In *Proceedings of the 31st International Workshop on Graph-Theoretic Concepts in Computer Science* (WG 2005), LNCS 3787, 2005, pp. 127-138. ©Springer-Verlag

Network Discovery and Verification*

Zuzana Beerliova[1], Felix Eberhard[1], Thomas Erlebach[2], Alexander Hall[1], Michael Hoffmann[2], Matúš Mihaľák[2], and L. Shankar Ram[1]

[1] Department of Computer Science, ETH Zürich
{bzuzana,mhall,lshankar}@inf.ethz.ch
[2] Department of Computer Science, University of Leicester
{te17,mh55,mm215}@mcs.le.ac.uk

Abstract. Consider the problem of discovering (or verifying) the edges and non-edges of a network, modeled as a connected undirected graph, using a minimum number of queries. A query at a vertex v discovers (or verifies) all edges and non-edges whose endpoints have different distance from v. In the network discovery problem, the edges and non-edges are initially unknown, and the algorithm must select the next query based only on the results of previous queries. We study the problem using competitive analysis and give a randomized on-line algorithm with competitive ratio $O(\sqrt{n \log n})$ for graphs with n vertices. We also show that no deterministic algorithm can have competitive ratio better than 3. In the network verification problem, the graph is known in advance and the goal is to compute a minimum number of queries that verify all edges and non-edges. This problem has previously been studied as the problem of placing landmarks in a graph or determining the metric dimension of a graph. We show that there is no approximation algorithm for this problem with ratio $o(\log n)$ unless $\mathcal{P} = \mathcal{NP}$.

1 Introduction

In recent years, there has been an increasing interest in the study of networks whose structure has not been imposed by a central authority but arisen from local and distributed processes. Prime examples of such networks are the Internet and unstructured peer-to-peer networks such as Gnutella. For these networks, it is very difficult and costly to obtain a "map" providing an accurate representation of all nodes and the links between them. Such maps would be useful for many purposes, e.g., for studying routing aspects or robustness properties.

In order to create maps of the Internet, a commonly used technique is to obtain local views of the network from various locations (vantage points) and combine them into a map that is hopefully a good approximation of the real network [2, 13]. More generally, one can view this technique as an approach for discovering the topology of an unknown network by using a certain type of queries—a query corresponds to asking for the local view of the network from

* Research partially supported by the EU within the 6th Framework Programme under contract 001907 (DELIS).

one specific vantage point. In this paper, we formalize *network discovery* as a combinatorial optimization problem whose goal is to minimize the number of queries required to discover all edges and non-edges of the network. We study the problem as an on-line problem using competitive analysis. Initially, the network is unknown to the algorithm. To decide the next query to ask, the algorithm can only use the knowledge about the network it has gained from the answers of previously asked queries. In the end, the number of queries asked by the algorithm is compared to the optimal number of queries sufficient to discover the network. We consider a query model in which the answer to a query at a vertex v consists of all edges and non-edges whose endpoints have different (graph-theoretic) distance from v.

In the off-line version of the network discovery problem, the network is known to the algorithm from the beginning. The goal is to compute a minimum number of queries that suffice to discover the network. Although an algorithm for this off-line problem would not be useful for network discovery (if the network is known in advance, there is no need to discover it), it could be employed for network verification, i.e., for checking whether a given map is accurate. Thus, we refer to the off-line version of network discovery as *network verification*. Here, we are interested in polynomial-time optimal or approximation algorithms.

Motivation. As mentioned above, the motivation for our research comes from the problem of discovering information about the topology of communication networks such as the Internet or peer-to-peer networks. The query model that we study is motivated by the following considerations. First, notice that our query model can be interpreted in the following way: A query at v yields the shortest-path subgraph rooted at v, i.e., the set of all edges on shortest paths between v and any other vertex. To see that this is equivalent to our definition (where a query yields all edges and non-edges between vertices of different distance from v), note that an edge connects two vertices of different distance from v if and only if it lies on a shortest path between v and one of these two vertices. Furthermore, the shortest-path subgraph rooted at v implicitly confirms the absence of all edges between vertices of different distance from v.

Real-life scenarios where the shortest-path subgraph rooted at a node of the network can be determined arise as follows. With traceroute tools, one can determine the path that packets take in the Internet if they are sent from one's node to some destination. If each traceroute experiment returns a random shortest path to the destination, one could use repeated traceroute experiments to all destinations to discover all edges of the shortest-path subgraph. Making a query at v would mean getting access to node v and running repeated traceroute experiments from v to all other nodes. If we assume that the cost of getting access to a node is much higher than that of running the traceroute-experiments, minimizing the number of queries is a meaningful goal. Along similar lines, in a network that routes all packets along arbitrary shortest paths, one could imagine a routing protocol in which each node stores the shortest-path subgraph rooted at that node. In this case, reading out the routing table at a node would correspond to making a query at that node.

Our model of network discovery is a simplification of reality. In real networks, routing is not necessarily along shortest paths, but may be affected by routing policies, link qualities, or link capacities. Furthermore, routing tables or traceroute experiments will often reveal only a single path (or at most a few different paths) to each destination, but not the whole shortest-path subgraph. Nevertheless, we believe that our model is a good starting point for a theoretical investigation of fundamental issues arising in network discovery.

Related Work. Graph discovery problems have been studied in distributed settings where one or several agents move along the edges of the graph (see, e.g., [3]); the problems arising in such settings appear to require very different techniques from the ones in our setting.

It turns out, however, that the network verification problem has previously been considered as the problem of placing landmarks in graphs [9]. Here, the motivation is to place landmarks in as few vertices of the graph as possible in such a way that each vertex of the graph is uniquely identified by the vector of its distances to the landmarks. The smallest number of landmarks that are required for a given graph G is also called the *metric dimension* of G [8]. For a survey of known results, we refer to [5]. Results for the problem variant where extra constraints are imposed on the set of landmarks (e.g., connectedness or independence) are surveyed in [11].

The problem of determining whether k landmarks suffice (i.e., of determining if the metric dimension is at most k) is $\mathcal{NP}$-complete [6]; see [9] for an explicit proof by reduction from 3-SAT. In [9] it is also shown that the problem of minimizing the number of landmarks admits an $O(\log n)$-approximation algorithm for graphs with n vertices, based on SetCover. For trees, they show that the problem can be solved optimally in polynomial time. Furthermore, they prove that one landmark is sufficient if and only if G is a path, and discuss properties of graphs for which 2 landmarks suffice. They also show that if k landmarks suffice for a graph with n vertices and diameter D, we must have $n \leq D^k + k$. For d-dimensional grids they show that d landmarks suffice. For d-dimensional hypercubes, a special case of d-dimensional grids, it was shown in [12] (using an earlier result from [10] on a coin weighing problem) that the metric dimension is asymptotically equal to $2d/\log_2 d$. See also [4] for further results on the metric dimension of Cartesian products of graphs.

Our Results. For network discovery, we give a lower bound showing that no deterministic on-line algorithm can have competitive ratio better than 3, and we present a randomized on-line algorithm with competitive ratio $O(\sqrt{n \log n})$ for networks with n nodes. For the network verification problem, we prove that it cannot be approximated within a factor of $o(\log n)$ unless $\mathcal{P} = \mathcal{NP}$, thus showing that the approximation algorithm from [9] is best possible (up to constant factors). We also give a useful lower bound formula for the optimal number of queries of a given graph. The remainder of the paper is structured as follows. Section 2 gives preliminaries and defines the problems formally. Sections 3 and 4 give our results for network discovery and network verification, respectively. Section 5 points to open problems and promising directions for future research.

2 Preliminaries and Problem Definitions

Throughout this paper, the term *network* refers to a connected, undirected graph. For a given graph $G = (V, E)$, we denote the number of nodes by $n = |V|$ and the number of edges by $m = |E|$. For two distinct nodes $u, v \in V$, we say that $\{u, v\}$ is an *edge* if $\{u, v\} \in E$ and a *non-edge* if $\{u, v\} \notin E$. The set of non-edges of G is denoted by $\bar{E}$. We assume that the set V of nodes is known in advance and that it is the presence or absence of edges that needs to be discovered or verified.

A *query* is specified by a vertex $v \in V$ and called a *query at* v. The query at v is also denoted by v. The answer of a query at v consists of a set E_v of edges and a set $\bar{E}_v$ of non-edges. These sets are determined as follows. Label every vertex $u \in V$ with its distance (number of edges on a shortest path) from v. We refer to sets of vertices with the same distance from v as *layers*. Then E_v is the set of all edges connecting vertices in different layers, and $\bar{E}_v$ is the set of all non-edges whose endpoints are in different layers. Because the query result can be seen as a layered graph, we refer to this query model as the *layered-graph* query model.

A set $Q \subseteq V$ of queries discovers (all edges and non-edges of) a graph $G = (V, E)$ if $\bigcup_{q \in Q} E_q = E$ and $\bigcup_{q \in Q} \bar{E}_q = \bar{E}$. In the off-line case, we also say "verifies" instead of "discovers". The network verification problem is to compute, for a given network G, a smallest set of queries that verifies G. The network discovery problem is the on-line version of the network verification problem. Its goal is to compute a smallest set of queries that discovers G. Here, the edges and non-edges of G are initially unknown to the algorithm, the queries are made sequentially, and the next query must always be determined based only on the answers of previous queries.

We denote by $OPT(G)$, for a given graph G, the cardinality of an optimal query set for verifying G, and by $A(G)$ the cardinality of the query set produced by an algorithm A. The quality of an algorithm is measured by the worst possible ratio $A(G)/OPT(G)$ over all networks G. In the off-line case, an algorithm is a ρ-approximation algorithm (and achieves approximation ratio ρ) if it runs in polynomial time and satisfies $A(G)/OPT(G) \leq \rho$ for all networks G. In the on-line case, an algorithm is ρ-competitive (and achieves competitive ratio ρ) if $A(G)/OPT(G) \leq \rho$ for all networks G. It is weakly ρ-competitive if $A(G) \leq \rho \cdot OPT(G) + c$ for some constant c. If the on-line algorithm is randomized, $A(G)$ is replaced by $\mathbb{E}[A(G)]$ in these definitions. We do not require on-line algorithms to run in polynomial time.

We use LG–ALL–DISCOVERY to refer to the network discovery problem with the layered-graph query model and the goal of discovering all edges and non-edges, and we use LG–ALL–VERIFICATION to refer to its off-line version.

3 Network Discovery

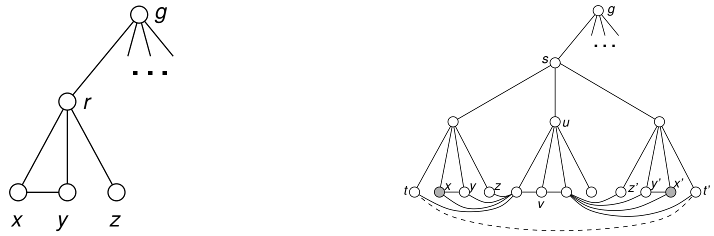
We consider the on-line scenario. Clearly, any algorithm that does not repeat queries has competitive ratio at most $n-1$, since $n-1$ queries are always sufficient to discover a network. Furthermore, the inapproximability result that we will derive in Section 4 (Theorem 3) shows that we cannot hope for a polynomial-time on-line algorithm with competitive ratio $o(\log n)$; it may still be possible to obtain such a ratio using exponential-time on-line algorithms, however. We present a lower bound on the competitive ratio of all deterministic on-line algorithms.

Theorem 1. *No deterministic on-line algorithm for* LG-ALL-DISCOVERY *can have weak competitive ratio* $3 - \varepsilon$ *for any* $\varepsilon > 0$.

Proof. Let A be any deterministic algorithm for LG–ALL–DISCOVERY. We first give a simpler proof that A cannot be better than 2-competitive. Consider Fig. 1(a). We refer to the subgraph induced by the vertices labeled r, x, y, and z as a 2-*gadget*. Assume that the given graph G consists of a global root g and k, $k \geq 2$, disjoint copies of the 2-gadget, with the r-vertex of each 2-gadget connected to the global root g. One can easily verify that $OPT(G) = k$ for this graph, and that the set of all x-vertices of the 2-gadgets constitutes an optimal query set. On the other hand, algorithm A can be forced to make the first query at g (as, initially, the vertices are indistinguishable to the algorithm). This will not discover any information about edges or non-edges between vertices x, y and z of each 2-gadget. The only queries that can discover this information are queries at x, y and z. In fact, a query at x or y suffices to discover the edge between x and y and the non-edges between x and z and between y and z. When A makes the first query among the vertices in $\{x, y, z\}$ of a 2-gadget, we can force it to make that query at z, since the three vertices are indistinguishable to the algorithm. The query at z does not discover the edge between x and y. The algorithm must make a second query in the 2-gadget to discover that edge. In total, the algorithm must make at least $2k+1$ queries. As the construction works for arbitrary values of k, this shows that no deterministic on-line algorithm can guarantee weak competitive ratio $2 - \varepsilon$ for any constant $\varepsilon > 0$.

To get a stronger lower bound of 3, we create a new gadget, called the 3-*gadget*, as shown in Fig. 1(b). The 3-gadget is the subgraph induced by all vertices except g in the figure. We claim that A can be forced to make 6 queries in each 3-gadget, whereas the optimum query set consists of only 2 vertices in each 3-gadget (drawn shaded in the figure). If we construct a graph with k, $k \geq 2$, disjoint copies of the 3-gadget, the s-vertex in each of them connected to the global root g as indicated in the figure, we get a graph G for which we claim that $OPT(G) = 2k$ and the algorithm A can be forced to make at least $6k + 1$ queries. This shows that no deterministic on-line algorithm can guarantee weak competitive ratio $3 - \varepsilon$ for any constant $\varepsilon > 0$.

To see that $OPT(G) = 2k$, let Q be the set of queries consisting of the two shaded vertices from each copy of the 3-gadget as shown in Fig. 1(b). We claim that Q discovers G. This can be verified manually as follows: For every vertex in a 3-gadget Π, consider the 3-tuple whose components are the distances from that vertex to the two query vertices in Π and the distance to an arbitrary query vertex from Q outside Π. One finds that each vertex in Π has a unique 3-tuple, showing that all edges and non-edges of Π are discovered by Q. Each non-edge



(a) (b)

Fig. 1. Lower bound constructions

between two different 3-gadgets is discovered by one of the queries inside these two 3-gadgets. The edges and non-edges between g and each 3-gadget are also discovered. Hence, $OPT(G) \leq 2k$. We have $OPT(G) \geq 2k$, because each of the edges $\{x, y\}$ and $\{x', y'\}$ (see Fig. 1(b)) of a 3-gadget requires a separate query.

To show that $A(G) \geq 6k + 1$, we argue as follows. First, we can force A to make the first query at g. This will not reveal any information about edges within the same layer of any of the 3-gadgets. We view each 3-gadget as consisting of s and a left part, a middle part, and a right part. The left part consists of the left child of s and its four adjacent vertices below (these four vertices are called *bottom vertices*, and the left child of s is called the *root* of that part); the middle and right part are defined analogously. The three parts of a 3-gadget Π are indistinguishable to A until it makes its first query inside Π. A query at s would not discover any new information about Π, so we can ignore queries that A might make at s in the following arguments. When A makes its first query inside Π, we can force this query to be in the middle part, and we can force it to be at u or v. In both cases, the query does not discover any information about the edges and non-edges between the bottom vertices of the left part, nor does it discover any information about the edges and non-edges between the bottom vertices of the right part, nor does it discover the edge drawn dashed. When A chooses its second query in Π, it could be in the left part, in the middle part, or in the right part. Assume that A chooses the left part; since the bottom vertices of the left part are still indistinguishable to A, we can force A to make the query either at the root of the left part or at the bottom vertex t. Similarly, in the right part we can force A to make the query at its root or at t'. In the middle part, A can make the query anywhere. In any case, the second query made by A does not discover any information about edges and non-edges between vertices in the set $\{x, y, z\}$ and in the set $\{x', y', z'\}$. Similarly as in the case of Fig. 1(a), for each of these sets we can force A to make the first query at z (at z') and thus require a second query at x or y (at x' or y') to discover everything about these groups. In total, A must make at least 6 queries in each 3-gadget. □

With the gadget of Fig. 1(a) one can prove easily that no randomized on-line algorithm for LG-ALL-DISCOVERY can have weak competitive ratio $4/3 - \varepsilon$ for any $\varepsilon > 0$; just observe that we can force a randomized algorithm to make the

```
E ← ∅;    /* discovered edges */
N ← ∅;    /* discovered non-edges */
A ← (V choose 2);    /* all pairs of distinct nodes */
/* Phase 1 */
for i = 1 to 3√(n ln n) do
    v ← randomly chosen node from V;
    (E_v, N_v) ← query(v);
    E ← E ∪ E_v;
    N ← N ∪ N_v;
od;
/* Phase 2 */
while E ∪ N ≠ A do
    {u, v} ← an arbitrary element of A \ (E ∪ N);
    (E_u, N_u) ← query(u);
    (E_v, N_v) ← query(v);
    E ← E ∪ E_u ∪ E_v;
    N ← N ∪ N_u ∪ N_v;
    S ← set of nodes from which the (non-)edge {u, v} is discovered;
    foreach x ∈ S \ {u, v} do
        (E_x, N_x) ← query(x);
        E ← E ∪ E_x;
        N ← N ∪ N_x;
    od;
od;
```

Fig. 2. On-line algorithm for LG-ALL-DISCOVERY

first query at z with probability at least $1/3$. Note that all lower bounds on the weak competitive ratio also hold for the (standard) competitive ratio where no additive constant c is allowed.

Theorem 2. *There is a randomized on-line algorithm that achieves competitive ratio $O(\sqrt{n \log n})$ for* LG-ALL-DISCOVERY.

Proof. The on-line algorithm is shown in Fig. 2. In the first phase, it makes $3\sqrt{n \ln n}$ queries at nodes chosen uniformly at random. In the second phase, as long as node pairs with unknown status exist, it picks an arbitrary such pair $\{u, v\}$ and proceeds as follows. First, it queries u and v in order to determine the distance of all nodes to u and v. From this it can deduce the set S of nodes from which the edge or non-edge between u and v can be discovered; these are simply the nodes for which the distance to u differs from the distance to v. Then, it queries all remaining nodes in S.

To analyze the algorithm, it is helpful to view LG-ALL-DISCOVERY as a HITTINGSET problem. For every edge or non-edge $\{u, v\}$, let S_{uv} be the set of nodes from which a query discovers $\{u, v\}$. The task of the LG-ALL-DISCOVERY problem translates into the task of computing a subset of V that hits all sets S_{uv}. The goal of the first phase is to hit all sets that have size at least $\sqrt{n \ln n}$

with high probability. If this succeeds, the problem remaining for the second phase is a HittingSet problem where all sets have size at most $\sqrt{n \ln n}$. The algorithm of the second phase repeatedly picks an arbitrary set that is not yet hit, and includes all its elements in the solution. As the sets have size at most $\sqrt{n \ln n}$, the number of queries made in the second phase is at most a factor of $\sqrt{n \ln n}$ away from the optimum.

Let us make this analysis precise. Consider a node pair $\{u, v\}$ for which the set S_{uv} has size at least $\sqrt{n \ln n}$. In each query of the first phase, the probability that S_{uv} is not hit is at most $1 - \frac{\sqrt{n \ln n}}{n} = 1 - \frac{\sqrt{\ln n}}{\sqrt{n}}$. Thus, the probability that S_{uv} is not hit throughout the first phase is at most $\left(1 - \frac{\sqrt{\ln n}}{\sqrt{n}}\right)^{3\sqrt{n \ln n}} \le e^{-3 \ln n} = \frac{1}{n^3}$. There are at most $\binom{n}{2}$ sets S_{uv} of cardinality at least $\sqrt{n \ln n}$. The probability that at least one of them is not hit in the first phase is at most $\binom{n}{2} \cdot \frac{1}{n^3} \le \frac{1}{n}$.

Now consider the second phase, conditioned on the event that the first phase has hit all sets S_{uv} of size at least $\sqrt{n \ln n}$. In each iteration of the while-loop of the second phase, the algorithm asks at most $\sqrt{n \ln n}$ queries. Let ℓ be the number of iterations. It is clear that the optimum must make at least ℓ queries, because no two unknown pairs $\{u, v\}$ considered in different iterations of the second phase can be resolved by the same query.

Since $OPT(G) \ge 1$ and $OPT(G) \ge \ell$, the number of queries made by the algorithm is at most $3\sqrt{n \ln n} + \ell\sqrt{n \ln n} = O(\sqrt{n \log n}) \cdot OPT(G)$.

With probability at least $1 - \frac{1}{n}$, the first phase succeeds and the algorithm makes $O(\sqrt{n \log n}) \cdot OPT(G)$ queries. If the first phase fails, the algorithm makes at most n queries. This case increases the expected number of queries made by the algorithm by at most $\frac{1}{n} \cdot n = 1$. Thus, the expected number of queries is at most $O(\sqrt{n \log n}) \cdot OPT(G) + \frac{1}{n} \cdot n = O(\sqrt{n \log n}) \cdot OPT(G)$. □

4 Network Verification

Theorem 3. *It is $\mathcal{NP}$-hard to approximate* LG–ALL–Verification *within ratio* $o(\log n)$.

Proof. We prove the inapproximability result using an approximation-preserving reduction from the *test collection problem (TCP)*:

Problem TCP
Input: ground set S and collection $\mathcal{C}$ of subsets of S
Feasible solution: subset $\mathcal{C}' \subseteq \mathcal{C}$ such that for each two distinct elements x and y of S, there exists a set $C \in \mathcal{C}'$ such that exactly one of x and y is in C.
Objective: minimize the cardinality of $\mathcal{C}'$

In the original application for TCP, S is a set of diseases and $\mathcal{C}$ is a collection of tests. A test $C \in \mathcal{C}$, applied to a patient, will give a positive result if the patient is infected by a disease in C. If a patient is known to be infected by exactly one of the diseases in S, the goal of TCP is to compute a minimum number of tests that together can uniquely identify that disease.

Without loss of generality, we can restrict ourselves to instances of TCP in which any two elements of the ground set can be separated by at least one of the sets in $\mathcal{C}$; instances without this property do not have any feasible solutions.

Halldórsson et al. [7] prove that TCP cannot be approximated with ratio $o(\log |S|)$ unless $\mathcal{P} = \mathcal{NP}$. Their proof uses an approximation-preserving reduction from SetCover; the latter problem was shown $\mathcal{NP}$-hard to approximate within $o(\log n)$, where n is the cardinality of the ground set, by Arora and Sudan [1]. The proof by Arora and Sudan establishes the inapproximability result for SetCover even for instances in which the size of the ground set and the number of sets are polynomially related. The reduction from SetCover to TCP maintains this property. Hence, we know that it is $\mathcal{NP}$-hard to approximate TCP with ratio $o(\log |S|)$ even for instances satisfying $|\mathcal{C}| \leq |S|^g$ for some positive constant g.

Let an instance $(S, \mathcal{C})$ of TCP be given. Let $n_{\mathrm{TCP}} = |S|$ and $m_{\mathrm{TCP}} = |\mathcal{C}|$. By the remark above, we can assume that $m_{\mathrm{TCP}} = n_{\mathrm{TCP}}^{O(1)}$. We construct an instance $G = (V, E)$ of LG–ALL–Verification as follows. First, we add $n_{\mathrm{TCP}} + m_{\mathrm{TCP}}$ vertices to V: an *element vertex* v_s for every element $s \in S$ and a *test vertex* u_C for every $C \in \mathcal{C}$. We initially add the following edges to E: Any two element vertices are joined by an edge, and every test vertex u_C is joined to all element vertices v_s with $s \in C$. The idea behind this construction is that queries at test vertices verify all edges in the clique of element vertices if and only if the corresponding tests form a test cover. We have to extend the construction slightly since, in LG–ALL–Verification, the edges and non-edges incident to the test vertices need to be verified as well. We add $h = 2(\lceil \log m_{\mathrm{TCP}} \rceil + 2)$ auxiliary vertices $w_1, \ldots, w_h$ to take care of this. For each i, $1 \leq i \leq h/2$, the auxiliary vertices w_{2i-1} and w_{2i} are said to form a *pair*. In addition, we add one extra node z. We add the following edges:

- The two auxiliary vertices in each pair are joined by an edge.
- Number the m_{TCP} test vertices arbitrarily from 0 to $m_{\mathrm{TCP}} - 1$. Both auxiliary vertices in the i-th pair, $1 \leq i \leq h/2 - 2$, are joined to those of the m_{TCP} test vertices whose number has a 1 in the i-th position of its binary representation.
- Both auxiliary vertices in the last two pairs are joined to all test vertices.
- The extra node z is joined to all other vertices of the graph.

The graph constructed in this way is denoted by $G = (V, E)$. See Fig. 3 for an illustration. We prove two claims:

Claim 1. Given a solution $\mathcal{C}'$ to the TCP instance $(S, \mathcal{C})$, there is a solution Q of the constructed instance $G = (V, E)$ of LG–ALL–Verification satisfying $|Q| = |\mathcal{C}'| + \lceil \log m_{\mathrm{TCP}} \rceil + 2$.

Proof (of Claim 1). Let a solution $\mathcal{C}'$ to the TCP instance $(S, \mathcal{C})$ be given. Let Q contain all test vertices corresponding to sets $C \in \mathcal{C}'$ as well as the first vertex of every pair of auxiliary vertices. Obviously, we have $|Q| = |\mathcal{C}'| + \lceil \log m_{\mathrm{TCP}} \rceil + 2$. It is not difficult to verify that Q discovers all edges and non-edges of G. $\square$

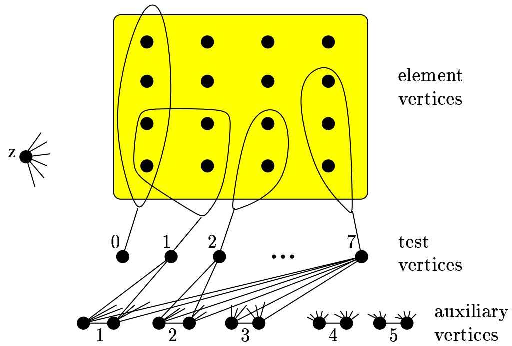


Fig. 3. Illustration of the construction of the graph $G = (V, E)$ that is an instance of LG–ALL–VERIFICATION. The auxiliary vertices in pairs 4 and 5 are adjacent to all test vertices. The auxiliary vertices in pair i, $1 \le i \le 3$, are adjacent to the test vertices whose number has a 1 in position i of the binary representation. For example, the auxiliary vertices in pair 2 are adjacent to test vertices $2, 3, 6$ and 7

Claim 2. Given a solution Q to the constructed instance $G = (V, E)$ of LG–ALL–VERIFICATION, one can construct in polynomial time a solution $\mathcal{C}'$ of the original TCP instance $(S, \mathcal{C})$ satisfying $|\mathcal{C}'| \le |Q| - \lceil \log m_{\mathrm{TCP}} \rceil - 2$.

Proof (of Claim 2). Observe that Q must contain at least one vertex from each pair of auxiliary vertices; otherwise, the edge joining this pair would not be discovered. The queries at these vertices do not discover any edges between element vertices (all element vertices are at distance 2 from any auxiliary vertex because of the extra vertex z). Let Q' be the vertices in Q that are not auxiliary vertices. We have $|Q'| \le |Q| - \lceil \log m_{\mathrm{TCP}} \rceil - 2$. Now, Q' is a set of element vertices and test vertices that, in particular, discovers all edges between element vertices.

Let Q_S be the set of element vertices in Q' and let $Q_{\mathcal{C}}$ be the set of test vertices in Q'. If Q_S is empty, the queries at the vertices in $Q_{\mathcal{C}}$ discover all edges of the clique of element vertices. In particular, this means that for any two distinct element vertices v_s and v_t in V, there must be a query at a vertex adjacent to one of v_s, v_t but not to the other. This shows that the set $\mathcal{C}' = \{C \in \mathcal{C} \mid u_C \in Q'\}$ is a solution of the original TCP instance of the required size.

Now assume Q_S is nonempty. The set of edges between element vertices that are not discovered by $Q_{\mathcal{C}}$ is a disjoint union of cliques. The queries in Q_S must discover all edges in these cliques. As the only edges between element vertices that a query at an element vertex discovers are the edges incident to that vertex, a clique of size k requires $k-1$ queries. Assume that there are p cliques and denote the number of vertices in these cliques by $k_1, \ldots, k_p$. Then Q_S contains at least $\sum_{i=1}^{p}(k_i - 1)$ vertices. All edges in a clique of size k can always be discovered by $k-1$ queries at test vertices: simply select these queries greedily by choosing, as long as there is an edge $\{u, v\}$ in the clique that has not yet been discovered, any test vertex that is adjacent to one of u, v but not the other. Hence, we can replace the queries in Q_S by at most $\sum_{i=1}^{p}(k_i - 1)$ queries at test vertices and add these to Q_C, obtaining a set of queries at test vertices that discovers all edges between element vertices. As in the previous paragraph, this set of test vertices gives a solution to the original TCP instance of cardinality at most $|Q'|$. □

Assume there is an approximation algorithm A for LG–ALL–VERIFICATION that achieves ratio $o(\log n)$, where $n = |V|$. Consider the algorithm B for TCP that, given an instance of TCP, constructs an instance of LG–ALL–VERIFICATION as described above, applies A to this instance, and transforms the result into a solution to the TCP instance following Claim 2. Recall that $m_{\mathrm{TCP}} = n_{\mathrm{TCP}}^{O(1)}$. We claim that B achieves ratio $o(\log n_{\mathrm{TCP}})$ for TCP. Let OPT_{TCP} be the optimum objective value for the given TCP instance and OPT_{LG} be the optimum objective value for the constructed instance of LG–ALL–VERIFICATION. Let B_{TCP} and A_{LG} denote the objective values of the solutions computed by B and A, respectively. Note that $OPT_{\mathrm{TCP}} \geq \log n_{\mathrm{TCP}}$ always holds, since n_{TCP} elements cannot be separated by fewer than $\log n_{\mathrm{TCP}}$ test sets.

Claims 1 and 2 imply that $OPT_{\mathrm{TCP}} = OPT_{\mathrm{LG}} - \lceil \log m_{\mathrm{TCP}} \rceil - 2$. We have $OPT_{\mathrm{LG}} = OPT_{\mathrm{TCP}} + \lceil \log m_{\mathrm{TCP}} \rceil + 2 \leq OPT_{\mathrm{TCP}} + O(\log n_{\mathrm{TCP}}) = O(OPT_{\mathrm{TCP}})$. Claim 2 implies $B_{\mathrm{TCP}} \leq A_{\mathrm{LG}}$ and thus we get $B_{\mathrm{TCP}} \leq o(\log n) \cdot OPT_{\mathrm{LG}} = o(\log n) \cdot O(OPT_{\mathrm{TCP}}) = o(\log n_{\mathrm{TCP}}) \cdot O(OPT_{\mathrm{TCP}})$, where the last equality follows from $n = n_{\mathrm{TCP}} + m_{\mathrm{TCP}} + 2(\lceil \log m_{\mathrm{TCP}} \rceil + 2) + 1 = n_{\mathrm{TCP}}^{O(1)}$. This shows $B_{\mathrm{TCP}} \leq o(\log n_{\mathrm{TCP}}) \cdot OPT_{\mathrm{TCP}}$ and completes the proof of Theorem 3. □

Theorem 4. *If a graph $G = (V, E)$ contains a subgraph H of diameter D_H with n_H vertices, then $OPT(G) \geq \log_{D_H+1} n_H$.*

Proof. Imagine the queries being performed sequentially. At any instant, the unknown edges and non-edges induce disjoint cliques, which we call *unknown groups*. Two vertices are in the same unknown group if and only if they were in the same layer of all queries made so far. Consider the n_H vertices of subgraph H. Initially, all vertices form an unknown group. For each query, the n_H vertices of H will be in at most $D_H + 1$ consecutive layers of the layered graph returned by the query. Therefore, after the first query, at least $n_H/(D_H + 1)$ vertices of H will still be in the same unknown group. Similarly, after k queries, at least $n_H/(D_H + 1)^k$ vertices of H will be in an unknown group together. If k queries suffice to verify all edges and non-edges, the unknown groups must be singletons in the end. So we must have $n_H/(D_H + 1)^k \leq 1$. This proves the theorem. □

This theorem implies that a graph containing a clique on k vertices requires at least $\log_2 k$ queries, and a graph with maximum degree Δ at least $\log_3(\Delta+1)$ queries. For the former, take H to be the clique on k vertices, and for the latter, take H to be the subgraph induced by a vertex of degree Δ and its neighbors.

5 Directions for Future Work

In this paper, we have considered network discovery and network verification problems in the layered-graph query model. The goal was to discover or verify all edges and non-edges of a network. For network discovery, the major problem left open by our work is to close the gap between our randomized upper bound of $O(\sqrt{n \log n})$ and the small constant lower bounds.

The subject of our study is an example of a family of problem settings in which the goal is to discover or verify information about a graph using queries. Different problems are obtained if the query model is varied, or if the objective is changed. Other natural query models are, e.g., that a query at v returns only the distances from v to all other vertices of the graph; that a query is specified by two vertices u and v, and returns the set of all edges on shortest paths between u and v; or that a query returns an arbitrary shortest-path tree rooted at v. Concerning the objective, the goal could be to discover or verify a certain graph parameter such as diameter, average path length, or independence number. One could also relax the requirement and only ask for an approximate answer, e.g., one could consider the problem of minimizing the number of queries required to approximate the average path length within a factor of $1 + \varepsilon$. We believe that the study of such problems could be a fruitful area of research with applications in the monitoring and analysis of communication networks such as the Internet.

References

1. S. Arora and M. Sudan. Improved low-degree testing and its applications. In *Proc. 29th Ann. ACM Symp. on Theory of Computing (STOC'97)*, pages 485–495, 1997.
2. P. Barford, A. Bestavros, J. Byers, and M. Crovella. On the marginal utility of deploying measurement infrastructure. In *Proc. ACM SIGCOMM Internet Measurement Workshop 2001*, November 2001.
3. M. A. Bender and D. K. Slonim. The power of team exploration: Two robots can learn unlabeled directed graphs. In *Proc. 35th Ann. IEEE Symp. on Foundations of Computer Science (FOCS'94)*, pages 75–85, 1994.
4. J. Cáceres, C. Hernando, M. Mora, I. M. Pelayo, M. L. Puertas, C. Seara, and D. R. Wood. On the metric dimension of Cartesian products of graphs. Manuscript, 2005.
5. G. Chartrand and P. Zhang. The theory and applications of resolvability in graphs: A survey. *Congr. Numer.*, 160:47–68, 2003.
6. M. R. Garey and D. S. Johnson. *Computers and Intractability. A Guide to the Theory of NP-Completeness.* W. H. Freeman and Company, New York, 1979.
7. B. V. Halldórsson, M. M. Halldórsson, and R. Ravi. On the approximability of the minimum test collection problem. In *Proc. 9th Ann. European Symposium on Algorithms (ESA'01)*, LNCS 2161, pages 158–169. Springer-Verlag, 2001.
8. F. Harary and R. Melter. The metric dimension of a graph. *Ars Combin.*, 2:191–195, 1976.
9. S. Khuller, B. Raghavachari, and A. Rosenfeld. Landmarks in graphs. *Discrete Appl. Math.*, 70:217–229, 1996.
10. B. Lindström. On a combinatory detection problem I. *Magyar Tud. Akad. Mat. Kutató Int. Közl.*, 9:195–207, 1964.
11. V. Saenpholphat and P. Zhang. Conditional resolvability in graphs: A survey. *Int. J. Math. Math. Sci.*, 38:1997–2017, 2004.
12. A. Sebő and E. Tannier. On metric generators of graphs. *Math. Oper. Res.*, 29(2):383–393, 2004.
13. L. Subramanian, S. Agarwal, J. Rexford, and R. Katz. Characterizing the Internet hierarchy from multiple vantage points. In *INFOCOM'02*, June 2002.