u_{i'}$ with $i \neq i'$, there exists at least one bit, say the j-th, in which the binary representation of i differs from the binary representation of i'. This implies that $|d_G(q_j, u_i) - d_G(q_j, u_{i'})| \geq 1$. Thus, Lemma 1 implies that Q verifies G w.r.t. the all-shortest-paths query model. Finally, as s is adjacent to all vertices of the graph, by Proposition 7 we have that Q verifies G w.r.t the routing-table query model.

We now show that the above example is asymptotically optimal in the class of all graphs of constant diameter, thanks to the following:

Corollary 3. *Let $G = (V, E)$ be a graph with n vertices of constant diameter, and let Q be a query set that verifies G w.r.t. the routing-table query model. Then $|Q| = \Omega(\log n)$.*

Proof. Let Δ be the maximum degree of G and let D be the diameter of G. By the well-known *Moore bound* [15], we have that $n \leq 1 + \Delta \cdot \sum_{i=0}^{D-1}(\Delta - 1)^i \leq \Delta^{D+1}$. Therefore, $\Delta \geq n^{\frac{1}{D+1}}$. From Corollary 1, we have that $|Q| = \Omega(\log \Delta)$. As a consequence, $|Q| = \Omega(\log n^{\frac{1}{D+1}})$, i.e., $|Q| = \Omega(\log n)$ for $D = O(1)$. □

5 Verifying Graphs of Diameter 2

Even though the problem of determining whether the Network Verification Problem w.r.t. the routing-table query model is in NPO is open, in this section we show the existence of a polynomial-time approximation algorithm for graphs of diameter equal to 2. More precisely, we first show that the size of the query set returned by this algorithm is within an $O(\log n)$ (multiplicative) factor from the size of any optimal solution. Furthermore, we also show that this result is asymptotically best possible. Indeed, for graphs of diameter equal to 2, we prove that unless P = NP, no polynomial time algorithm can compute a set of queries that verifies the graph w.r.t. the routing-table query model whose size is within an $o(\log n)$ (multiplicative) factor from the size of any optimal solution.

To describe our algorithm, we need to introduce some definitions. Let $G = (V, E)$ be a graph. A set $U \subseteq V$ is a *locating-dominating code* of G iff (i) $N_G[v] \cap U \neq \emptyset$ for every $v \in V$ and (ii) $N_G(v) \cap U \neq N_G(u) \cap U$ for every $u, v \in V \setminus U, u \neq v$. The optimization problem of computing a minimum cardinality locating-dominating code of a graph G of n vertices can be approximated within a factor of $O(\log n)$ and this ratio is asymptotically tight [10, 21]. We start by proving the following:

Lemma 3. *Let $G = (V, E)$ be a graph of diameter equal to 2, let U^* be a minimum cardinality locating-dominating code of G, and let $Q \subseteq V$ be a set of vertices that verifies G w.r.t. the routing-table query model. Then $|Q| \geq |U^*| - 1$.*

Proof. Let $u, v \in V \setminus Q$, with $u \neq v$. By Proposition 6 we have that if Q verifies G, then $|d_G(q, u) - d_G(q, v)| \geq 1$, for some $q \in Q$. As G has diameter equal to 2, we either have that $d_G(q, u) = 1$ and $d_G(q, v) = 2$, or $d_G(q, u) = 2$ and $d_G(q, v) = 1$. As this has to be true for every two distinct vertices $u, v \in V \setminus Q$, it follows that there exists at most one vertex, say $\bar{v}$, such that $d_G(q, \bar{v}) = 2$

for every $q \in Q$. As a consequence, $Q \cup \{\bar{v}\}$ is a locating-dominating code of G. Thus, $|Q| + 1 \geq |U^*|$. □

A locating-dominating code U of G is said to be *connected* iff $G[U]$ is a connected graph. We now prove the following:

Lemma 4. *Any connected locating-dominating code (CLDC) of a graph $G = (V, E)$ verifies G w.r.t. the routing-table query model.*

Proof. Let Q be a CLDC of G. We will show how to use the information returned by querying Q to infer the existence/non-existence of (u, v) in G.

Let us fix any two distinct vertices u and v of G such that $u, v \notin Q$ (otherwise edge/non-edge (u, v) is trivially discovered). As Q is a CDLC of G, there exist (i) a vertex $q \in Q$ such that, w.l.o.g., $q \in N_G(u)$ and $q \notin N_G(v)$, (ii) a vertex $q' \in Q$ such that $q' \in N_G(v)$, and (iii) a path P between q and q' in $G[Q]$.

Let then $q_1, \ldots, q_\ell$ be the vertices of P in the order from q' to q. Thus, $q_1 = q'$ and $q_\ell = q$. We show that $\{q_1, \ldots, q_\ell\}$ discovers the edge/non-edge (u, v) of G by first showing that $\{q_1, \ldots, q_\ell\}$ *discovers* $d_G(q, v)$ (i.e., we are able to certify the distance in G between q and v by using the information returned by the queries at $\{q_1, \ldots, q_\ell\}$). We use induction to this aim, by showing that $\{q_1, \ldots, q_i\}$ discovers $d_G(q_i, v)$, for $i = 1, \ldots, \ell$. The base case $i = 1$ is clearly true as $\langle v, v\rangle \in \texttt{table}_G(q_1)$ means to discover that the distance in G between q_1 and v is equal to 1. Now, assume that $\{q_1, \ldots, q_i\}$ discovers $d_G(q_{i-1}, v)$. Observe that $|d_G(q_{i-1}, v) - d_G(q_i, v)| \leq 1$, since (q_i, q_{i+1}) is an edge of G. Therefore, using induction, we have that $\{q_1, \ldots, q_i\}$ discovers $d_G(q_i, v)$ by observing that:

– $d_G(q_i, v) = d_G(q_{i-1}, v) + 1$ iff $\langle v, q_{i-1}\rangle \in \texttt{table}_G(q_i)$;

– $d_G(q_i, v) = d_G(q_{i-1}, v) - 1$ iff $\langle v, q_i\rangle \in \texttt{table}_G(q_{i-1})$;

– $d_G(q_i, v) = d_G(q_{i-1}, v)$ iff $\langle v, q_{i-1}\rangle \notin \texttt{table}_G(q_i)$ and $\langle v, q_i\rangle \notin \texttt{table}_G(q_{i-1})$.

Then, once we have discovered $d_G(q, v)$, we can deduce that (u, v) is either an edge or not of G as follows (observe we discovered that $d_G(q, v) > 1$ since after querying q we discovered the non-edge (q, v)):

– if $d_G(q, v) = 2$, then (u, v) is either an edge or not of G depending on whether $\langle v, u\rangle \in \texttt{table}_G(q)$ or not;
– if $d_G(q, v) > 2$, then (u, v) is a non-edge of G, since we discovered that $d_G(q, u) = 1$ (i.e., we discovered the edge (q, u)).

□

We are now ready to prove the following:

Theorem 2. *Let $G = (V, E)$ be a graph of diameter equal to 2 and let Q^* be a minimum cardinality set of queries that verifies G w.r.t. the routing-table query model. There exists a polynomial-time algorithm that computes a set Q that verifies G w.r.t. the routing-table query model such that $\frac{|Q|}{|Q^*|} = O(\log n)$.*

Proof. Let U^* be a minimum cardinality locating-dominating code of G. As U^* is also a dominating set of G, it is easy to construct a CLDC U of G such that $U^* \subseteq U$ and $|U| = O(|U^*|)$ (see also [11]). Therefore, thanks to the $O(\log n)$-approximation algorithm for computing a locating-dominating code of G (see [10]), we can also compute a CLDC Q of G such that $\frac{|Q|}{|U^*|} = O(\log n)$. Thus, from Lemma 4 we know that Q verifies G, and from Lemma 3 we have that if G has diameter 2, then $\frac{|Q|}{|Q^*|} \leq \frac{|Q|}{|U^*|-1} = O(\log n)$. □

We observe that the result of Theorem 2 is asymptotically tight due to the following:

Theorem 3. *The Network Verification Problem w.r.t. the routing-table query model cannot be approximated within a ratio of* $o(\log n)$ *in the class of all graphs of diameter 2, unless* $\mathsf{P} = \mathsf{NP}$.

Proof. In [1], the authors proved that the Network Verification Problem w.r.t. the all-shortest-paths query model has a lower bound of $\Omega(\log n)$ on its approximability ratio, unless $\mathsf{P} = \mathsf{NP}$. Their reduction consists of a graph G having a vertex which is adjacent to all other vertices of G. The claim now follows as a consequence of Proposition 6 and Proposition 7. □

6 Optimal Algorithms for Classical Network Topologies

In what follows, we prove that the Network Verification Problem w.r.t. the routing-table query model can be solved in linear time for paths, trees, and cycles of even length. We first provide a general algorithm for trees, which clearly applies to paths as well. Then, we try to characterize the size of an optimal query set for them, as a function of the corresponding number of nodes. While this can be explicitly done for paths, we will see that for a tree such a characterization is depending on its structure. Finally, we turn our attention to cycles, and for those of even length, we compute an optimal query set of size $2\lfloor \frac{n}{6} \rfloor + \frac{n}{2} \bmod 3$. Our approach heavily relies on the existence of antipodal nodes in the cycle, and so it is not easily extendible to cycles of odd length (for which however we are able to provide a lower bound of $\lfloor \frac{n}{8} \rfloor$).

6.1 Paths and trees

In this section, we first provide the linear time algorithm for trees (and paths), and then we try to characterize the size of an optimal query set, as a function of the corresponding number of nodes. While this can be explicitly done for paths, for trees we will see that such a characterization is depending on the tree structure.

We start by observing that a tree of 2 vertices can be easily verified by querying any of the two vertices. For trees of larger order, we need to provide a sufficient condition for verifying edges first.

Proposition 9. *Let (u,v) be an edge of a tree T and let $Q = \big(N_T(u) \setminus \{v\}\big) \cup \big(N_T(v) \setminus \{u\}\big)$. Then Q verifies the edge (u,v) of T.*

Proof. Let G be any graph such that $\texttt{table}_T(q) = \texttt{table}_G(q)$ for every $q \in Q$. Let P be any shortest path between u and v in G. We show that $V(P) = \{u,v\}$, thus proving that (u,v) is an edge of G.

First of all, observe that for every $q \in N_T(u) \setminus \{v\}$, $\langle u,u\rangle, \langle v,u\rangle \in \texttt{table}_G(q)$. Similarly, for every $q \in N_T(v) \setminus \{u\}$, $\langle v,v\rangle, \langle u,v\rangle \in \texttt{table}_G(q)$. Thus, for every $q \in N_T(u) \setminus \{v\}$, $V(P) \subseteq \texttt{T}^q_G(u) = \texttt{T}^q_T(u)$. Similarly, for every $q \in N_T(v) \setminus \{u\}$, $V(P) \subseteq \texttt{T}^q_G(v) = \texttt{T}^q_T(v)$. Therefore, if T_u and T_v denote the two trees of $T-(u,v)$ rooted at u and v, respectively, then

$$V(P) \subseteq \bigcap_{q \in N_T(u)\setminus\{v\}} \texttt{T}^q_T(u) = V(T_v) \cup \{u\}$$

and

$$V(P) \subseteq \bigcap_{q \in N_T(v)\setminus\{u\}} \texttt{T}^q_T(v) = V(T_u) \cup \{v\}$$

imply

$$V(P) \subseteq \big(V(T_v) \cup \{u\}\big) \cap \big(V(T_u) \cup \{v\}\big) = \{u,v\}.$$

□

Now, we provide a necessary and sufficient condition a set of queries must satisfy to verify a tree of order greater than or equal to 3.

Lemma 5. *Let T be a tree of order $n \geq 3$ and let $Q \subseteq V(T)$. Q verifies T iff for every two vertices $u,v \in V(T)$ with $d_T(u,v) = 2$, $Q \cap \{u,v\} \neq \emptyset$.*

Proof. Let $u,v \in V(T)$ be two vertices with $d_T(u,v) = 2$. Observe that (u,v) is a non-edge. First we show that if Q verifies the non-edge (u,v), then $Q \cap \{u,v\} \neq \emptyset$. To this end, it is enough to show that for every vertex $u' \in V(T) \setminus \{u,v\}$ $\texttt{table}_T(u') = \texttt{table}_{T+(u,v)}(u')$, i.e., the routing tables of vertices other than u and v would not change if we added edge (u,v) to T, and so we need to query at least one of u,v in order to distinguish between the existence or not of edge (u,v).

Let x be the (unique) vertex in the (unique) shortest path between u and v in T. Root T at x and let T_u, T_v be the subtrees of T rooted at u and v, respectively. Let T_x be the subtree induced by $V(T) \setminus (V(T_u) \cup V(T_v))$. Fix a vertex $u' \in V(T_x)$ and observe that $d_T(u',u) = d_T(u',v)$. As a consequence, for every $v' \in V(T)$, P is a shortest path from u' to v' in T iff P is a shortest path from u' to v' in $T+(u,v)$. Therefore, $\texttt{table}_T(u') = \texttt{table}_{T+(u,v)}(u')$, for every $u' \in V(T_x)$.

Now, fix a vertex $u' \in V(T_u) \setminus \{u\}$ (the proof for the case $u' \in V(T_v) \setminus \{v\}$ is analogous). For any vertex $v' \in V(T)$, let P and P' be the (unique) shortest path from u' to v' in T and $T+(u,v)$, respectively. First of all, observe that if $v' \in V(T_u)$, then $P = P'$ and therefore, $\langle v', v''\rangle \in \texttt{table}_T(u')$ iff $\langle v', v''\rangle \in$

$\texttt{table}_{T+(u,v)}(u')$. Next, if $v' \in V(T_x) \cup V(T_v)$, then both P and P' contain vertex u. Moreover, the subpath of P from u' to u coincides with the subpath of P' from u' to u and has length greater than or equal to 1. Therefore $\langle v', v'' \rangle \in \texttt{table}_T(u')$ iff $\langle v', v'' \rangle \in \texttt{table}_{T+(u,v)}(u')$. Hence, $\texttt{table}_T(u') = \texttt{table}_{T+(u,v)}(u')$ for every $u' \in V(T_u) \setminus \{u\}$.

To complete the proof, we have to show that if $Q \cap \{u, v\} \neq \emptyset$ for every $u, v \in V(T)$ with $d_T(u, v) = 2$, then Q verifies T. Let u and v be two distinct vertices of T. We show that Q verifies whether (u, v) is an edge or a non-edge of T. Clearly, if $Q \cap \{u, v\} \neq \emptyset$, then Q verifies whether (u, v) is an edge or a non-edge of T. Therefore, assume that $u, v \notin Q$. As a consequence, either $d_T(u, v) = 1$ (in this case (u, v) is an edge of T) or $d_T(u, v) \geq 3$ (in this case (u, v) is a non-edge of T).

In the latter case, let P be the shortest path in T between u and v. It is not hard to see that either Q contains a vertex $q \in V(P)$ which is also a neighbor of either u or v, or Q contains two vertices $q, q' \in V(P)$ and $(q, q') \in E(T)$. In the first case, from Proposition 4 we have that Q verifies the non-edge (u, v) while in the second case, Proposition 5 implies that Q verifies the non-edge (u, v).

In the first case, i.e., $d_T(u, v) = 1$ (or equivalently, (u, v) is an edge of T), we have that $N_T(u) \setminus \{v\} \subseteq Q$ and $N_T(v) \setminus \{u\} \subseteq Q$ as $d_T(u', v) = 2$ for every $u' \in N_T(u) \setminus \{v\}$ and $d_T(u, v') = 2$ for every $u' \in N_T(v) \setminus \{u\}$. Therefore, from Proposition 9 we have that Q verifies the edge (u, v) of T. □

Thanks to Lemma 5, we can prove the following:

Theorem 4. *The Network Verification Problem w.r.t. the routing-table query model on trees can be solved in linear time.*

Proof. Let G' be a graph with $V(G') = V(T)$ and $E(G') = \{(u, v) \mid u, v \in V(T), d_T(u, v) = 2\}$, and let $Q \subseteq V(T)$. Observe that from Lemma 5, Q verifies T iff Q is a *vertex cover* of G', i.e., for every edge $(u, v) \in E(G')$, $Q \cap \{u, v\} \neq \emptyset$. Let r be any vertex of T and root T at r. Let $X = \{v \mid v \in V(T), d_T(r, v) = 2k-1, k \in \mathbb{N}\}$, and let $X' = V(T) \setminus X$. Observe that G' has exactly two connected components: one containing all the vertices in X, and the other one containing all the vertices in X'. To understand the structure of $G'[X]$ and $G'[X']$ we need to introduce the class of *block graphs* (a.k.a. *clique trees*) which was defined in [12], as well as the notion of *block cut vertex tree* [13].

Given a graph G, let V_c be the set of cut vertices of G, and let $\mathcal{B} = \{B_1, \ldots, B_h\}$ be the set of biconnected components of G.[2] G is said to be a *block graph* iff every biconnected component of G is a clique. The *block cut vertex tree* $\mathcal{T}_G$ of G is the tree with vertex set $V(\mathcal{T}_G) = V_c \cup V_b := \{v_i \mid B_i \in \mathcal{B}\}$, and edge set $E(\mathcal{T}_G) = \{(u, v_i) \mid u \in V_c, u \in V(B_i)\}$. Notice that $\mathcal{T}_G$ can be computed in linear time [16]. Notice also that $G'[X]$ and $G'[X']$ are both block graphs (see also Figure 1).

[2] A *cut vertex* of a graph is a vertex v of the graph whose removal from G results in a graph which is not connected. A graph G is said to be *biconnected* iff it has no cut vertex. The *biconnected components* of a graph G are the maximal -w.r.t. vertex addition- biconnected induced subgraphs of G.

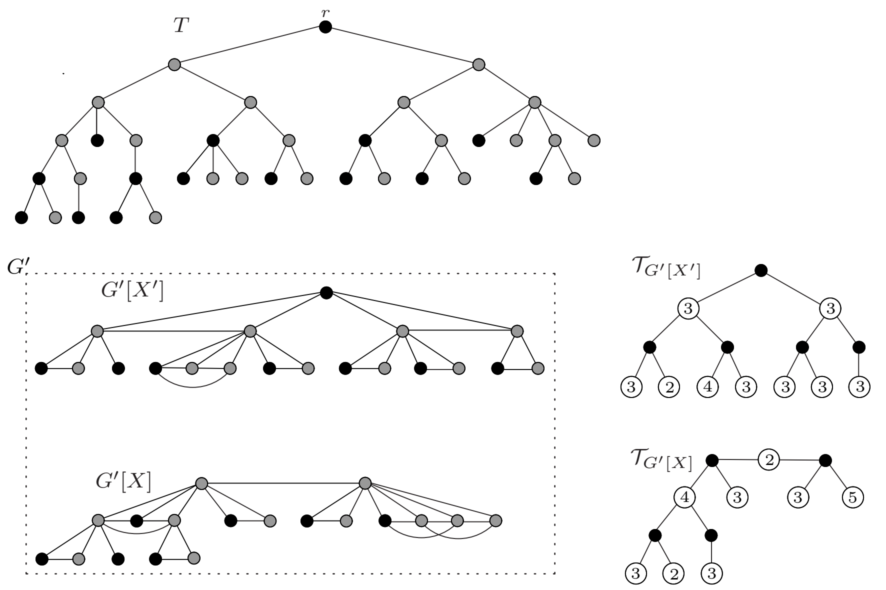


Fig. 1. An example of an optimal query set for a tree T depicted on top. On the bottom, the graph G' on $V(T)$ containing an edge between two vertices iff their distance in T is equal to 2. X is the set of vertices of T whose distance from r is an odd number while $X' = V(T) \setminus X$. Observe that both connected components of G' are block graphs. On the right side of $G'[X]$ and $G'[X']$ the corresponding block cut vertex trees $\mathcal{T}_{G'[X]}$ and $\mathcal{T}_{G'[X']}$ are depicted. Black vertices of $\mathcal{T}_{G'[X]}$ (resp., $\mathcal{T}_{G'[X']}$) are the cut vertices of $G'[X]$ (resp., $G'[X']$). White vertices of $\mathcal{T}_{G'[X]}$ (resp., $\mathcal{T}_{G'[X']}$) are the biconnected components (i.e., maximal cliques) of $G'[X]$ (resp., $G'[X']$). The number appearing inside a white vertex denotes the size of the clique corresponding to that vertex. The set of gray vertices is a minimum cardinality vertex cover of G' as well as a minimum cardinality set of queries that verifies T.

We now show that the minimum cardinality vertex cover problem on block graphs can be solved in linear time by using the following greedy algorithm. Let then G be a block graph, and let $\mathcal{T}_G = V_c \cup V_b$ be the corresponding block cut vertex tree. Let s be a vertex of V_c, and root $\mathcal{T}_G$ at s. Let ϕ be a function assigning the number $|V(B_i)|$ to the vertex $v_i \in V_b$ representing (clique) B_i. We compute the size t of a minimum cardinality vertex cover of G by using function ϕ in the following way. At the beginning, $t = 0$. As long as $\mathcal{T}_G$ is not empty, let $u \in V_c$ be a vertex that is farthest from s in $\mathcal{T}_G$ (ties are broken arbitrarily), and, w.l.o.g., let $v_{i_1}, \ldots, v_{i_k} \in V_b$ be the k children of u (notice they are leaves of $\mathcal{T}_G$). First, add $\sum_{j=1}^{k} \phi(v_{i_j}) - 2k + 1$ to t. Next, remove from $\mathcal{T}_G$ vertices $u, v_{i_1}, \ldots, v_{i_k}$, and decrease $\phi(v)$ by 1, where $v \in V_b$ is the parent of u in $\mathcal{T}_G$. Finally, if $\phi(v) = 1$, then remove v from $\mathcal{T}_G$, and if v has no siblings in $\mathcal{T}_G$, then remove also its parent.

To see why this algorithm is correct, simply observe that every vertex cover of G has to contain at least $\phi(v_i)-1$ vertices for every clique B_i of G. Furthermore, it is easy to see that any vertex cover U of G can be transformed into a vertex cover U' of G such that $|U'| \leq |U|$, $u \in U'$, and $|U' \cap V(B_{i_j})| = \phi(v_{i_j}) - 1$ for every $j = 1, \ldots, k$, thus including exactly $\sum_{j=1}^{k} \phi(v_{i_j}) - 2k + 1$ vertices of $\bigcup_{j=1}^{k} V(B_{i_j})$ in a minimum cardinality vertex cover of G. From this, the claim follows. □

6.2 Size of the optimal query set for paths and trees

We now turn our attention to the problem of providing a measure on the size of the optimal query set for paths and trees. By a simple application of Lemma 5, we can first prove the following exact bound on the minimum number of queries needed to verify a path.

Theorem 5. *A minimum cardinality set of queries that verifies a path P_n of order $n \geq 3$ has size equal to $2\lfloor \frac{n}{4} \rfloor + \lfloor \frac{n \bmod 4}{3} \rfloor$.*

Proof. Number the vertices of P_n from 1 to n by traversing the path from one endvertex to the other one. First of all, observe that the graph G' as defined in the proof of Theorem 4 is a forest of two paths, one containing all the $\lceil \frac{n}{2} \rceil$ odd vertices, and the other one containing all the $\lfloor \frac{n}{2} \rfloor$ even vertices (see also Figure 2). As the minimum cardinality vertex cover of a path of k vertices is $\lfloor \frac{k}{2} \rfloor$,[3] we have that a minimum cardinality set of queries that verifies P_n has size equal to $\lfloor \frac{n}{4} \rfloor + \lfloor \frac{\lceil \frac{n}{2} \rceil}{2} \rfloor = 2\lfloor \frac{n}{4} \rfloor + \lfloor \frac{n \bmod 4}{3} \rfloor$ (see also Figure 2). □

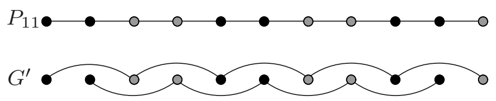


Fig. 2. An example of an optimal query set for P_{11}. Graph G' on $V(P_{11})$ contains an edge between two vertices iff their distance in P_{11} is 2. The set of gray vertices is a minimum vertex cover of G' as well as a minimum-size set of queries that verifies P_{11}.

Concerning trees, we start by proving the following two lower bounds on the minimum number of queries needed to verify trees of small constant radius.

[3] Let P_k be a path of order k. Number the vertices of P_k from 1 to k by traversing the path from one endvertex to the other one. The set $X = \{i \mid 1 \leq i \leq k, i \text{ is even}\}$ of $\lfloor k/2 \rfloor$ vertices is a vertex cover of the path. To see that it is minimum, first of all observe that $|X|$ is equal to the size of $\{(i-1, i) \mid 1 \leq i \leq k, i \text{ is even}\}$, a maximum matching of the path. Next, use the well-known König-Egerváry theorem stating that the size of a minimum cardinality vertex cover of a bipartite graph G is equal to the size of a maximum matching of G.

Proposition 10. *Let T be a rooted tree of order n and height 1. Then, any set of queries that verifies T has size greater than or equal to $n-2$.*

Proof. From Lemma 5, a set of queries that verifies T must contain at least all the leaves of T but one. □

Proposition 11. *Let T be a rooted tree of order n and height 2 and let ℓ be the number of non-leaf vertices of T. Then any set of queries that verifies T has size greater than or equal to $n-\ell$.*

Proof. Let ℓ_1 be the number of non-leaf vertices of T at distance 1 from the root. Clearly, $\ell = 1+\ell_1$. Let X_i be the set of vertices of T at distance i from the root. Clearly $n = |X_0| + |X_1| + |X_2|$. Let Q be a set of queries that verifies T. From Lemma 5, $|Q \cap X_1| \geq |X_1| - 1$ while $|Q \cap (X_0 \cup X_2)| \geq |X_0| + |X_2| - \ell_1$. As all the sets X_0, X_1, X_2 are pairwise disjoint, it follows that

$$|Q| \geq |Q \cap X_1| + |Q \cap (X_0 \cup X_2)| \geq |X_1| - 1 + |X_0| + |X_2| - \ell_1 = n - \ell.$$

□

Proposition 10 and Proposition 11 are useful to prove a lower bound on the minimum number of queries needed to verify any tree. Let T be a rooted tree of order n and let L be the set of leaves of T. For every $v \in V(T) \setminus L$, let C_v be the set of children of v in T. We define by $\widetilde{\delta}(T)$ the average number of children of the non-leaf vertices of T, i.e., $\widetilde{\delta}(T) = \frac{\sum_{v \in V(T)\setminus L} |C_v|}{n-|L|}$. Observe that $\widetilde{\delta}(T) = \frac{n-1}{n-|L|}$.

Theorem 6. *Let T be a rooted tree of order $n \geq 3$. Then, any set of queries that verifies T w.r.t. the routing-table query model has size greater than or equal to $\frac{\widetilde{\delta}(T)-2}{\widetilde{\delta}(T)}(n-1)$.*

Proof. Root T at any vertex r of degree greater than or equal to 2 and let L be the set of leaves of T. We decompose T into a collection $\mathcal{F} = \{T_1, \ldots, T_{n-|L|}\}$ of edge-disjoint subtrees as follows. Let T' be a copy of the tree. Let $i = 1$ and let $\mathcal{F} = \emptyset$. Repeat the following procedure. While the height of T' is strictly greater than 1, let v be a leaf vertex that has maximum distance from r. Let v_i be equal to the parent of v. Let T_i be the subtree of T' rooted at v_i. Update $T' = T' - C_{v_i}$, $\mathcal{F} = \mathcal{F} \cup \{T_i\}$, and increase i by 1. At the end of the while loop, let $T_i = T'$, add T_i to $\mathcal{F}$, and set $v_i = r$. Let Q be a set of queries that verifies T and observe that for every $i \leq n - |L|$, T_i is a rooted tree of height 1 and order $1 + |C_{v_i}|$. As a consequence, from Proposition 10, $|Q \cap V(T_i)| \geq |C_{v_i}| - 1$. Therefore, as each vertex in $V(T) \setminus L$ appears in at most two distinct trees of $\mathcal{F}$ while each vertex in L appears in exactly one tree of $\mathcal{F}$, we have that

$$\begin{aligned}
|Q| &\geq \sum_{i=1}^{n-|L|} |Q \cap V(T_i)| - (n-|L|) \geq \sum_{i=1}^{n-|L|} (|C_{v_i}| - 1) - (n-|L|) \\
&= \sum_{v \in V(T)\setminus L} |C_v| - (n-|L|) - (n-|L|) \geq (n-1) - (n-|L|) - (n-|L|) \\
&= \frac{\widetilde{\delta}(T) - 2}{\widetilde{\delta}(T)}(n-1),
\end{aligned}$$

where the last inequality follows because $\sum_{v\in V(T)\setminus L}|C_v| = n-1$, while the last equality follows by substituting $(n-|L|) = \frac{n-1}{\delta(T)}$. □

Using a similar technique, we can prove a linear lower bound for any tree of order $n \geq 3$.

Theorem 7. *Any query set that verifies a tree T of order $n \geq 3$ w.r.t. the routing-table query model has size greater than or equal to $\lceil \frac{n}{3} \rceil$.*

Proof. We partition the vertices of T into a collection $\mathcal{F} = \{T_1, \ldots, T_k\}$ of vertex-disjoint subtrees as follows. Let T' be a copy of the tree. Let $i = 1$ and let $\mathcal{F} = \emptyset$. Repeat the following procedure. While the radius of T' is strictly greater than 2, root T' at a center r of T'. Let v be a leaf vertex that has maximum distance from r. Let v_i be equal to the grandparent of v. Let T_i be the subtree of T' rooted at v_i. Update $T' = T' - V(T_i)$, $\mathcal{F} = \mathcal{F} \cup \{T_i\}$, and increase i by 1. At the end of the while loop, let $T_i = T'$, add T_i to $\mathcal{F}$, and set $v_i = r$.

First of all, observe that every T_i is a tree of radius equal to 1 or 2 such that $|V(T_i)| \geq 3$. Let Q be a set of queries that verifies T. For each $i = 1, \ldots, k$, let n_i be the order of T_i and let ℓ_i be the number of non-leaf vertices of T_i. If T_i is of radius 1, from Proposition 10, $|Q \cap V(T_i)| \geq n_i - 2 \geq \frac{n_i}{3}$ as $n_i \geq 3$. If T_i is of radius 2, $n_i - \ell_i \geq \frac{n_i}{3}$. Hence, from Proposition 11, $|Q \cap V(T_i)| \geq n_i - \ell_i \geq \frac{n_i}{3}$. Therefore, as all the trees $T_1, T_2, \ldots, T_k$ are vertex-disjoint,

$$|Q| \geq \sum_{i=1}^{k} |Q \cap V(T_i)| \geq \sum_{i=1}^{k} \frac{n_i}{3} = \frac{n}{3}\,.$$

□

The lower bound of Theorem 7 is tight for every $n \geq 3$. Indeed, for $n = 3k-i$, $i \in \{0,1,2\}$, consider the tree T of order n obtained by connecting the endvertices $v_1, v_2, \ldots, v_k$ of k vertex-disjoint paths, out of which $k-1$ paths have length 2 and 1 path has length $2-i$, as a path of k nodes. Observe that from Lemma 5, $\{v_1, v_2, \ldots, v_k\}$ is a query set that verifies T w.r.t. the routing-table query model. Furthermore, $|\{v_1, v_2, \ldots, v_k\}| = k = \lceil \frac{n}{3} \rceil$.

6.3 Cycles of Even Length

In this section we prove that the Network Verification Problem w.r.t. the routing-table query model on cycles of even length can be solved in linear time. Let C_n be a cycle of even length, i.e., $n = 2k$ for some integer $k \geq 2$. We will show that $2\lfloor \frac{n}{6} \rfloor + \frac{n}{2} \bmod 3$ queries are necessary and sufficient to verify C_n. We number the vertices of C_n from 0 to $n-1$ clockwise. Thus, the two neighbours of vertex i are $(i+1) \bmod n$ and $(i-1) \bmod n$. In what follows, we will assume that all indices are modulo n. We start by proving the following necessary conditions.

Proposition 12. *Let $n = 2k$ for some integer $k \geq 2$. Let $Q \subseteq V(C_n)$. If Q verifies C_n, then for every $i \in \{0, \ldots, n-1\}$, one among $i-1, i+1, i+k-1, i+k+1$ is contained in Q.*

Proof. For the sake of contradiction, assume that Q verifies C_n but $i-1, i+1, i+k-1, i+k+1 \notin Q$. Consider the graph $G' = C_n + \{(i-1, i+1), (i+k-1, i+k+1)\}$ and observe that $\texttt{table}_{C_n}(q) = \texttt{table}_{G'}(q)$ for every $q \in Q$ (see also Figure 3 (a)). □

Proposition 13. *Let $n = 2k$ for some integer $k \geq 2$. Let $Q \subseteq V(C_n)$. If Q verifies C_n, then for every $i \in \{0, \dots, n-1\}$, one among $i, i+1, i+k-1, i+k, i+k+1, i+k+2$ is contained in Q.*

Proof. The proof immediately follows from Proposition 8. As an alternative proof, consider the graph $G' = C_n - \{(i+k-1, i+k), (i+k+1, i+k+2)\} + \{(i+k-1, i+k+1), (i+k, i+k+2)\}$ and assume, for the sake of contradiction, that Q does not contain any of the vertices in the statement. Observe that $\texttt{table}_{C_n}(q) = \texttt{table}_{G'}(q)$ for every $q \in Q$ (see also Figure 3 (b)). □

Proposition 14. *Let $n = 2k$ for some integer $k \geq 2$. Let $Q \subseteq V(C_n)$. If Q verifies C_n, then for every $i \in \{0, \dots, n-1\}$, one among $i-1, i, i+1, i+k, i+k+1, i+k+2$ is contained in Q.*

Proof. Consider the graph $G' = C_n + \{(i-1, i+1), (i+k, i+k+2)\}$ and assume, for the sake of contradiction, that Q does not contain any of the vertices in the statement. Observe that $\texttt{table}_{C_n}(q) = \texttt{table}_{G'}(q)$ for every $q \in Q$ (see also Figure 3 (c)). □

Thanks to the above propositions, we can prove the following lower bound on the number of queries needed to verify C_n.

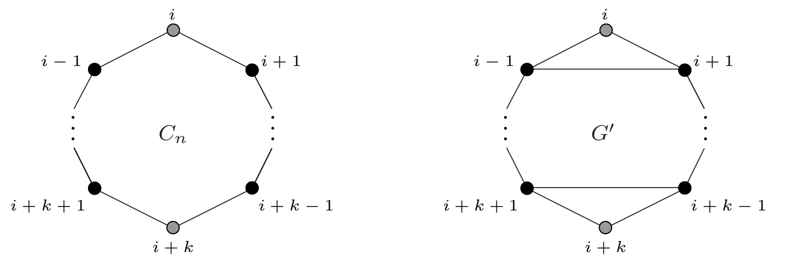
Lemma 6. *Let $n = 2k$ for some integer $k \geq 2$. Let $Q \subseteq V(C_n)$. If Q verifies C_n, then $|Q| \geq 2\lfloor \frac{n}{6} \rfloor + \frac{n}{2} \bmod 3$.*

Proof. Let $Q \subseteq V(C_n)$ be a set of queries that verifies C_n. For every $i \in \{0, \dots, 2k-1\}$, let $\texttt{counter}(i) = |Q \cap \{i, i+k\}|$. Clearly, $|Q| = \sum_{i=0}^{k-1} \texttt{counter}(i)$. As Q verifies C_n, then it also satisfies Propositions 12, 13, and 14. The following holds:

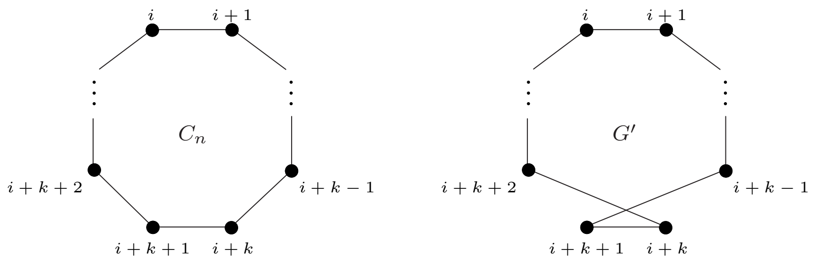
Claim 1. *For every i such that $\texttt{counter}(i) = \texttt{counter}(i+1) = 0$ we have that $\texttt{counter}(i-1) = 2$ or $\texttt{counter}(i+2) = 2$.*

Proof. By assumption, we have that $i, i+1, i+k, i+k+1 \notin Q$. Moreover, by Proposition 12, we have that $\texttt{counter}(i-1), \texttt{counter}(i+2) \geq 1$. To show that $\texttt{counter}(i-1) \geq 2$ or $\texttt{counter}(i+2) \geq 2$, assume for the sake of contradiction that $\texttt{counter}(i-1) = \texttt{counter}(i+2) = 1$. Then, by Proposition 13, we have that $i+k-1 \in Q$ or $i+k+2 \in Q$. W.l.o.g., assume that $i+k+2 \in Q$. As $\texttt{counter}(i+2) = 1$, we have that $i+2 \notin Q$. Then, $i, i+1, i+2, i+k, i+k+1 \notin Q$, and then from Proposition 14, we have that $i+k-1 \in Q$. As $\texttt{counter}(i-1) = 1$, this implies $i-1 \notin Q$. Therefore, $i+k, i+k+1, i-1, i, i+1, i+2 \notin Q$ thus violating the necessary condition of Proposition 13. □

(a)



(b)



(c)

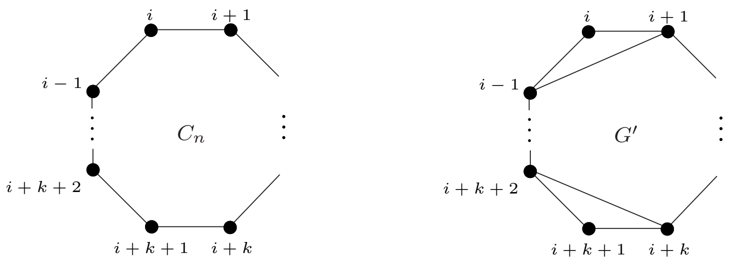


Fig. 3. Necessary conditions for the query set to verify a cycle C_n of even length. In each row of the picture, at least one black vertex of C_n on the left side must be part of the query set in order to distinguish between C_n and the graph G' on the right side.

Let $X = \{i \mid 0 \leq i \leq k-1, \texttt{counter}(i) = 0, \texttt{counter}(i-1), \texttt{counter}(i+1) \geq 1\}$. Let $Y = \{(i, i+1) \mid 0 \leq i \leq k-1, \texttt{counter}(i), \texttt{counter}(i+1) = 0\}$. Finally, let ℓ be the number of vertices i, with $0 \leq i \leq k-1$ such that $\texttt{counter}(i), \texttt{counter}(i-1), \texttt{counter}(i+1) \geq 1$. Since Proposition 12 holds, we have that $\texttt{counter}(i) + \texttt{counter}(i+2) \geq 1$, for every $i \in \{0, 1, \ldots, k-1\}$. As a consequence, for every $i \in X$, we have that $\texttt{counter}(i-2), \texttt{counter}(i-1), \texttt{counter}(i+1), \texttt{counter}(i+2) \geq 1$. Similarly, for every $(i, i+1) \in Y$, we have that $\texttt{counter}(i-2), \texttt{counter}(i-1), \texttt{counter}(i+2), \texttt{counter}(i+3) \geq 1$. This implies that $k = 3|X| + 4|Y| + \ell$ and $|X| + |Y| \leq \lfloor k/3 \rfloor$. Finally, note that we cannot have three consecutive nodes with $\texttt{counter}(i-1) = \texttt{counter}(i) = \texttt{counter}(i+1) = 0$, as this would also imply $\texttt{counter}(i+k-1) = \texttt{counter}(i+k) = \texttt{counter}(i+k+1) = 0$, which would contradict Proposition 12. Therefore, by the above claim, we have that $|Q| \geq 2|X| + 3|Y| + \ell$. We conclude that

$$|Q| \geq 2|X| + 3|Y| + \ell \geq k - |X| - |Y| \geq 2\left\lfloor \frac{k}{3} \right\rfloor + k \bmod 3 = 2\left\lfloor \frac{n}{6} \right\rfloor + \frac{n}{2} \bmod 3.$$

□

We now show that the above bound is tight:

Lemma 7. *The set* $Q = \{i \mid 0 \leq i \leq k-1, i \bmod 3 \in \{0, 1\}\}$ *of size* $2\left\lfloor \frac{n}{6} \right\rfloor + \frac{n}{2} \bmod 3$ *verifies* C_n*.*

Proof. We have to show that Q verifies every edge and every non-edge of C_n. Our first goal is to establish that Q verifies every edge and non-edge of the form (i, j), where $0 \leq i \leq k-1$ and $j \neq i$.

We remark immediately that Q clearly verifies every edge and non-edge of the form (i, j), where $i \in Q$ and $j \neq i$. Now, consider any vertex $i \in \{0, \ldots, k-1\}$ with $i \notin Q$. We distinguish between the following three cases:

- If $i \leq k-3$, then, by construction, $i-2, i-1, i+1, i+2 \in Q$. By Proposition 4, $\{i+1\}$ verifies all the non-edges (i, j) with $j \in \{i+3, i+4, \ldots, i+k\}$. Similarly, $\{i-1\}$ verifies all the non-edges (i, j) with $j \in \{i+k, i+k+1, \ldots, i+2k-3 = i-3\}$. Finally, $\{i-2\}$ verifies the non-edge $(i, i-2)$, $\{i-1\}$ verifies the edge $(i, i-1)$, $\{i+1\}$ verifies the edge $(i, i+1)$, and $\{i+2\}$ verifies the non-edge $(i, i+2)$.
- If $i = k-2$, then, by construction, $k-4, k-3, k-1 \in Q$. Similarly to the previous case, by Proposition 4, $\{k-1\}$ verifies all the non-edges $(k-2, j)$ with $j \in \{k+1, k+2, \ldots, 2k-2\}$ and $\{k-3\}$ verifies all the non-edges $(k-2, j)$ with $j \in \{2k-2, 2k-1, \ldots, 2k+k-5 = k-5\}$. Moreover, $\{k-4\}$ verifies the non-edge $(k-2, k-4)$, $\{k-3\}$ verifies the edge $(k-2, k-3)$, and $\{k-1\}$ verifies the edge $(k-2, k-1)$. It remains to show that the non-edge $(k-2, k)$ is verified.
  Note that, by combining the knowledge we obtained so far about the edges and non-edges of node $k-2$ with the knowledge we obtained in the previous case about the edges and non-edges of nodes $i \leq k-3$ that do not belong

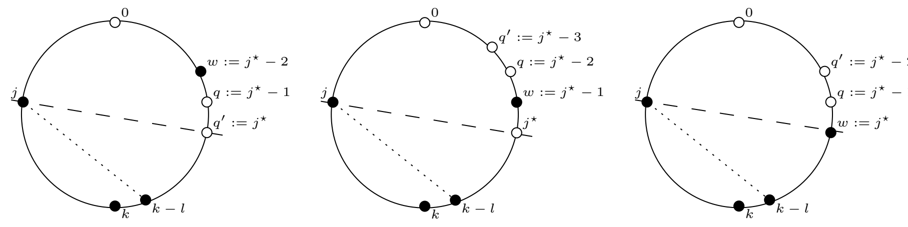


Fig. 4. The three cases in which the queries $\{q, q'\}$ verify non-edge $(k-1, j)$ in C_n by Proposition 5. Nodes depicted by hollow circles are in Q. Nodes j and $j^\star$ are antipodal in C_n. *Left:* $j^\star \bmod 3 = 1$. *Middle:* $j^\star \bmod 3 = 0$. *Right:* $j^\star \bmod 3 = 2$.

in Q and the knowledge about the edges and non-edges of nodes in Q, we know at this point that the nodes $0 \to 1 \to \cdots \to k$ form a path and there is no other edge with an endpoint in $\{1, \ldots, k-1\}$ except, possibly, $(k-2, k)$. Suppose now that $(k-2, k)$ is an edge. Since the query at $\{0\}$ returns the pair $\langle k, 1\rangle$, we know that $0 \to 1 \to \cdots \to k-2 \to k$ is a shortest path between 0 and k of length $k-1$. Moreover, the query at $\{k-1\}$ returns the pair $\langle 2k-1, k-2\rangle$, so we know that $k-1 \to k-2 \to \cdots \to 0 \to 2k-1$ is a shortest path between $k-1$ and $2k-1$ of length k. However, the query at $\{0\}$ also returns the pair $\langle k, 2k-1\rangle$, which implies that there exists a path between $2k-1$ and k of length $k-2$, since the shortest path between 0 and k is of length $k-1$. This, in turn, implies that there exists a path between $2k-1$ and $k-1$ of length $k-1$, which contradicts the fact that $k-1 \to k-2 \to \cdots \to 0 \to 2k-1$ is a shortest path between $k-1$ and $2k-1$. Therefore, $(k-2, k)$ has to be a non-edge.

– If $i = k-1$, then, by construction, $k-3, k-2 \in Q$. As before, $\{k-2\}$ verifies all the non-edges $(k-1, j)$ with $j \in \{2k-1, 0, \ldots, k-4\}$. Moreover, $\{k-3\}$ verifies the non-edge $(k-1, k-3)$ and $\{k-2\}$ verifies the edge $(k-1, k-2)$. Next, consider non-edge $(k-1, j)$, where $j \in \{k+1, k+2, \ldots, 2k-2\}$. This non-edge is verified as follows: Let $j^\star = (j+k) \bmod n$ be the antipodal node of j in C_n.
  - If $j^\star \bmod 3 = 1$ (in this case, $j^\star$ is in the range $1 \leq j^\star \leq k-2$), let $q = j^\star - 1$, $q' = j^\star$, and $w = j^\star - 2$. We have $q, q' \in Q$, $(q, q') \in E(C_n)$, $k-1 \notin \mathtt{T}^q_{C_n}(w)$, $j \in \mathtt{T}^q_{C_n}(w)$, $k-1 \in \mathtt{T}^q_{C_n}(q')$, and $j \in \mathtt{T}^{q'}_{C_n}(q)$ (see also Figure 4, left). Therefore, non-edge $(k-1, j)$ is verified by Proposition 5.
  - If $j^\star \bmod 3 \in \{0, 2\}$ (in this case, $j^\star$ is in the range $2 \leq j^\star \leq k-3$), let $q = \max\{i \mid 0 \leq i < j^\star \wedge i \in Q\}$, $q' = q-1$, and $w = q+1$. We have $q, q' \in Q$, $(q, q') \in E(C_n)$, $j \notin \mathtt{T}^q_{C_n}(w)$, $k-1 \in \mathtt{T}^q_{C_n}(w)$, $j \in \mathtt{T}^q_{C_n}(q')$, and $k-1 \in \mathtt{T}^{q'}_{C_n}(q)$ (see also Figure 4, middle and right). Therefore, non-edge $(k-1, j)$ is verified by Proposition 5.

  Finally, in order to verify edge $(k-1, k)$, note that, as in the previous case, we know that the nodes $0 \to 1 \to \cdots \to k-1$ form a path and there is no other edge with an endpoint in $\{1, \ldots, k-1\}$ except, possibly, $(k-1, k)$.

Since the query at $\{0\}$ returns $\langle k, 1\rangle$, it follows that $(k-1, k)$ must be an edge.

As a consequence of the above reasoning, we have verified so far that the nodes $0 \to 1 \to \cdots \to k$ form a path and that there do not exist any other edges with an endpoint in $\{1, \ldots, k-1\}$. Additionally, since the query at $\{0\}$ returns both $\langle k, 1\rangle$ and $\langle k, 2k-1\rangle$, we deduce that $0 \to 1 \to \cdots \to k$ is a shortest path of length k from 0 to k and, consequently, the nodes in $\{k, k+1, \ldots, 2k = 0\}$ must also form a shortest path of length k from 0 to k. Therefore, the graph must be a cycle and it only remains to verify the order in which the vertices in $\{k, k+1, \ldots, 2k = 0\}$ are traversed by the shortest path from 0 to k having $2k-1$ as first hop.

First, observe that for any $i \in Q$, there exists a unique node, $i+k$, for which the query at $\{i\}$ returns more than one pair: $\langle i+k, i-1\rangle$ and $\langle i+k, i+1\rangle$. Since we are on a cycle, this node must be antipodal to i, and therefore the distance between i and $i+k$ must be equal to k.

We will now prove that for all $x \in \{0, 1, \ldots, k-2\}$, we can verify that $0 \to 1 \to \cdots \to k \to \cdots \to k+x$ is a path. The proof proceeds by induction on x. For $x = 0$, we have already verified that $0 \to 1 \to \cdots \to k$ is a path. Now, assume that we have verified the path up to $k+x$, for some $x \in \{0, 1, \ldots, k-3\}$. In order to verify the edge $(k+x, k+x+1)$, we consider the following two cases:

- If $x+1 \in Q$, then we know that the shortest path from $x+1$ to $k+x+1$ has length k, we know that there exists a shortest path from $x+1$ to $k+x+1$ having $x+2$ as first hop, and we know that $x+1 \to x+2 \to \cdots \to k+x$ is a path of length $k-1$ on the cycle. From these, we deduce that $(k+x, k+x+1)$ must be an edge.
- If $x+1 \notin Q$, then we must have $x+2 \in Q$. We know that $x+2$ and $k+x+1$ are not antipodal and we know that the unique shortest path from $x+2$ to $k+x+1$ must go through $x+3$ and have length $k-1$ or less. However, $x+2 \to x+3 \to \cdots \to k+x$ is a path that goes through $x+3$, has length $k-2$, and does not contain $k+x+1$. Therefore, $(k+x, k+x+1)$ must be an edge, otherwise the length of the shortest path from $x+2$ to $k+x+1$ would be more than $k-1$.

Finally, the last remaining edge $(2k-2, 2k-1)$ is easily deduced since we know the graph is a cycle, $0 \to 1 \to \cdots \to 2k-2$ is a path, and we have already verified the edge $(0, 2k-1)$. □

Thus, from Lemma 6 and 7, we have obtained the following:

Theorem 8. *The Network Verification Problem w.r.t. the routing-table query model on cycles of even length can be solved in linear time.*

We conclude this section by proving a necessary condition of a query set to verify a cycle of *odd* length. Let C_n be a cycle of odd length, i.e., $n = 2k+1$, for some integer $k \geq 1$. We number the vertices of C_n from 0 to $n-1$ clockwise. Thus, the two neighbours of vertex i are $(i+1) \bmod n$ and $(i-1) \bmod n$. In what follows, we will assume that all indices are modulo n.

Proposition 15. *Let $n = 2k + 1$ for some integer $k \geq 1$. Let $Q \subseteq V(C_n)$. If Q verifies C_n, then for every $i \in \{0, \ldots, n-1\}$, one among $i, i+k-1, i+k, i+k+1, i+k+2$ is contained in Q.*

Proof. The proof immediately follows from Proposition 8. As an alternative proof, consider the graph $G' = C_n - \big\{(i+k-1, i+k), (i+k+1, i+k+2)\big\} + \big\{(i+k-1, i+k+1), (i+k, i+k+2)\big\}$ and observe that $\texttt{table}_{C_n}(q) = \texttt{table}_{G'}(q)$ for every $q \in Q$. $\square$

Observe that Proposition 15 implies a lower bound of $\left\lfloor \frac{n}{8} \right\rfloor$ on the number of queries needed to verify C_n.

7 Conclusions

In this paper, we addressed the problem of verifying a graph w.r.t. the newly defined routing-table query model. On the one hand, we showed that the problem is NP-hard to approximate within $o(\log n)$ (which is tight for graphs of diameter 2), and on the other hand that it can be solved optimally in linear time for some basic network topologies. A first open problem is therefore the development of an approximation algorithm for general graphs. Actually, Lemma 4 states that a CLDC of a graph always verifies the graph, independently of its diameter. Interestingly, this could open the way for providing an approximation algorithm for verifying any graph, if one would be able to lower bound the size of a feasible query set via the size of an optimal CLDC. Concerning the size of an optimal query set, we provided a lower bound of $\Omega(\log \log n)$ queries to verify a network of n nodes. Thus, another interesting question to address is whether this result can be improved, or alternatively to show that it is tight.

We argued that our query model is much closer to reality than the previously used all-shortest-paths and all-distances query models, as it relies on local information that can be gathered by simply exploring the routing tables of the nodes of a given network. In practice, however, routing tables could contain much more information than the one we used in defining our query model. A first natural extension is that in which the routing tables contain also the distances to the destination nodes. Apparently, this richer *routing-distances query model* should be more powerful than both the routing-table and the all-distances query model. However, a more in-depth analysis that we carried out showed that this is only partially true. Indeed, as for the all-distances query model, also for the routing-distances query model it can be shown that it is NP-hard to approximate the Network Verification Problem within $o(\log n)$. On the positive side, an $O(\log n)$-approximation algorithm for the problem can be given for arbitrary graphs (as in the all-distances query model), and this favorably compares with the routing-table query model, where we were able to exhibit such a result only for graphs of diameter 2. Thus, adding just distances is not enough to really strengthen the model. Therefore, we plan in the future to investigate further variants of the introduced model, with the goal of being as much adherent as possible to the *true* set of information that routing tables can return in the various used network communication protocols. Moreover, for the presented query model and its envisioned variants, establishing whether the network verification problem is in NPO is a challenging research question. Notice that the best currently known upper bound on the complexity of the corresponding decision problem (given a graph G and a positive integer k, can G be verified in the routing-table model with at most k queries?) is Σ_2^p, which follows easily from the definition of the problem. Finally, a major task which is completely left open is that of understanding the computational hardness of the network discovery problem in the routing-table query model.

Acknowledgements. The authors wish to thank the anonymous referees for their insightful comments, which were helpful to substantially improve the paper.

References

1. Z. Beerliova, F. Eberhard, T. Erlebach, A. Hall, M. Hoffman, M. Mihal'ák, and S. Ram, Network discovery and verification, *IEEE Journal on Selected Areas in Communications*, 24(12):2168–2181, 2006.
2. Y. Bejerano and M. Rastogi, Rubust monitoring of link delays and faults in IP networks, *IEEE/ACM Transactions on Networking*, 14(5):1092–1103, 2006.
3. D. Bilò, T. Erlebach, M. Mihal'ák, and P. Widmayer, Discovery of network properties with all-shortest-paths queries, *Theoretical Computer Science*, 411(14-15):1626–1637, 2010.
4. N.H. Bshouty and H. Mazzawi, Reconstructing weighted graphs with minimal query complexity, *Theoretical Computer Science*, 412(19):1782–1790, 2011.
5. S.-S. Choi and J.H. Kim, Optimal query complexity bounds for finding graphs, *Artificial Intelligence*, 174(9-10):551–569, 2010.
6. L. Dall'Asta, J.I. Alvarez-Hamelin, A. Barrat, A. Vázquez, and A. Vespignani, Exploring networks with traceroute-like probes: Theory and simulations, *Theoretical Computer Science* 355(1):6–24, 2006,
7. T. Erlebach, A. Hall, and M. Mihal'ák, Approximate discovery of random graphs, *Proc. of the 4th Int. Symp. on Stochastic Algorithms: Foundations and Applications (SAGA'07)*, Vol. 4665 of LNCS, 82–92, 2007.
8. M.R. Garey and D.S. Johnson, *Computers and intractability: a guide to the theory of NP-completeness*, W.H. Freeman & Co., New York, NY, USA, 1979.
9. R. Govindan and H. Tangmunarunkit, Heuristics for Internet map discovery, *Proc. of the 19th IEEE Int. Conf. on Computer Communications (INFOCOM'00)*, 1371–1380, 2000.
10. S. Gravier, R. Klasing, and J. Moncel, Hardness results and approximation algorithms for identifying codes and locating-dominating codes in graphs, *Algorithmic Operations Research*, 3:43–50, 2008.
11. S. Guha and S. Khuller, Approximation algorithms for connected dominating sets, *Algorithmica*, 20:374–387, 1998.
12. F. Harary, A characterization of block graphs, *Canadian Mathematical Bulletin*, 6(1):1–6, 1963.
13. F. Harary, *Graph Theory*, Addison-Wesley, Reading, MA, USA, 1969.
14. F. Harary and R. Melter, The metric dimension of a graph, *Ars Combinatoria*, 191–195, 1976.

15. A.J. Hoffman and R.R. Singleton, Moore graphs with diameter 2 and 3, *IBM Journal of Research and Development*, 5(4):497–504, 1960.
16. J. Hopcroft and R.E. Tarjan, Efficient algorithms for graph manipulation, *Communication of the ACM*, 16:372–378, 1973.
17. S. Khuller, B. Raghavachari and A. Rosenfeld, Landmarks in graphs, *Discrete Applied Mathematics*, 70:217–229, 1996.
18. H. Mazzawi, Optimally reconstructing weighted graphs using queries, *Proc. of the 21st Annual ACM-SIAM Symp. on Discrete Algorithms (SODA'10)*, 608–615, 2010.
19. L. Reyzin and N. Srivastava, Learning and verifying graphs using queries with a focus on edge counting, *Proc. of the 18th Int. Conf. on Algorithmic Learning Theory (ALT'07)*, Vol. 4754 of LNCS, 285–297, 2007.
20. S. Sen and V.N. Muralidhara, The covert set-cover problem with application to network discovery, *Proc. of the 4th Int. Workshop on Algorithms and Computation (WALCOM'10)*, Vol. 5942 of LNCS, 228–239, 2010.
21. J. Suomela, Approximability of identifying codes and locating-dominating codes. *Information Processing Letters*, 103(1):28–33, 2007.