Distributed Testing of Graph Isomorphism in the CONGEST Model

Reut Levi
Efi Arazi School of Computer Science, The Interdisciplinary Center, Herzliya, Israel
reut.levi1@idc.ac.il

Moti Medina
School of Electrical & Computer Engineering, Ben-Gurion University of the Negev,
Beer Sheva, Israel
medinamo@bgu.ac.il

Abstract

In this paper we study the problem of testing graph isomorphism (GI) in the CONGEST distributed model. In this setting we test whether the distributive network, G_U, is isomorphic to G_K which is given as an input to all the nodes in the network, or alternatively, only to a single node.

We first consider the *decision* variant of the problem in which the algorithm should distinguish the case where G_U and G_K are isomorphic from the case where G_U and G_K are not isomorphic. Specifically, if G_U and G_K are not isomorphic then w.h.p. at least one node should output reject and otherwise all nodes should output accept. We provide a randomized algorithm with $O(n)$ rounds for the setting in which G_K is given only to a single node. We prove that for this setting the number of rounds of any deterministic algorithm is $\tilde{\Omega}(n^2)$ rounds, where n denotes the number of nodes, which implies a separation between the randomized and the deterministic complexities of deciding GI. Our algorithm can be adapted to the semi-streaming model, where a single pass is performed and $\tilde{O}(n)$ bits of space are used.

We then consider the *property testing* variant of the problem, where the algorithm is only required to distinguish the case that G_U and G_K are isomorphic from the case that G_U and G_K are *far* from being isomorphic (according to some predetermined distance measure). We show that every (possibly randomized) algorithm, requires $\Omega(D)$ rounds, where D denotes the diameter of the network. This lower bound holds even if all the nodes are given G_K as an input, and even if the message size is unbounded. We provide a randomized algorithm with an almost matching round complexity of $O(D + (\epsilon^{-1} \log n)^2)$ rounds that is suitable for dense graphs (namely, graphs with $\Omega(n^2)$ edges).

We also show that with the same number of rounds it is possible that each node outputs its mapping according to a bijection which is an *approximate* isomorphism.

We conclude with simple simulation arguments that allow us to adapt centralized property testing algorithms and obtain essentially tight algorithms with round complexity $\tilde{O}(D)$ for special families of sparse graphs.

2012 ACM Subject Classification Theory of computation → Distributed algorithms; Theory of computation → Graph algorithms analysis

Keywords and phrases the CONGEST model, graph isomorphism, distributed property testing, distributed decision, graph algorithms

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2020.19

Category RANDOM

Related Version A full version of the paper is available at `https://arxiv.org/abs/2003.00468`.

Funding *Reut Levi*: This research was supported by the ERC advanced grant 834735.
Moti Medina: This research was supported by the Israel Science Foundation under Grant 867/19.

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2020).
Editors: Jarosław Byrka and Raghu Meka; Article No. 19; pp. 19:1–19:24
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Testing graph isomorphism is one of the most fundamental computational problems in graph theory. A pair of graphs G and H are isomorphic if there is a bijection that maps the nodes of G to the nodes of H such that every edge of G is mapped to an edge of H and likewise for non-edges. Currently, it is not known whether there exists an efficient algorithm for this problem and in fact it is one of the few natural problems which is a candidate for being in NP-intermediate, that is, neither in P nor NP-complete. In order to obtain efficient algorithms for this problem, various restrictions and relaxations were considered (e.g. [34, 30]). This problem has been extensively studied also in other computational models such as parallel computation models [29, 41, 10, 32, 46, 35, 24, 9] and in the realm of property testing in which the main complexity measure is the query complexity [17, 43, 25, 42, 38, 3].

In the context of distributive models such as the CONGEST [44] and the LOCAL [40] models, the main complexity measure is the round complexity and the computational power is usually considered to be unbounded. Therefore in these models the complexity of the problem may change dramatically. While there seem to be many sensible settings, one of the simplest settings of the problem for distributive models is to test for isomorphism between the distributed network, G_U, and a known graph, G_K, which is given as an input to all the nodes in the network, or alternatively, only to a subset of the nodes [1]. The requirement from the algorithm is that if G_K and G_U are isomorphic, then with high probability [2] all nodes should output accept and that at least one node should output reject otherwise.

Since the property of being isomorphic to a specific graph is inherently global, intuitively we expect the round complexity to be $\Omega(D)$ where D denotes the diameter of the network (even for the case in which G_K is given as an input to all the nodes in the network). As we show, this intuition is correct even for the LOCAL model, in which there is no bound on the message size. Therefore, in the LOCAL model, it is not possible to improve over the trivial algorithm that collects the entire information on the network at a single node in $O(D)$ rounds and tests for graph isomorphism in a centralized manner. In the CONGEST model, in which the message-size is bounded by $O(\log n)$, where n denotes the number of nodes in the network, implementing this trivial solution may require $O(n^2)$ rounds. This leads to the obvious question whether is it possible to obtain round complexity which is better than $O(n^2)$ in the CONGEST model.

Another interesting question is whether we can obtain better bounds if we relax the decision problem (as considered in the realm of *property testing*) such that the algorithm is only required to distinguish between pairs of graphs which are isomorphic and pairs of graphs which are *far* from being isomorphic (according to some predetermined distance measure).

In this setting we define the problem as follows. Let G_U be the distributed network and let m denote the number of edges in the network or an upper bound on this number. We say that a pair of graphs are ϵ-far from being isomorphic if ϵm edges need to be deleted/inserted in order to make the graphs isomorphic, where m denotes the number of edges in the network. An adversarially chosen node, r, receives as an input the graph G_K and a proximity parameter $\epsilon \in (0, 1)$. The requirement from the algorithm is as follows. If G_K and G_U are isomorphic, then w.h.p. all nodes should output accept. If G_K and G_U are ϵ-far from being isomorphic, then w.h.p. at least one node should output no.

[1] This formulation in which G_K is a parameter falls into the category of massively parameterized problems and is also considered in the setting of property testing [17, 25].

[2] We say that an algorithm succeed with high probability, if it succeeds with probability at least $1 - 1/n^c$ for any constant c (without changing the round complexity asymptotically).

1.1 Our Results

In this section we outline our results. We further elaborate on our results in the next sections. In all that follows, unless explicitly stated otherwise, when we refer to distributed algorithms, we mean in the CONGEST model.

A Decision Algorithm and a Lower Bound for the Decision Problem. For the (exact) decision problem we provide a randomized one-sided error algorithm that runs in $O(n)$ rounds and succeeds with high probability (see Theorem 15). The algorithm works even for the setting in which G_K is given only to a single node (that may be chosen adversely). For this setting we prove that any deterministic algorithm requires $\tilde{\Omega}(n^2)$ rounds, which implies a separation between the randomized and the deterministic complexity of the decision problem (see Theorem 17 in Appendix A). We note that our algorithm can be adapted to the semi-streaming model [15] in which it uses $\tilde{O}(n)$ bits of space and performs only one pass (see Theorem 16).

A Lower Bound for the Property Testing Variant. For property testing algorithms we show that even under this relaxation $\Omega(D)$ rounds are necessary even for constant ϵ and constant error probability. This lower bound holds even in the LOCAL model, for two-sided error algorithms, and even if all the nodes receive G_K as an input (see Theorem 18 in Appendix A). It also holds for dense graphs, namely, when $m = \Theta(n^2)$ and for sparse graphs, that is when $m = \Theta(n)$.

A Property Testing Algorithm and Computation of Approximate Isomorphism. We provide a distributed two-sided error property testing algorithm that runs in $O(D + (\epsilon^{-1} \cdot \log n)^2)$ rounds and succeeds with high probability for the case that $m = \Theta(n^2)$, implying that our result is tight up to an additive term of $(\epsilon^{-1} \cdot \log n)^2$ (see Theorem 1). This algorithm works even in the setting in which G_K is given only to a single node. We note that the graphs that are constructed for the lower bound for the exact variant are dense and have a constant diameter. Therefore for these graphs, the property testing algorithm runs in only $O((\epsilon^{-1} \cdot \log n)^2)$ rounds (while the decision algorithm runs in $O(n)$ rounds).

If G_K is given to all the nodes and the graphs are indeed isomorphic then we show that we can also approximately recover the isomorphism with the same round complexity as of testing. Specifically, each node v outputs $g(v)$ where g is a bijection such that the graph $g(G_U)$, namely the graph in which we re-name the nodes according to g, is ϵ-close to G_K.

Simulation Arguments and their application to special families of sparse graphs. Finally, we show, by simple simulation arguments, that it is possible to obtain essentially tight algorithms with $\tilde{O}(D)$ round complexity for special families of sparse graphs by adapting centralized property testing algorithms. In particular, these algorithms apply for bounded-degree minor-free graphs and general outerplanar graphs (see Appendix B) .

1.2 The Decision Algorithm

As described above, a naive approach for testing isomorphism to G_K is to gather the entire information on the network at a single node and then to test for isomorphism in a centralized manner. By the brute-force approach, we may go over all possible bijections between the nodes of the graphs and test for equality between the corresponding graphs. Our algorithm follows this approach with the difference that it only gathers a compressed version of the network as in the algorithm of Abboud et al. [2] for the Identical Subgraph Detection problem. The idea of their algorithm is to reduce the problem of testing if two graphs are equal to the problem of testing equality between a pair of binary strings. From the fact that the test for equality has a one-sided error, namely it never rejects identical graphs, it follows that our algorithm never rejects isomorphic graphs. To ensure that our algorithm is sound we amplify the success probability of the equality test by repeating the test $\Theta(n)$ times and, as a result, obtain a total round complexity of $O(n)$.

1.3 A Lower Bound for the Decision Problem

We reduce Set-Equality to the problem of deciding isomorphism in the setting in which only a single node receives G_K as an input (as it is the case for our upper bound). The idea is to construct a graph $G_{x,y}$ over n nodes for every pair of strings $x, y \in \{0,1\}^k$ where $k = \Theta(n^2)$ such that $G_{x,y}$ is isomorphic to $G_{x',y'}$ if and only if $x = x'$ and $y = y'$. Let x and y denote the input of Alice and Bob, respectively. In the reduction, G_K is known to Alice and is taken to be $G_{x,x}$. Alice and Bob simulate the distributed algorithm on the graph $G_{x,y}$, which by construction is isomorphic to $G_{x,x}$ if and only if $x = y$, as desired. This reduction yields a lower bound of $\Omega(n^2/\log n)$ rounds for any deterministic algorithm.

1.4 A High-Level Description of the Property Testing Algorithm

Our algorithm closely follows the approach taken by Fischer and Matsliah [17] for testing graph isomorphism in the dense-graph model [26] with two sided-error. However, in order to obtain a round complexity which only depends poly-logarithmically in n (rather than a dependency of $\tilde{O}(\sqrt{n})$ as the query complexity in [17]), we need to diverge from their approach as described next.

1.4.1 The Algorithm of Fischer-Matsliah

The algorithm of Fischer-Matsliah receives as an input a graph G_K (over n vertices), referred to as the known graph, and an oracle access to the adjacency matrix of a graph G_U, referred to as the unknown graph. It also receives the proximity parameter ϵ. The algorithm begins with picking, u.a.r., a sequence of $s = \text{poly}(\epsilon^{-1}, \log(n))$ nodes from the unknown graph. The selection of these nodes induces labels for each node in the graph as follows. The label of each node v is a string of s bits where the i-th bit indicates whether v is a neighbor of the i-th node in the sequence. This labeling scheme guarantees that, with high probability, only "similar" nodes, that is, nodes with similar sets of neighbors, might have identical labels. It is not hard to see that if the graphs are isomorphic, then given that we managed to map the nodes in the sequence according to the isomorphism, both graphs should have the same frequency over labels. More surprisingly, it is shown by Fischer and Matsliah that if the nodes in the sequence are mapped according to the isomorphism then it is possible to extend this mapping on-the-fly and obtain, roughly speaking, an approximate isomorphism. In particular, they showed that as long as each node in the graph is mapped to a node with the same label in the other graph (with respect to the mapped sequence), then the obtained function is close to being an isomorphism. This is due to the fact that nodes which are too "different" are likely to have different labels and the fact that similar nodes are exchangeable. Given a candidate for the approximate isomorphism, the problem is then reduced to testing closeness of graphs. Therefore, if the graphs are isomorphic then by going over all possible mappings of the selected sequence (there are only quasi-polynomially many ways to map these nodes) and for each such partial mapping obtaining a candidate by extending the partial mapping as described above, one should be able to obtain a candidate that passes the test, namely a function, f, which is close to being an isomorphism. On the other hand, if the graphs are far from being isomorphic then by definition any bijection gives two graphs which are far from each other. Therefore, these two cases can be distinguished by approximating the Hamming distance of the corresponding adjacency-matrices. In turn, this can be done by selecting random locations (that is, potential edges) and checking the values of both matrices in these locations. Since constructing f entirely would be too costly, one needs to be able to generate f on-the-fly. A crucial point is that its generation can not depend on the selection of the random locations. In other words, its generation should be query-order-oblivious. To this end, in the algorithm of Fischer-Matsliah they first test if the distributions over the labels are close. If so, they can safely generate f on-the-fly while ensuring that there is only little dependency between f and the queries the algorithm makes to f. This is done by simply mapping a node v to a random node in G_K that have the same label as v. [3] The query complexity of testing identity of distributions, which is $\Theta(\sqrt{n})$ [4, 45, 28, 5], dominates the query complexity of the algorithm [4]. As shown in [17], in the centralized setting this algorithm is essentially tight.

1.4.2 Our Algorithm

In the CONGEST model, by straight-forward simulation arguments it follows that one can simulate the algorithm of Fischer-Matsliah in $\tilde{O}(D + \sqrt{n})$ rounds by collecting the answers to the queries of the algorithm at a single node and simulating the centralized algorithm (see Claim 19). A crucial observation for improving this bound is that nodes that have the same label also have at least one neighbor in common (with the only exception of the all-zero label), therefore they can be coordinated by one of their common neighbors. Moreover, it is possible to obtain both samples and access to the frequencies of the labels via these coordinators. Our algorithm proceeds as follows. As in [17] a sequence C of s random nodes is selected and is sent to the entire network (in $O(D)$ rounds). Each node figures out its label and broadcasts this label to its neighbors. The node r, that received G_K as an input, selects a random set of potential edges $(i_1, j_1), \ldots, (i_k, j_k)$, where $k = \mathrm{poly}(\log n, \epsilon^{-1})$. It then broadcasts this set to the entire network. For each potential edge, the information whether it is an actual edge in the network is sent to r. For each label of a node in $I = \{i_1, j_1, \ldots, i_k, j_k\}$, the corresponding coordinator sends to r the frequency of this label. From this point the rest of the computation is done centrally at r. We say that a sequence, P, of nodes in G_K is *good* with respect to C and I if it induces the same frequency of labels as C when restricted to labels of nodes in I. The node r goes over all possible mappings of C to the known graph and looks for good sequences (with respect to C and I). For every good sequence, P, r generates a function f on-the-fly: on query v, f maps v to a random node in V_K which is still unmatched and has the same label as v (w.r.t. P). As we show, from the fact that the sequence is good it follows that f is query-order-oblivious. Let $f(G_U)$ denote the graph obtained from G_U after applying f on V_K. As in the algorithm of Fischer-Matsliah, if the graphs are isomorphic and the sequence P is the mapping of C according to the isomorphism, then $f(G_U)$ is guaranteed

[3] The little dependency between f and the queries that the algorithm makes to f comes from the fact that the frequencies of labels of the two graphs are not necessarily identical as they are only guaranteed to be close (w.h.p.).

[4] In fact, in [17] the failure probability of the test for testing identity of distributions needs to be negligible and consequently the query complexity of this test is $\tilde{\Theta}(\sqrt{n})$.

to be (w.h.p.) ϵ-close to G_K. The set of potential edges is then used to approximate the distance between G_K and $f(G_U)$. This allows us to obtain a significant improvement in the round complexity (over the straight-forward simulation), in terms of n, from $\tilde{O}(\sqrt{n})$ to $O(\log^2 n)$.

1.5 High-level Approach for Computing an Approximate Isomorphism

As described in the previous section, if the graphs are isomorphic then w.h.p. the algorithm finds a sequence P that corresponds to a bijection f such that $f(G_U)$ is guaranteed to be (w.h.p.) ϵ-close to G_K. The algorithm accesses f only on a small set of random locations. It is tempting to try to output $f(v)$ for every v in the network. Assume now that every node in the network knows G_K. If the sequence P is indeed the mapping of C according to an isomorphism then the following naive approach should work. Each coordinator can independently map to V_K the nodes that are assigned to it according to their labels. However, it might be the case that P is not the mapping of C according to any isomorphism (although it passed the test). In particular it might be that it is not good with respect to C and V_U (recall that P is good w.r.t. C and I). In this case we may want the coordinators of the nodes to be coordinated such that they exchange the mapping of nodes with "overflow" and "underflow" labels (that is, labels that are more/less frequent in G_U than in G_K, respectively). Since there might be $O(n)$ labels, such coordination might cause too much congestion. To this end we cluster the labels according to their most significant bit and assign a single coordinator to each cluster. Since there are only $\text{poly}(\epsilon^{-1}, \log(n))$ many clusters, these coordinators can coordinate without causing too much congestion as long as they do not need to communicate too much information. The naive approach of sending the IDs of the nodes that are still un-matched is too costly as there might be a small but a constant fraction of such nodes. Alternatively, we show how the matching of these nodes can be coordinated such that each coordinator sends only $O(\log n)$ bits. In particular, each coordinator sends a number that indicates by how much it is "under-flowing" or "overflowing". Each coordinator with an under-flow reserve a set of nodes in V_K that are reserved to be the image of nodes that are assigned to other coordinators. The set of reserved nodes can be determined by the above-mentioned numbers (i.e., the numbers which indicate the under-flows and over-flows of the coordinators). Moreover, we show that each coordinator with an over-flow can determine which nodes are reserved to be the image of its over-flowing nodes. We prove that the resulting mapping, g, is close enough to f in the sense that the distance between $g(G_U)$ and $f(G_U)$ is small. We prove the latter by coupling g and f and showing that they agree on the mapping of most nodes as long as these nodes have enough neighbors (and if they do not, then their contribution to the distance between $g(G_U)$ and $f(G_U)$ has to be small).

1.6 A Lower Bound for the Property Testing Variant

We prove that for any D there exists a family of graphs with diameter $\Theta(D)$ such that any distributed two-sided error property testing algorithm for testing isomorphism on this family of graphs requires $\Omega(D)$ rounds. In the construction we start with a pair of graphs G_1 and G_2 that have diameter $O(D)$ which are far from being isomorphic. The graph G_U is then defined to be composed of G_1 and G_2 and a path of length $\Theta(D)$ that connects the two graphs. Roughly speaking, the idea is to argue that for round complexity which is at most D/c, where c is some absolute constant, the nodes in G_U which belong to the side of G_1 cannot distinguish the case in which the network is composed of two graphs which are isomorphic to G_1 (connected by a path). Likewise for the nodes that belong to the side of

G_2 (that cannot distinguish the case in which the network is composed of two graphs which are isomorphic to G_2). It then follows that the algorithm must err. In the detailed proof, which appears in the appendix, there are some technicalities that need to be addressed in order to prove that the above argument still holds when the nodes may use randomness, port numbers and IDs.

1.7 Related work

In this section we overview results in distributed decision and property testing in the CONGEST model. We also overview related results in centralized property testing.

Distributed Decision. There is a large body of algorithms and lower bounds for the *subgraph detection problem*: given a fixed graph H, and an input graph G, the task is to decide whether G contains a subgraph which is isomorphic to H. The subgraphs considered include: paths [36], cycles [36, 19, 12], triangles [31, 1, 8], cliques [11, 12, 6]. Abboud et al. [2, Sec. 6.2] considered the *identical subgraph detection problem.* In this problem the graph's nodes are partitioned into two equal sets. The task is to decide whether the induced graphs on these two sets are identical w.r.t. to a fixed mapping between the nodes of these two sets. They showed an $\Omega(n^2)$ lower bound on the number of rounds of any deterministic algorithm and a randomized algorithm that performs $O(D)$ rounds which succeeds w.h.p.

Distributed Property Testing for Graph Problems. Distributed property testing was initiated by Censor-Hillel et al. [7]. In particular, they designed and analyzed distributed property testing algorithms for: triangle-freeness, cycle-freeness, and bipartiteness. They also proved a logarithmic lower bound for the latter two properties. While they mainly focus on the bounded degree model and the general model they also studied the dense model[5]. In this model they showed that for a certain class of problems, any centralized property testing algorithm can be emulated in the distributed model such that number of rounds is q^2 where q denotes the number of queries made by the centralized tester. Fraigniaud et al. [23] studied distributed property testing of excluded subgraphs of size 4 and 5. Since the appearance of the above papers, there was a fruitful line of research in distributed property testing for various properties, mainly focusing on properties of whether a graph excludes a fixed sub-graph [22, 20, 21, 14, 13, 18]. Other problems on graphs such as testing planarity, and testing the conductance were studied in [39, 16], respectively.

Centralized Property Testing. Fischer and Matsliah [17] studied the graph isomorphism problem in the dense-graph model [26].[6] They considered four variations of the Graph Isomorphism testing problem: (1) one-sided error, where one of the graphs is known, and there is a query access to the graph which is tested, i.e., the tested graph is "unknown", (2) one-sided error, where there is a query access for both graphs, i.e., both graphs are unknown, (3) two-sided error, where one graph is known, (4) two sided error, where both graphs are unknown. For the first three variants Fischer and Matsliah [17] showed (almost)

[5] In centralized property testing, these models describe the distance measure as well as how the input graph is represented. In the distributive setting the difference between the models is only in the distance measure. The access to the input graph (which is the underlying network) is clearly the same in all models.

[6] In the dense-graph model, a graph G is considered to be ϵ-far from a property Π if the symmetric difference between its edge set to the edge set of any graph in Π is greater than $\epsilon|V(G)|^2$.

matching lower and upper bounds of, respectively: (1) $\tilde{O}(n)$, $\Omega(n)$, (2) $\tilde{O}(n^{3/2})$, $\Omega(n^{3/2})$, and (3) $\tilde{O}(n^{1/2})$, $\Omega(n^{1/2})$, where n is the number of vertices of each input (known or unknown) graph. For the fourth variant they showed an upper-bound of $\tilde{O}(n^{5/4})$ and a lower-bound of $\Omega(n)$. Onak and Sun [43] improved the upper bound of the fourth case to $O(n) \cdot 2^{\tilde{O}\left(\sqrt{\log n}\right)}$ by bypassing the distribution testing reduction that was used by [17]. Property testing of graph isomorphism was also considered in the bounded-degree model [27][7]. Goldreich [25] proved that the query complexity of any property testing algorithm is at least $\tilde{\Omega}(n^{1/2})$, for the variant in which one graph is known, and $\tilde{\Omega}(n^{2/3})$ when both graphs are unknown. Newman and Sohler [42] provide an algorithm for minor-free graphs with degree bounded by $d = O(1)$ (this class includes for example bounded degree planar graphs) whose query complexity is independent of the size of the graph. Moreover, they showed that any property is testable in this class of graphs with the same query complexity. Kusumoto and Yoshida [38], and Babu, Khoury, and Newman [3] considered testing of isomorphism between graphs which are forests and outerplanar in the general model [33][8], respectively. They both proved an upper bound of $\operatorname{poly}\log n$ and a lower bound of $\Omega(\sqrt{\log n})$ was shown in [38]. Moreover, they proved that any graph property is testable on these family of graphs with $\operatorname{poly}\log n$ queries.

2 The Algorithm for Testing Isomorphism in Dense Graphs

In this section, we describe and analyze the distributed algorithm for testing graph isomorphism in dense graphs. We begin with several useful definitions and observations, followed by the listing Algorithm 1 and the proof of its correctness (which follows from Lemma 11 and Lemma 12). Finally we discuss in more details how the algorithm is implemented in the CONGEST model.

We establish the following theorem.

▶ **Theorem 1.** *There exists a distributed two-sided error property testing algorithm for testing isomorphism (of dense graphs) that runs in $O(D + (\epsilon^{-1} \log n)^2)$ rounds in the CONGEST model. The algorithm succeeds with high probability.*

2.1 Definitions and Notation

We shall use the following definitions in our algorithm and in its analysis.

Let G be a graph and let $C = (c_1, \ldots, c_s)$ be a sequence of s nodes from $V(G)$.

▶ **Definition 2** ([17]). *For every node $v \in V(G)$, the C-label of v in G, denoted by $\ell_C^G(v)$, is a string of s bits defined as follows:*

$$\ell_C^G(v)_i = 1 \Leftrightarrow c_i \in N_G(v) \ ,$$

where $N_G(v)$ denotes the neighbors of v in G.

For the graphs we consider, $G = (V, E)$, we assume $V \subseteq [|V|^c]$ for some absolute constant $c \geq 1$. In particular we assume that there is a total order defined on V. We use the $\triangle$-operator to denote both the symmetric difference between two sets and when applied on

[7] In the bounded-graph model, a graph G with maximum degree d, is considered to be ϵ-far from a property Π if the symmetric difference between its edge set to the edge set of any graph in Π is greater than $\epsilon d|V(G)|$

[8] In the general-graph model, a graph G, is considered to be ϵ-far from a property Π if the symmetric difference between its edge set to the edge set of any graph in Π is greater than $\epsilon|E(G)|$.

graphs it denotes the Hamming distance between the corresponding adjacency matrices. We assume that the columns and rows of the adjacency matrices are ordered according to the order on the vertices.

▶ **Definition 3** ([17]). *For $\beta \in (0,1]$, we say that C is β-separating if for every pair of nodes u,v such that $|\triangle(N(u),N(v))| \geq \beta n$ it holds that u and v have different C-labels in G.*

▶ **Definition 4** (inverse of ℓ^G_C). *For a label $x \in \{0,1\}^s$, define $S^G_C(x) \triangleq \{v \in V(G) : \ell^G_C(v) = x\}$. Namely, $S^G_C(x)$ is the set of nodes in G for which the C-label is x.*

Let G and H be a pair of graphs such that $|V(G)| = |V(H)|$. The following definitions are defined with respect to a pair of sequences of $s \geq 1$ nodes from $V(G)$ and $V(H)$, $C_G = (c^G_1, \ldots, c^G_s)$ and $C_H = (c^H_1, \ldots, c^H_s)$, respectively.

We next define what we mean by saying that the mapping of a function $f : V(G) \to V(H)$ is consistent w.r.t. the labels of C_G and C_H.

▶ **Definition 5.** *For $f : V(G) \to V(H)$ which is a bijection, we say that f is* (C_G, C_H)-label-consistent *if the following holds:*

1. *f maps C_G to C_H: $f(c^G_i) = c^H_i$ for every $i \in [s]$.*
2. *The label of a node and its image is the same: $\ell^G_{C_G}(v) = \ell^H_{C_H}(f(v))$ for every $v \in V(G)$.*

For $f : V(G) \to V(H)$ and a sequence $C = (c_1, \ldots, c_s)$, we define $f(C)$ to denote $(f(c_1), \ldots, f(c_s))$ and $f(G)$ to denote the graph whose nodes are $V(H)$ and its edge set is $\{\{f(u), f(v)\} : \{u,v\} \in E(G)\}$.

We next observe that if G and H are isomorphic then for any sequence C and any function f which is an isomorphism between G and H, f is consistent w.r.t. C and $f(C)$.

▶ **Observation 1.** *If G and H are isomorphic and π is an isomorphism from $V(G)$ to $V(H)$ then for every sequence of nodes , C, from $V(G)$, π is $(C, \pi(C))$-label consistent. In particular, $|S^G_C(x)| = |S^H_{\pi(C)}(x)|$ for every $x \in \{0,1\}^s$.*

If f is not an isomorphism then it might be the case that it is not consistent w.r.t. C and $f(C)$. We next define a weaker notion of consistency which is being maximally-label-consistent.

▶ **Definition 6.** *We say that a function $f : V(G) \to V(H)$ is* maximally (C_G, C_H)-label-consistent *if the following holds:*

1. *f is a bijection.*
2. *f maps C_G to C_H: $f(c^G_i) = c^H_i$ for every $i \in [s]$.*
3. *For every $x \in \{0,1\}^s$ such that $|S^G_{C_G}(x)| = |S^H_{C_H}(x)|$, f maps the elements of $S^G_{C_G}(x)$ to the elements of $S^H_{C_H}(x)$.*

▶ **Definition 7.** *For a pair of graphs G and H, we say that a function, $f : V(G) \to V(H)$,* ϵ-approximates isomorphism *if f is a bijection and $f(G)$ is ϵ-close to H.*

See Figure 1 for illustration for the definitions in this section.

2.2 Distributed Algorithm Description

The listing of the distributed algorithm appears in Algorithm 1. The detailed description of the distributed implementation of Algorithm 1 appears in Section 2.4.

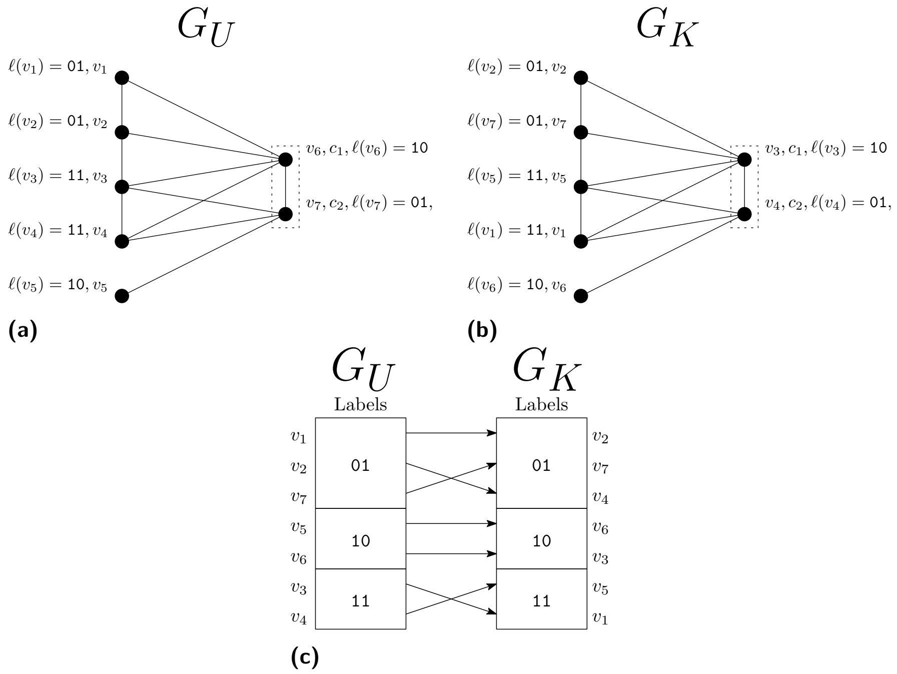

Figure 1 The unknown G_U and known G_K graphs are depicted in Sub-figure 1a, and 1b, respectively. The C sequence consists of the nodes $c_1 = v_6$ and $c_2 = v_7$. Hence, the label of a node v is a binary string $\ell_C^{G_U} = (b_2, b_1) \in \{0,1\}^2$, where $b_i = 1 \Leftrightarrow c_i \in N(v)$, and the same for G_K. In the figures, we removed the super (sub) scripts as they are clear from the context. The induced labels are depicted next to the node number. Note that all the nodes have a neighbor in C, hence the label class 00 is empty. In Sub-figure 1c two tables are depicted: (1) label assignment to the nodes of G_U, and (2) label assignment to the nodes of G_K. The order of which the nodes are presented is according to the (single) isomorphism function between the two graphs. One can observe that the number of nodes per each corresponding label-classes, e.g., a buckets, is the same, as well as buckets which have the same msb bit, e.g., a cluster of labels. A random bijection that preserves the bucket of labels is depicted by the arrows. We show in Thm. 13 that such a bijection yields graphs which are close to being isomorphic.

2.3 Correctness of the Distributed Testing Algorithm

In this subsection we prove the correctness of our algorithm. We begin with a couple of claims and lemmas that we use in our proof. Missing proofs are deferred to Appendix C.

The proof of the following claim appears in [17]. The proof of Lemma 9 can be derived from the proof of Lemma 4.11 in [17] (for the sake of completeness we provide both proofs in the appendix).

▷ Claim 8 ([17]). For $\beta \in (0,1]$ and a sequence, C, of $s = \Theta(\log(n/\delta)/\beta)$ nodes, chosen uniformly at random, C is β-separating with probability at least $1 - \delta$.

▶ **Lemma 9** ([17]). *Let G and H be isomorphic graphs and let π be an isomorphism between them. For any C that is an ϵ-separating sequence of nodes of G and for any f that is $(C, \pi(C))$-label-consistent it holds that $\Delta(f(G), H) \leq \epsilon n^2$.*

The following claim is implied directly from the multiplicative Chernoff's bound.

▷ Claim 10. Let G and H be two graphs such that $V(G) = V(H)$. Then by querying the adjacency-matrices of G and H in $\Theta(\log(1/\delta)/\epsilon)$ random entries it is possible to distinguish between the case that $\Delta(G,H) > \epsilon n^2$ from the case that $\Delta(G,H) \leq \epsilon n^2/2$ with probability at least $1 - \delta$.

Algorithm 1 Testing Isomorphism: The distributed network is $G_U = (V_U, E_U)$.

```
Input: A "known" graph, $G_K = (V_K, E_K)$ and a proximity parameter $\epsilon$ are given as
       an input to a single node $r$ (may be chosen adversarially).
Output: with high probability, all nodes output yes if $G_K$ is isomorphic to $G_U$ and
        no otherwise.

1 Compute a BFS tree, $T$, in $G_U$ rooted at $r$.
2 Pick, u.a.r., a sequence of $s \triangleq \Theta(\epsilon^{-1} \log |V_U|)$ nodes in $G_U$. Let $C \triangleq (c_1, \ldots, c_s)$
   denote this sequence.
3 Each node $v \in V(G_U)$ computes its label, $\ell_C^{G_U}(v) \in \{0,1\}^s$, according to $C$ and its
   neighbors in $G_U$ (see Definition 2), and sends this label to its neighbors.
4 The node $r$ picks a sequence of $t \triangleq \Theta(\epsilon^{-1} \log(|V_K|^s))$ pairs of nodes,
   $A = ((i_1, j_1), \ldots, (i_t, j_t))$, u.a.r. from $V_U \times V_U$. Let $I \triangleq \{i_1, j_1, \ldots, i_t, j_t\}$.
5 For every $e \in A$, $r$ sends $e$ down the BFS tree and learns whether $e \in E_U$ or not.
   Similarly, for every $v \in I$, it learns $\ell_C^{G_U}(v)$, i.e., the $C$-label of $v$ in $G_U$.
6 For each sequence, $P = (p_1, \ldots, p_s)$, of $s$ nodes from $V_K$, $r$ proceeds as follows:
    1. For every $i \in [s]$, verify that $\ell_C^{G_U}(c_i) = \ell_P^{G_K}(p_i)$. If not, then reject $P$ as a
       candidate and proceed to the next sequence.
    2. For every $v \in I$, let $\ell$ denote $\ell_C^{G_U}(v)$. Check if $|S_C^{G_U}(\ell)| = |S_P^{G_K}(\ell)|$.
       If not, then reject $P$ as a candidate and proceed to the next sequence.
    3. Pick uniformly at random a function $f$ from the set of all functions that are
       maximally $(C, P)$-label-consistent (see Definition 6).
    4. Compute the number of edges in $A$ which are non-edges in $f(A)$ and vice-versa.
       That is, the number of edges $(i_k, j_k) \in A$ such that $(i_k, j_k) \in E_U$ and
       $(f(i_k), f(j_k)) \notin E_K$, or $(i_k, j_k) \notin E_U$ and $(f(i_k), f(j_k)) \in E_K$).
       If it is at most $(3\epsilon)|A|/2$ then return yes.
If all sequences, $P$, failed to pass the previous step then return no.
```

▶ **Lemma 11.** *If G_U is isomorphic to G_K then Algorithm 1 accepts with high probability.*

Proof. Assume G_U is isomorphic to G_K and let $\pi : V(G_U) \to V(G_K)$ denote an isomorphism from G_U to G_K. Since the algorithm goes over every sequence P of s nodes from V_K, it also checks $\pi(C)$. By Observation 1, the probability that $\pi(C)$ passes Step 2 is 1. By Claim 8, with high probability, C is $(\epsilon/2)$-separating (see Definition 3). By Lemma 9, if C is $(\epsilon/2)$-separating, then $f(G_U)$ is $(\epsilon/2)$-close to $\pi(G_U) = G_K$. If $f(G_U)$ is $(\epsilon/2)$-close to G_K, then by Claim 10, with high probability $\pi(C)$ passes Step 4. Therefore by the union bound, the algorithm accepts with high probability. ◀

▶ **Lemma 12.** *If G_U is ϵ-far from being isomorphic to G_K then Algorithm 1 rejects with high probability.*

Proof. Assume G_U is ϵ-far from being isomorphic to G_K. We claim that with high probability, any sequence P, fails to pass Step 6. We show this by bounding the probability that a fixed P passes Step 6 and then apply the union bound over all possible sequences. Fix a sequence P and assume that P passes Step 2 (otherwise we are done). Let f be the corresponding f from Step 3 (that is chosen at random). Since G_U is ϵ-far from being isomorphic to G_K, by definition, $\Delta(f(G_U), G_K) \geq \epsilon n^2$. Recall that f is chosen uniformly at random from the set of functions that are maximally (C, P)-label-consistent. By Construction the selection of f is independent of the selection of A. Therefore we can apply Claim 10 on Step 4 as A is a set of potential edges chosen uniformly at random and, in particular, independently from the random coins in which we use to select f. By Claim 10, P succeeds to pass Step 4 with probability at most $1/|V_K|^{s+c}$ for any absolute constant c. Thus, the lemma follows by a union bound over all possible sequences, as their number is bounded by $|V_K|^s$. ◀

2.4 A Detailed Description of the Distributed Implementation of Algorithm 1

In this section we provide a detailed description of the distributed implementation of our algorithm. We focus on steps for which the implementation is not straightforward and analyze their round complexity. In particular, we focus on Steps 2,6.2,6.3-6.4.

Step 2: Selecting s Nodes u.a.r. In Appendix C.4 we describe a simple procedure to select s nodes uniformly at random (which is a kind of folklore). The round complexity of this procedure is $D + s$.

Step 6.2: Computing the Size of $S_C^{G_U}(\ell)$. Clearly, r can compute $S_P^{G_K}(\ell)$ for any label ℓ as r knows P and G_K. Therefore, in order to describe the implementation of Step 2 it suffices to explain how r can obtain $|S_C^{G_U}(\ell)|$.

We begin with the special case of $\ell = (0, \dots, 0)$. The nodes in G_U that have this label are nodes that are not adjacent to any one of the nodes in C. Their number can be computed in $O(D)$ rounds by summing it up the BFS tree as follows. Assume w.l.o.g. that every node knows its layer in the BFS-tree. In the first round, every node that is in the last layer (which is also a leaf) sends 1 up to its parent if its C-label is $(0, \dots, 0)$. In the next round, all nodes in the next layer sum up the received numbers and add 1 if their C-label is $(0, \dots, 0)$. They send this number up to their parents and so on until we get to the root.

Consider a label ℓ for which at least one bit is 1. Let $\mathrm{msb}(\ell)$ denote the maximum i such that $\ell_i = 1$. Since the node $c_{\mathrm{msb}(\ell)}$ is connected to all nodes whose C-label is ℓ, it can compute their total number (recall that in Step 3 every node sends its C-label to all its neighbors) and send it to the root. Therefore, by a pipelining argument, the root can obtain $|S_C^{G_U}(\ell)|$ for every ℓ which is a C-label of a node in I in $O(D + |I|) = O(D + (\epsilon^{-1} \log n)^2)$ rounds.

Steps 6.3-6.4: Accessing f. Recall that we require from f to be chosen u.a.r. from the set of all functions that are maximally (C, P)-label-consistent (see Definition 6). Recall that f is only evaluated on nodes in I but at the same time its selection has to be independent of A (and I). In particular, the selection of f should be independent of the queries we make to f. To this end, the root verifies the following:

1. In Sub-step 1 of Step 6 it verifies that for the selected sequences, C and P, corresponding nodes have matching labels. Namely, $\ell_C^{G_U}(c_i) = \ell_P^{G_K}(p_i)$ for every $i \in [s]$.
2. In Sub-step 2 of Step 6 it verifies that $|S_C^{G_U}(\ell)| = |S_P^{G_K}(\ell)|$ for every ℓ which is a C-label of a vertex in I.

If both conditions hold, then it follows that by mapping the nodes in $S_C^{G_U}(\ell)$ to the nodes in $S_P^{G_K}(\ell)$ u.a.r. and independently from the mapping of all other nodes (except for the mapping of C to P which is already determined) for every ℓ such that $|S_C^{G_U}(\ell)| = |S_P^{G_K}(\ell)|$, we are in fact accessing f which is drawn according to the desired distribution. Therefore the root r simply maps every $v \in I$ to a uniform node $u \in V_K$ such that: (1) $\ell_C^{G_U}(v) = \ell_P^{G_K}(u)$ (2) u is still unmapped (such node always exists). Since r knows G_K and $\ell_C^{G_U}(v)$ for every $v \in I$, it is able to compute $f(v)$ for every $v \in I$, as desired.

Notice that if we evaluated f in the same manner as described above for nodes, v, whose labels, ℓ, are such that $|S_C^{G_U}(\ell)| \neq |S_P^{G_K}(\ell)|$ then the selection of f would not be independent of I. For instance, if $|S_C^{G_U}(\ell)| > |S_P^{G_K}(\ell)|$ then the first queries of nodes in $S_C^{G_U}(\ell)$ will be mapped to $S_P^{G_K}(\ell)$ but then we will run out of nodes in $S_P^{G_K}(\ell)$, implying that the rest of the nodes in $S_C^{G_U}(\ell)$ will be mapped to nodes with a different label than ℓ. Hence, in this case the selection of f will not be oblivious to the order of queries we make to it (and in particular is not independent of I). In the next section we describe a bijection g which can be evaluated on all the nodes in G_U (with the same round complexity as the testing algorithm) and at the same time is close enough to f.

3 Computing an Approximate Isomorphism

In this section we prove the following theorem.

▶ **Theorem 13.** *Let G_U denote the input graph and let G_K be a graph which is isomorphic to G_U and is given as an input to all nodes in the network. There exists a randomized algorithm such that each node in G_U, v, outputs $g(v)$ where $g : V_U \to V_K$ is a bijection such that $g(G_U)$ is ϵ-close to G_K (i.e. g ϵ-approximates isomorphism). The round complexity of the algorithm is $O(D + (\epsilon^{-1} \log n)^2)$. The algorithm succeeds with high probability.*

Proof. The first step of the algorithm is to run Algorithm 1 with the only difference that in Step 6 the root also verifies that $|S_C^{G_U}(\ell)| = |S_P^{G_K}(\ell)|$ for $\ell = (0, \ldots, 0)$. Since G_K and G_U are isomorphic, by Lemma 11, w.h.p. the algorithm accepts and hence finds P and the corresponding f that pass Step 6. Recall that since f passes the test in Sub-step 6.4 then w.h.p. $f(G_U)$ is ϵ-close to G_K. If every node v could output $f(v)$ then we were done. However, we can not compute f for every node v because for a constant fraction of the nodes its computation might require global information on G_U. Instead, our goal is to output g which is $O(\epsilon)$-close to f and can be computed for every node without causing too much congestion. We next describe g and its computation.

We begin with some notation. Let $L_i \subseteq \{0,1\}^s$ denote the set of labels ℓ for which $\mathrm{msb}(\ell) = i$. Namely, L_i contains all the labels in $\{0,1\}^s$ that can be generated by the following regular expression $0^{i-1}1(0 \cup 1)^{s-i}$. Let Y denote the set of nodes in V_U whose C-label is $(0, \ldots, 0)$. Similarly, let Y' denote the set of nodes in V_K whose P-label is $(0, \ldots, 0)$. For a graph H and a sequence D let $J_D^H(i) \triangleq \cup_{\ell \in L_i} S_D^H(\ell)$, namely, this is the set of all nodes in H whose D-label belongs to L_i. We may refer to $J_D^H(i)$ as the i-th cluster of the graph H w.r.t. D. For $i \in [s]$, define $j_i \triangleq |J_C^{G_U}(i)| - |J_P^{G_K}(i)|$. Namely, j_i is the difference between the sizes of the i-th clusters in both graphs (w.r.t. C and P, respectively).

We next define the set of *reserved* nodes of V_K, denote by R. For each i such that $j_i < 0$, $|j_i|$ nodes from $J_P^{G_K}(i)$ belong to R. Specifically, these are the nodes whose order [9] is the least from the vertices in $J_P^{G_K}(i)$. We consider the order to be the same order as in V_K only that elements in P have the highest order (this is to ensure that none of the elements in P belong to R).

The function g is selected randomly. We next describe the selection of g the outcome of a global random process. Thereafter we describe how this random process can be implemented distributively.

[9] Recall that we assume that there is a total order on V_K which is known to all the nodes in G_U.

Description of g . We describe the selection of g by describing how the nodes in $J_C^{G_U}(i)$ are mapped to V_K, for every $i \in [s]$. We consider three cases, according to the value of j_i.

The first case is when $j_i = 0$. We have the following sub-cases.

1. For every $\ell' \in L_i$ such that $|S_C^{G_U}(\ell')| = |S_P^{G_K}(\ell')|$, g matches u.a.r. the elements in $S_C^{G_U}(\ell')$ to the elements in $S_P^{G_K}(\ell')$.
2. The rest of the elements in $J_C^{G_U}(i)$ are matched u.a.r. to the unmatched elements in $J_P^{G_K}(i)$.

Therefore, in this case the elements in $J_C^{G_U}(i)$ are matched only to the elements in $J_P^{G_K}(i)$ and vice versa.

The second case is when $j_i < 0$. We have the following sub-cases.

1. For every $\ell' \in L_i$ such that $|S_C^{G_U}(\ell')| = |S_P^{G_K}(\ell')|$ and $S_P^{G_K}(\ell') \cap R = \emptyset$, g matches the elements in $S_C^{G_U}(\ell')$ u.a.r. to the elements in $S_P^{G_K}(\ell')$.
2. For every $\ell' \in L_i$ such that $|S_C^{G_U}(\ell')| = |S_P^{G_K}(\ell')|$ and $S_P^{G_K}(\ell') \cap R \neq \emptyset$, g matches u.a.r. a random set of $|S_P^{G_K}(\ell') \setminus R|$ elements from $S_C^{G_U}(\ell')$ to $S_P^{G_K}(\ell') \setminus R$.
3. The rest of the un-matched elements in $J_C^{G_U}(i)$ are mapped u.a.r. to the un-matched elements in $J_P^{G_K}(i) \setminus R$.

Observe that all the elements in $J_C^{G_U}(i)$ are matched to elements in $J_P^{G_K}(i)$ and that the elements that belong to $J_P^{G_K}(i) \cap R$ are still un-matched.

The third case is when $j_i > 0$. We have the following sub-cases.

1. For every $\ell' \in L_i$ such that $|S_C^{G_U}(\ell')| = |S_P^{G_K}(\ell')|$, g matches the elements in $S_C^{G_U}(\ell')$ u.a.r. to the elements in $S_P^{G_K}(\ell')$.
2. The rest of the elements in $J_C^{G_U}(i)$ are matched u.a.r. to the unmatched elements in $J_P^{G_K}(i)$.
3. The remaining $|j_i|$ elements in $J_C^{G_U}(i)$ are matched to the nodes of order $(\sum_{a<i} j_a) + 1$ to $(\sum_{a<i} j_a) + j_i$ in R.

This concludes the description of g for nodes that do not belong to Y. It remains to describe g for nodes that belong to Y. We defer this description to Appendix C.5 as well as the description on how it is computed distributively.

We next describe how g can be computed distributively for nodes that have at least one neighbor in C (namely, nodes that do not belong to Y). Each node $c_i \in C$ is responsible to compute and to send to each node v whose C-label is in L_i the value $g(v)$ (note that c_i and v are necessarily neighbors). As a preliminary step, each node c_i computes $j_i = |J_C^{G_U}(i)| - |J_P^{G_K}(i)|$ and sends (i, j_i) up the BFS tree. Notice that $\sum_{i \in s} j_i = 0$ as $|S_C^{G_U}(\ell)| = |S_P^{G_K}(\ell)|$ for $\ell = (0, \ldots, 0)$ and $|V_U| = |V_K|$. The root sends the set $\{(i, j_i)\}_{i \in s}$ down the BFS tree. By knowing G_K and the set $\{(i, j_i)\}_{i \in s}$, every node c_i can easily compute R. It is not hard to see that this suffices in order to match the elements in $J_C^{G_U}(i)$ to V_K as described above.

By construction it follows that g is a bijection. The bound on the round complexity follows from the bound on the round complexity of Algorithm 1 and the fact that there are only $s = O(\epsilon^{-1} \log n)$ clusters. It remains to prove the following claim.

▷ Claim 14. With high probability $\Delta(g(G_U), G_K) < \epsilon n^2$.

Proof. We observe that both f and g are functions that are chosen randomly. To prove the claim about g we couple g to f and show that they agree on the mapping of most vertices from the set of vertices that have $\Omega(\epsilon n)$ neighbors. This will imply that $\Delta(f(G_U), g(G_U)) = O(\epsilon n^2)$. By Claim 10, since f passed the test in Step 4, w.h.p. $\Delta(f(G_U), G_K) < \epsilon n^2$. We denote this event by E_1. From the fact that w.h.p. E_1 occurs, with combination with the coupling it will follow that w.h.p. $\Delta(g(G_U), G_K) = O(\epsilon n^2)$. Therefore, by setting the proximity parameter to be $\Theta(\epsilon)$ the claim will follow.

Consider the following description of g in terms of f. Let $B = \{\ell \in \{0,1\}^s : |S_C^{G_U}(\ell)| \neq |S_P^{G_K}(\ell)|\}$. For every $v \in V_U$, $g(v) = f(v)$ unless $v \in Y \cup B$ or, $v \notin Y \cup B$ and $f(v) \in R$. In these cases we match v to a node in V_K as listed in the description of g. Observe that under this formulation the distribution of g remains the same. The only difference is that now, for the sake of the analysis, it is coupled to the distribution of f.

It follows that the number of nodes $v \in V_U$ for which $f(v) \neq g(v)$ is at most $|Y|+|R|+|B|$. By Step 2 of Algorithm 1, with high probability it holds that $\sum_{\ell \in B} |S_C^{G_U}(\ell)| \leq \epsilon n$. On the other hand, w.h.p., the number of neighbors of every node in Y is at most ϵn. This implies that the number of nodes in the neighborhood of Y' is at most $O(\epsilon n^2)$ (since otherwise would reject f w.h.p.). Therefore, w.h.p., the contribution of the nodes in Y and Y' to $\Delta(f(G_U), g(G_U))$ is at most $O(\epsilon n^2)$. Thus, w.h.p. $\Delta(f(G_U), g(G_U)) = O(\epsilon n^2)$, as desired. We denote this event by E_2.

Hence, we obtain by union bound over the probability that either $\overline{E_1}$ or $\overline{E_2}$ occur, that w.h.p. $\Delta(g(G_U), G_K) = O(\epsilon n^2)$. By setting the proximity parameter of the algorithm to be ϵ/c for some absolute constant $c > 1$ (which does not affect the asymptotic complexity of the algorithm), we obtain the desired result. ◁

This concludes the proof of the Theorem. ◀

4 Distributed Algorithm for Deciding Isomorphism

In this section we prove the following theorem.

▶ **Theorem 15.** *There exists a randomized distributed algorithm in the CONGEST model that decides if G_K and G_U are isomorphic with high probability. The round complexity of the algorithm is $O(n)$. The algorithm works even if G_K is given only to a single (arbitrary) node.*

Proof. The idea of the algorithm is to go over all possible one-to-one mappings between the nodes of G_K and the nodes of G_U and to test for equality of the corresponding adjacency matrices. The test for equality is performed with very high confidence level in order to ensure that the total error probability is bounded by a small constant. We note that a similar reduction for testing equality also appears in the algorithm of [2, Sec. 6.2] for the Identical Subgraph Detection problem (ISDP).

The first step of our algorithm is to construct a BFS tree and to assign to each node in the network a unique label in $[n]$ where $n \triangleq |V(G_K)|$. This step requires $\Omega(D)$ rounds where D denotes the diameter of G_U (see details on the implementation of this step in the proof of Theorem 13). Consider the adjacency matrix of G_U, M, in which the rows are sorted according to the labels assigned to the nodes. We consider the natural total order on the following set of pairs of nodes $P = \{(i,j) : i \in [n], j \in [n], i < j\}$ in which the pairs are sorted according to the first element and ties are broken according to the second element. Let $\ell(i,j)$ denote the order of the pair (i,j). Each pair $(i,j) \in P$ corresponds to the potential edge between the pair of nodes with labels i and j, respectively. The matrix M can be represented as an integer $s(M)$ where for each $(i,j) \in P$ the $\ell(i,j)$-th lsb (least significant bit) of the binary representation of $s(M)$ indicates whether (i,j) is an edge in the graph. Observe that we can calculate $s(M)$ in D rounds by starting the calculation at the lowest layer of the BFS tree and summing up the outcomes as we go up the tree, layer by layer. The calculation is performed such that each power of two, 2^ℓ, is added (once) if and only if the corresponding edge is present in the graph (i.e. $2^{\ell(i,j)}$ is added if and only if (i,j) is present in the graph).

Since the representation of $s(M)$ requires $O(n^2)$ bits, for our goal we calculate $s(M)$ mod p instead, where p is a prime number in $[n^2]$ chosen uniformly at random. To this end we proceed in the same manner as mentioned above only that before the nodes send up the tree the outcome of the intermediate sums, they apply the $\mod p$ operation on the outcome. Let $\mathcal{P} = \{p_1, \ldots, p_k\}$ be a multiset of k prime numbers, where $k = \Theta(n)$, each chosen, by a single node v, independently and uniformly at random from the first n^2 primes. [10] The node v broadcasts $\mathcal{P}$ to all the nodes in network in $O(n)$ rounds. Let M' denote the adjacency matrix of G_K. If $M' \neq M$ then the probability that $s(M) \equiv s(M') \mod p$ for a random prime number in $[n^2]$ is at most $1/\Omega(n)$ [37]. Therefore the probability that $s(M) \equiv s(M') \mod p$ every $p \in \mathcal{P}$ is at most $1/\Omega(n^k) = 1/\Omega(n^n)$. Thus we can test with one-sided error if G_U and G_K are equal. The soundness of the equality test is $1 - 1/\Omega(n^n)$. We can apply the equality test for every mapping π between G_K and G_U and return yes if and only if there exists a mapping for which the test accepts. Namely, we go over all possible permutations over the nodes of G_K and for each permutation, π, we perform the equality test between $s(M)$ and $s(\pi(M'))$ where $\pi(M')$ denotes adjacency matrix of G_K after applying the permutation π on the nodes. By the soundness of the equality test and the union bound, the probability that this test returns no when G_K and G_U are not isomorphic is at least $2/3$ (for an appropriate adjustment of the parameters).

By standard pipelining, it is possible to calculate $s(M) \mod p$ for every $p \in \mathcal{P}$ in $O(D+k)$ rounds, therefore the round complexity of the above test is $O(n)$. To verify this observe that in order to execute the above test the only information we need is of G_K and the result of $s(M) \mod p$ for every $p \in \mathcal{P}$. This concludes the proof. ◀

We observe that the above-mentioned algorithm can be adapted to the semi-streaming in a straight-forward way as follows.

▶ **Theorem 16.** *There exists an algorithm in the semi-streaming model that receives a graph G_K over n nodes as an input, where the space for storing G_K is a read-only memory, and a stream of the edges of another graph G_U (according to any order) and decides, with one-sided error, whether G_K and G_U are isomorphic or not. The algorithm performs one-pass and uses $O(n \log n)$ bits of space.*

Proof. Assume w.l.o.g. that the labels of the nodes in G_U and G_K are taken from $[n]$. Otherwise we can re-name then by using a table of size $O(n \log n)$ bits. Let M denote the adjacency matrix of G_U. We first compute $s(M) \mod p$ for every $p \in \mathcal{P}$ as in the proof of Theorem 15, in one-pass, using $O(n \log n)$ bits of space. We then go over all permutations of the nodes of G_K and perform the same computation for the corresponding adjacency matrix. We accept if and only if there exists a permutation π for which $s(M) \equiv s(M') \mod p$ every $p \in \mathcal{P}$, where M' denotes the adjacency matrix of G_K after we permuted the nodes according to π. Observe that we can go over the permutations one by one according to the lexicographical order by using $O(n \log n)$ bits of space. Therefore, the total space the algorithm uses is $O(n \log n)$, as desired. ◀

References

1 Amir Abboud, Keren Censor-Hillel, Seri Khoury, and Christoph Lenzen. Fooling views: A new lower bound technique for distributed computations under congestion. *arXiv preprint*, 2017. arXiv:1711.01623.

2 Amir Abboud, Keren Censor-Hillel, Seri Khoury, and Ami Paz. Smaller cuts, higher lower bounds. *CoRR*, abs/1901.01630, 2019. arXiv:1901.01630.

[10] It is well known that for sufficiency large number $x \in \mathbb{N}$ the number of prime numbers that are at most x is $\Theta(x/\log x)$.

3 Jasine Babu, Areej Khoury, and Ilan Newman. Every property of outerplanar graphs is testable. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2016, September 7-9, 2016, Paris, France*, volume 60 of *LIPIcs*, pages 21:1–21:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.APPROX-RANDOM.2016.21.

4 Tugkan Batu, Eldar Fischer, Lance Fortnow, Ravi Kumar, Ronitt Rubinfeld, and Patrick White. Testing random variables for independence and identity. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pages 442–451. IEEE, 2001.

5 Tuğkan Batu, Lance Fortnow, Ronitt Rubinfeld, Warren D Smith, and Patrick White. Testing closeness of discrete distributions. *Journal of the ACM (JACM)*, 60(1):1–25, 2013.

6 Matthias Bonne and Keren Censor-Hillel. Distributed detection of cliques in dynamic networks. In *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece*, pages 132:1–132:15, 2019. doi:10.4230/LIPIcs.ICALP.2019.132.

7 Keren Censor-Hillel, Eldar Fischer, Gregory Schwartzman, and Yadu Vasudev. Fast distributed algorithms for testing graph properties. *Distributed Computing*, 32(1):41–57, 2019. doi:10.1007/s00446-018-0324-8.

8 Yi-Jun Chang, Seth Pettie, and Hengjie Zhang. Distributed triangle detection via expander decomposition. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 821–840. SIAM, 2019.

9 Lin Chen. Parallel graph isomorphism detection with identification matrices. In *Proceedings of the International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN)*, pages 105–112. IEEE, 1994.

10 Lin Chen. Graph isomorphism and identification matrices: Parallel algorithms. *IEEE Transactions on Parallel and Distributed Systems*, 7(3):308–319, 1996.

11 Artur Czumaj and Christian Konrad. Detecting cliques in congest networks. *Distributed Computing*, pages 1–11, 2019.

12 Talya Eden, Nimrod Fiat, Orr Fischer, Fabian Kuhn, and Rotem Oshman. Sublinear-time distributed algorithms for detecting small cliques and even cycles. In *33rd International Symposium on Distributed Computing (DISC 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

13 Guy Even, Orr Fischer, Pierre Fraigniaud, Tzlil Gonen, Reut Levi, Moti Medina, Pedro Montealegre, Dennis Olivetti, Rotem Oshman, Ivan Rapaport, and Ioan Todinca. Three notes on distributed property testing. In *Proceedings of the 41st International Symposium on Distributed Computing (DISC)*, pages 15:1–15:30, 2017. doi:10.4230/LIPIcs.DISC.2017.15.

14 Guy Even, Reut Levi, and Moti Medina. Faster and simpler distributed algorithms for testing and correcting graph properties in the congest-model. *CoRR*, abs/1705.04898, 2017. arXiv:1705.04898.

15 Joan Feigenbauma, Sampath Kannanb, Andrew McGregorb, Siddharth Surib, and Jian Zhanga. On graph problems in a semi-streaming model. *Theoretical Computer Science*, 348:207–216, 2005.

16 Hendrik Fichtenberger and Yadu Vasudev. Distributed testing of conductance. *arXiv preprint*, 2017. arXiv:1705.08174.

17 Eldar Fischer and Arie Matsliah. Testing graph isomorphism. *SIAM Journal on Computing*, 38(1):207–225, 2008.

18 Orr Fischer, Shay Gershtein, and Rotem Oshman. On the multiparty communication complexity of testing triangle-freeness. In *Proceedings of the 2017 ACM Symposium on Principles of Distributed Computing (PODC)*, pages 111–120. ACM, 2017.

19 Orr Fischer, Tzlil Gonen, Fabian Kuhn, and Rotem Oshman. Possibilities and impossibilities for distributed subgraph detection. In *Proceedings of the 30th on Symposium on Parallelism in Algorithms and Architectures*, pages 153–162, 2018.

20 Orr Fischer, Tzlil Gonen, and Rotem Oshman. Distributed property testing for subgraph-freeness revisited. *CoRR*, abs/1705.04033, 2017. `arXiv:1705.04033`.

21 Pierre Fraigniaud, Pedro Montealegre, Dennis Olivetti, Ivan Rapaport, and Ioan Todinca. Distributed subgraph detection. *CoRR*, abs/1706.03996, 2017. `arXiv:1706.03996`.

22 Pierre Fraigniaud and Dennis Olivetti. Distributed detection of cycles. *ACM Transactions on Parallel Computing (TOPC)*, 6(3):1–20, 2019.

23 Pierre Fraigniaud, Ivan Rapaport, Ville Salo, and Ioan Todinca. Distributed testing of excluded subgraphs. In *Proceedings of the 30th International Symposium on Distributed Computing (DISC)*, volume 9888 of *LNCS*, pages 342–356. Springer, 2016.

24 Hillel Gazit and J Reif. A randomized parallel algorithm for planar graph isomorphism. In *Proceedings of the second annual ACM symposium on Parallel algorithms and architectures*, pages 210–219, 1990.

25 Oded Goldreich. Testing isomorphism in the bounded-degree graph model. *Electronic Colloquium on Computational Complexity (ECCC)*, 26:102, 2019. URL: `https://eccc.weizmann.ac.il/report/2019/102`.

26 Oded Goldreich, Shafi Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. *Journal of the ACM (JACM)*, 45(4):653–750, 1998.

27 Oded Goldreich and Dana Ron. Property testing in bounded degree graphs. *Algorithmica*, 32(2):302–343, 2002. `doi:10.1007/s00453-001-0078-7`.

28 Oded Goldreich and Dana Ron. On testing expansion in bounded-degree graphs. In *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation*, pages 68–75. Springer, 2011.

29 Martin Grohe and Oleg Verbitsky. Testing graph isomorphism in parallel by playing a game. In *International Colloquium on Automata, Languages, and Programming*, pages 3–14. Springer, 2006.

30 John E Hopcroft and Jin-Kue Wong. Linear time algorithm for isomorphism of planar graphs (preliminary report). In *Proceedings of the sixth annual ACM symposium on Theory of computing*, pages 172–184. ACM, 1974.

31 Taisuke Izumi and François Le Gall. Triangle finding and listing in congest networks. In *Proceedings of the ACM Symposium on Principles of Distributed Computing*, pages 381–389, 2017.

32 Joseph Jaja and S Rao Kosaraju. Parallel algorithms for planar graph isomorphism and related problems. *IEEE Transactions on Circuits and Systems*, 35(3):304–311, 1988.

33 Tali Kaufman, Michael Krivelevich, and Dana Ron. Tight bounds for testing bipartiteness in general graphs. *SIAM Journal on Computing*, 33(6):1441–1483, 2004. `doi:10.1137/S0097539703436424`.

34 Paul J Kelly et al. A congruence theorem for trees. *Pacific Journal of Mathematics*, 7(1):961–968, 1957.

35 Johannes Köbler. On graph isomorphism for restricted graph classes. In *Conference on Computability in Europe*, pages 241–256. Springer, 2006.

36 Janne H Korhonen and Joel Rybicki. Deterministic subgraph detection in broadcast congest. In *21st International Conference on Principles of Distributed Systems (OPODIS 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

37 Eyal Kushilevitz and Noam Nisan. *Communication complexity.* Cambridge University Press, 1997.

38 Mitsuru Kusumoto and Yuichi Yoshida. Testing forest-isomorphism in the adjacency list model. In *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I*, volume 8572 of *Lecture Notes in Computer Science*, pages 763–774. Springer, 2014. `doi:10.1007/978-3-662-43948-7_63`.

39 Reut Levi, Moti Medina, and Dana Ron. Property testing of planarity in the CONGEST model. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing, PODC 2018, Egham, United Kingdom, July 23-27, 2018*, pages 347–356, 2018. Full version is available in http://arxiv.org/abs/1805.10657. URL: `https://dl.acm.org/citation.cfm?id=3212748`.

40 Nathan Linial. Locality in distributed graph algorithms. *SIAM Journal on Computing*, 21(1):193–201, 1992.

41 Eugene M Luks. Parallel algorithms for permutation groups and graph isomorphism. In *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*, pages 292–302. IEEE, 1986.

42 Ilan Newman and Christian Sohler. Every property of hyperfinite graphs is testable. *SIAM Journal on Computing*, 42(3):1095–1112, 2013. doi:10.1137/120890946.

43 Krzysztof Onak and Xiaorui Sun. The query complexity of graph isomorphism: bypassing distribution testing lower bounds. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 165–171, 2018. doi:10.1145/3188745.3188952.

44 David Peleg. Distributed computing. *SIAM Monographs on discrete mathematics and applications*, 5, 2000.

45 Gregory Valiant and Paul Valiant. The power of linear estimators. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, pages 403–412. IEEE, 2011.

46 Oleg Verbitsky. Planar graphs: Logical complexity and parallel isomorphism tests. In *Annual Symposium on Theoretical Aspects of Computer Science*, pages 682–693. Springer, 2007.

47 Andrew Chi-Chin Yao. Probabilistic computations: Toward a unified measure of complexity. In *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*, pages 222–227. IEEE, 1977.

A Lower Bounds

In this section we establish two lower bounds. The first is for the decision variant of GI, and the second is for the testing variant.

For the decision variant, we prove a near-quadratic lower bound for any deterministic distributed algorithm in the CONGEST model. The second lower bound states that any Isomorphism testing distributed algorithm requires diameter time. This lower bound holds also for randomized algorithms, and in fact holds in the LOCAL model, even if *all* vertices are given as an input the graph G_K.[11]

A.1 An $\Omega(n^2/\log n)$ Lower Bound for Deciding Isomorphism Deterministically

The decision variant of our Isomorphism testing problem is as follows: if G_K and G_U are isomorphic, then all nodes should output yes, while if the graphs are not isomorphic, at least one node should output no.

▶ **Theorem 17.** *Any distributed deterministic algorithm in the CONGEST model for deciding whether G_U is isomorphic to G_K requires $\Omega(n^2/\log n)$ rounds for the setting in which G_K is given as an input to a single node. Moreover, this bound holds even for graphs with constant diameter.*

Proof. We reduce the problem of Set-Equality to the problem of deciding Isomorphism. The reduction is as follows. Alice and Bob each receives as an input a subset of k^2 elements, $A \subseteq \{(x, y) \mid x \in \{a_1^0, \ldots, a_1^{k-1}\}, y \in \{a_2^0, \ldots, a_2^{k-1}\}\}$ and $B \subseteq \{(x, y) \mid x \in \{b_1^0, \ldots, b_1^{k-1}\}, y \in \{b_2^0, \ldots, b_2^{k-1}\}\}$, respectively. According to their inputs they construct a pair of graphs, G_U and G_K, such that $A = B$ if and only if G_U is isomorphic to G_K, as described momentarily.

[11] In the LOCAL model there is no limitation on message size per round per edge. Obviously, a lower bound in the LOCAL model also applies to the CONGEST model.

Since, Set-Equality has a deterministic communication complexity which is linear in size of the universe [37], which is $\Omega(k^2)$ in this case, it follows that any distributed deterministic algorithm in the CONGEST model for deciding whether G_U is isomorphic to G_K requires $\Omega(n^2/\log n)$ rounds.

We now describe the graph G_U which is constructed by Alice and Bob. The set of nodes in Alice's graph is composed of u, u', u'', the subsets $A_1 = \{a_1^0, \dots, a_1^{k-1}\}$ and $A_2 = \{a_2^0, \dots, a_2^{k-1}\}$, and the nodes $t_{A_1}, t_{A_2}, t_u^1, t_u^2$ (see Figure 2). The subgraph induced on the nodes in A_1 is the path $a_1^0, \dots, a_1^{k-1}$. Similarly, the subgraph induced on the nodes in A_2 is the path $a_2^0, \dots, a_2^{k-1}$. The nodes u and u'' are adjacent to all nodes in A_1 and A_2, respectively. Both nodes u and u'' are adjacent to the node u'. The node u is adjacent to t_u^1 and t_u^2. The nodes a_1^0 and a_2^0 are adjacent to the nodes t_{A_1} and t_{A_2}, respectively. Additionally, the edge (a_1^x, a_2^y) belongs to Alice's subgraph if and only if the corresponding (x, y) element is in A.

The subgraph of Bob is defined similarly only that the nodes $v, v', v'', t_{B_1}, t_{B_2}$ take the role of $u, u', u'', t_{A_1}, t_{A_2}$, respectively, and the subsets B_1 and B_2 take the role of A_1 and A_2, respectively. The edges between B_1 and B_2 are determined by Bob's input B. The other difference is that v is adjacent to three "tails" t_v^1, t_v^2, t_v^3 (whereas u is adjacent to only two, t_v^1, t_v^2).

The subgraphs of Alice and Bob are connected by a single edge (u, v).

The graph G_K is constructed by Alice exactly as the graph G_U, the only difference is that Alice does not know the subset B, instead of B Alice uses A to determine both the edges between A_1 and A_2 and the edges between B_1 and B_2.

Clearly, if A and B are such that $(a_1^i, a_2^j) \in A$ if and only if $(b_1^i, b_2^j) \in B$, then G_U and G_K are isomorphic. We next prove that if this is not the case then G_U and G_K are not isomorphic.

We assume towards contradiction that G_K and G_U are isomorphic and that there exists $(a_1^i, a_2^j) \in A$ such that $(b_1^i, b_2^j) \notin B$ or vice versa. We first observe that it is possible to identify u and v based on the structure of G_U (i.e. regardless to the labels of the nodes) because these are the only nodes that are neighbors to 2 and 3 nodes with degree exactly 1, respectively. Next, we identify u' and v' - these are the nodes that have degree exactly 2 and are adjacent to u and v, receptively. Once we identify u' and v' we can identify u'' and v'' and in turn identify A_2 and B_2. The nodes in A_1 and B_1 are the neighbors of u and v, excluding u' and v', receptively. Finally, we are able to identify a_1^0, a_2^0, b_1^0 and b_2^0 since these are the only nodes that are adjacent to a node of degree exactly one - $t_{A_1}, t_{A_2}, t_{B_1}$ and t_{B_2}, respectively. Once we identify a_1^0 and A_1 we can identify the path $a_1^0, \dots, a_1^{k-1}$. Likewise for the paths induced on A_2, B_1 and B_2. This implies that there if there exists $(a_1^i, a_2^j) \in A$ such that $(b_1^i, b_2^j) \notin B$ or vice versa then it has to be that the graph G_U, which is constructed with correspondence to the to the pair (A, B), has to be not isomorphic to the graph G_K, which is constructed with correspondence to the to the pair (A, A). This concludes the proof. ◀

A.2 An $\Omega(D)$ lower bound for the Testing Isomorphism Problem

In this section we establish the following theorem.

▶ **Theorem 18.** *For any D there exists a family of graphs with diameter $\Theta(D)$ such that any distributed two-sided error property testing algorithm for testing isomorphism on this family of graphs requires $\Omega(D)$ rounds. This lower bound applies also in the LOCAL model and even if all nodes receive G_K as an input.*

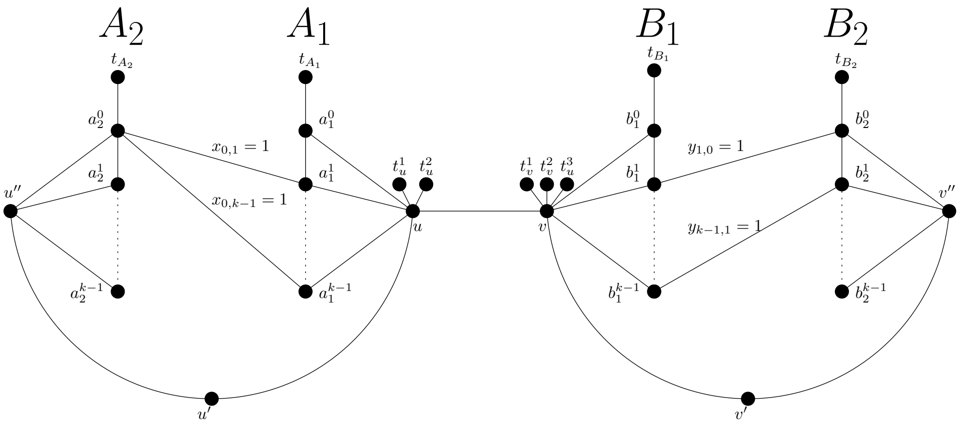

Figure 2 Deciding Isomorphism lower bound construction.

Proof. We define a family of graphs $\mathcal{H}$, where each $H \in \mathcal{H}$ takes the role of the unknown graph G_U. We fix $G_K \in \mathcal{H}$ and require from $\mathcal{H}$ that all graphs in $\mathcal{H} \setminus \{G_K\}$ are ϵ'-far from being isomorphic to G_K. Theorem 18 then follows by applying Yao's principle [47].

Construction of $\mathcal{H}$. We begin our construction with a pair of graphs on n nodes and diameter $\Theta(D)$, G_1 and G_2, such that $|E(G_1)| = (1+\epsilon)|E(G_2)|$. [12] Clearly, G_2 is ϵ-far from being isomorphic to G_1 (since G_1 has $\epsilon|E(G_2)|$ more edges than G_2). Let $\mathcal{H}$ denote a family of graphs where each graph in $\mathcal{H}$, denoted by $G_{i,j}$ for $i, j \in \{1,2\}$, is constructed by using two copies from $\{G_1, G_2\}$. The definition of $G_{i,j}$ is as follows:

1. Let $p \triangleq (p_1, \ldots, p_D)$ denote a path of D nodes.
2. Let v_i and v_j denote arbitrary nodes in G_i and G_j, respectively.
3. Identify, v_i with p_1 and v_j with p_D, to obtain $G_{i,j}$. Observe that $|V(G_{i,j})| = 2n + D - 2 = O(n)$, and that the diameter of $G_{i,j}$ is $\Theta(D)$.

Note that from the reasoning above, for every $G_{i,j}, G_{a,b} \in \mathcal{H}$, where $\{i,j\} \neq \{a,b\}$, $G_{i,j}$ is $\Omega(\epsilon)$-far from being isomorphic to $G_{a,b}$.

We assume towards a contradiction that there is a two-sided tester $\mathcal{A}$ that is correct with probability at least $2/3$ and runs for $D(G_U)/3$ rounds, where $D(G_U)$ denotes the diameter of the distributed network G_U. The lower bound then follows from Yao's principle [47]. We next specify the distribution over the inputs and explicitly apply Yao's principle.

We define a set of inputs to the distributed tester: (1) we fix the known graph G_K to be $G_{1,2}$, (2) the unknown graph G_U is chosen randomly from the distribution $\tilde{\mathcal{H}}$ which is defined as follows: the probability to obtain the graph $G_{1,2}$ is $1/2$ and each of the graphs $G_{1,1}$ and $G_{2,2}$ is obtained with probability $1/4$. Now we consider two cases. The first case is that $\mathcal{A}$ outputs (correctly) yes when G_U is $G_{1,2}$. By the definition of the problem this implies that all nodes output yes. Since the number of rounds of the tester is at most $D/3$, it follows that all nodes in both graphs $G_{1,1}$, and $G_{2,2}$, output yes. To verify this observe that for any node u in either $G_{1,1}$ or $G_{2,2}$ there exists a node in $G_{1,2}$, v, such that the $(D/3)$-hop

[12] Observe that such graphs exist for any D for G_1 and G_2 that are dense. On the other extreme, if we want G_1 and G_2 to be bounded degree trees then it is possible to obtain such graphs for any $D = \Omega(\log n)$.

neighborhoods of v and u are the same (up to labels and port numbers [13]). Thus, in this case, $\mathcal{A}$ errs on $G_{1,1}$ and $G_{2,2}$. In the second case $\mathcal{A}$ outputs no when G_U is $G_{1,2}$. Thus, in both cases $\mathcal{A}$ errs with probability at least $1/2$ when G_U is drawn according to $\tilde{\mathcal{H}}$.

By Yao's principle it is implied that any randomized tester must err with probability at least $1/2$ as well, in contradiction to our assumption that $\mathcal{A}$ is correct with probability of at least $2/3$. ◀

B Simulating Centralized Property Testing Algorithms

In this section we prove the following claim.

▷ Claim 19. Let $\mathcal{A}$ be a centralized property testing algorithm that is allowed to make adjacency queries, incidence queries and degree queries. There exists a distributed algorithm that can simulate $\mathcal{A}$ in $O(D \cdot q)$ rounds if $\mathcal{A}$ is adaptive and in $O(D + q)$ rounds if $\mathcal{A}$ is non-adaptive, where D denotes the diameter of the graph and q denotes the number of queries that $\mathcal{A}$ makes.

Proof. In order to simulate $\mathcal{A}$ on the distributed network we first pick a leader r and construct a BFS tree rooted at r in D rounds. We also assume w.l.o.g. that r knows the size of the network, n. If $\mathcal{A}$ is non-adaptive then r can determine the queries that $\mathcal{A}$ makes by using (only) the knowledge of n and random bits. The root r sends this information to the entire network in $D + q$ rounds and in additional $D + q$ rounds the answers are gathered by pipelining back at r. Once that all the answers are gathered, r can complete the simulation of $\mathcal{A}$. The outcome is then sent to all the nodes in the network. If $\mathcal{A}$ is adaptive then r simulates $\mathcal{A}$ step by step where each query is gathered in $2D$ rounds. This yields a round complexity of $O(D \cdot q)$, as desired. ◁

Application to minor-free graphs and Outerplanar graphs. For graphs which are minor-free with a bound $d = O(1)$ on the degree Newman and Sohler [42] argued the following in the centralized property testing model: (a) graph isomorphism is possible with constant number of queries, and (b) any property is testable with constant number of queries in the respective families of graphs. The latter means that there is an algorithm that performs a constant number of queries and succeeds with constant probability. Here, "constant" means independent of n.

For graphs which are forests and outerplanar (which include forests), Kusumoto and Yoshida [38] and Babu, Khoury, and Newman [3] argued, respectively, the following in the centralized property testing model: (a) graph isomorphism is possible with $\operatorname{poly}\log n$ number of queries, and (b) any property is testable with the same number of queries in the respective families of graphs. The latter means that there is an algorithm that performs $\operatorname{poly}\log n$ number of queries and succeeds with constant probability.

Their results are summarized in the following theorems.

▶ **Theorem 20** ([42, Thm. 3.2, 3.3]). *Given an oracle access to a minor-free graph with maximum degree $d = O(1)$, any graph property is testable with constant number of queries. This testing algorithm succeeds with constant probability. Specifically, testing isomorphism of two such graphs can be done in a constant number of queries and with constant probability.*

[13] We assume that the output of the algorithm is invariant to the labeling of the nodes and the port numbers of the edges. In the full version we remove this assumption and provide a general and more detailed proof.

▶ **Theorem 21** ([38, Thm. 1.3, 1.1], [3, Thm. 4.3, 4.2]). *Given an oracle access to a k-edge-outerplanar graph, any graph property is testable with* $\mathrm{poly}\log n$ *queries.* [14] *This testing algorithm succeeds with constant probability. Specifically, testing isomorphism of two k-edge-outerplanar graph can be done in* $\mathrm{poly}\log n$ *queries and with constant probability.*

Given the simulation argument above and Theorems 20, 21, we obtain the following corollaries.

▶ **Corollary 22.** *Any property is testable in the CONGEST model for minor-free graphs with maximum degree $d = O(1)$ within $O(D)$ rounds in the bounded-degree model. Specifically, this holds for the graph isomorphism problem.*

▶ **Corollary 23.** *Any property is testable in the CONGEST model for trees and k-edge-outerplanar graphs within $\tilde{O}(D)$ rounds in the general model. Specifically, this holds for the graph isomorphism problem.*

C Missing proofs and details

C.1 Proof of Claim 8

Proof. Let u and v be such that $|\triangle(N(u), N(v))| \geq \beta n$. The probability that C does not contain a node from $|\triangle(N(u), N(v))|$ is at most $(1 - \beta)^s \leq \delta/n^2$, for an appropriate setting of s. Therefore, the claim follows by a union bound over all pairs u, v. ◀

C.2 Proof of Lemma 9

Proof. Let C be an ϵ-separating sequence of the nodes of G. Since π is an isomorphism between G are H, it follows that $\pi(C)$ is ϵ-separating in H (since for any u, v that have the same $\pi(C)$-label in H it must hold that $\pi^{-1}(u), \pi^{-1}(v)$ have the same C-label in G). Thus, by definition, for any pair of nodes v, u such that $\ell^H_{\pi(C)}(v) = \ell^H_{\pi(C)}(u)$ it holds that $|\triangle(N_H(u), N_H(v))| < \epsilon n$. Let f be $(C, \pi(C))$-label-consistent. By definition, $\ell^G_C(v) = \ell^H_{\pi(C)}(f(v))$, for every $v \in V(G)$. Therefore, there exists a bijection $g : V(H) \to V(H)$ such that $f = g \circ \pi$ and g only maps between nodes with the same $\pi(C)$-label. Thus, f can be obtained from π by making at most n swaps, one by one, between elements of $V(H)$ that have the same $\pi(C)$-label. Since each swap changes the adjacency matrix by at most ϵn, we obtain the desired result. ◀

C.3 Proof of Claim 10

Proof. Consider the outcome of querying the adjacency-matrices of G and H in $y \triangleq \Theta(\log(1/\delta)/\epsilon)$ random locations. Define the random variables $\{x_i\}_{i\in[y]}$ as follows: $x_i = 1$ if the values in the i-th location of both matrices are the same, and 0 otherwise. Let $p \triangleq \Delta(G, H)/n^2$ and define $\hat{p} = \sum_{i=1}^{y} x_i/|y|$. If $p = \epsilon/2$, then by the Multiplicative Chernoff's Bound, the probability that $\hat{p} > (3\epsilon)/2$ is at most δ (for the right setting of the parameter in the Θ-notation). Clearly, the same is true if $p < \epsilon/2$. On the other hand, if $p > \epsilon$ then by the Multiplicative Chernoff's Bound, the probability that $\hat{p} \leq (3\epsilon)/2$ is at most δ. Therefore, by accepting if and only if $\hat{p} \leq (3\epsilon)/2$ we can distinguish $\Delta(G, H) = p \cdot n^2 > \epsilon n^2$ from $\Delta(G, H) \leq \epsilon n^2/2$, as desired. ◀

[14] A graph G is 1-edge-outerplanar if it has a planar embedding in which all vertices of G are on the outer face. A graph G is k-edge-outerplanar if G has a planar embedding such that if all edges on the exterior face are deleted, the connected components of the remaining graph are all $(k-1)$-edge-outerplanar.

C.4 Selecting s Nodes u.a.r.

The procedure proceeds as follows. Each node selects a random number in $[n^{c+2}]$ where c is an absolute constant. For a fixed pair of nodes, the probability that both nodes pick the same number is at most $1/n^{c+2}$. Therefore, by union bound over all pairs, with probability at least $1 - 1/n^c$, all selected numbers are distinct. Conditioned on this event, the nodes with the s highest numbers are distributed uniformly at random. Each node sends its ID and its selected number up the BFS tree and the messages are forwarded up the tree in a manner that prioritizes messages whose number is higher. Therefore, the root receives the s highest numbers (along with the IDs of the corresponding nodes) in $D + s$ rounds. To see this observe that the message with the highest number is never delayed and in general the message with the i-highest number may be delayed for at most $i - 1$ rounds.

C.5 Matching of Y to Y'

We next describe the matching of Y to Y' according to g and explain how it is computed distributively. We aim to assign to each node in Y a label in $[|Y|]$ uniquely. This way each node in Y can match itself to a node in Y' (recall that all nodes know V_K and the total order on V_K). To this end, we use the BFS tree as follows. Each node in the BFS tree computes how many nodes in its subgraph are in Y. This can be done in $O(D)$ rounds as follows. We assume w.l.o.g. that each node knows its layer in the BFS tree. Let b denote the number of layers. We proceed in b rounds. In the first round, every node in Y which is in the b-th layer sends to its parent the message 1. In the next round, all the nodes in layer $b - 1$ sum up the messages they received and add 1 if they belong to Y. Then they send the result to their parents and so on until we end up at the root. Now the root partitions the interval $[1, \ldots, |Y|]$ into consecutive sub-intervals and assigns these sub-intervals to its children. Each child receives an interval whose size equals to the number of nodes in its subgraph that are in Y. In a similar manner, these sub-intervals are partitioned recursively down the tree until each node in Y is assigned with a unique number in $|Y|$, as desired.