Journal of Computer and System Sciences 113 (2020) 1–17

Contents lists available at ScienceDirect

Journal of Computer and System Sciences

www.elsevier.com/locate/jcss

Graph reconstruction in the congested clique [☆]

P. Montealegre [a,*], S. Perez-Salazar [b], I. Rapaport [c], I. Todinca [d]

[a] Facultad de Ingeniería y Ciencias, Universidad Adolfo Ibáñez, Santiago, Chile
[b] ISyE, Georgia Institute of Technology, Atlanta, USA
[c] DIM-CMM (UMI 2807 CNRS), Universidad de Chile, Santiago, Chile
[d] Université d'Orléans, INSA Centre Val de Loire, LIFO EA 4022, France

ARTICLE INFO

Article history:
Received 12 July 2018
Received in revised form 10 April 2020
Accepted 17 April 2020
Available online 6 May 2020

Keywords:
Distributed computing
Congested clique
Round complexity
Reconstruction problem
Graph classes

ABSTRACT

In this paper we study the *reconstruction problem* in the congested clique model. Given a class of graphs $\mathcal{G}$, the problem is defined as follows: if $G \notin \mathcal{G}$, then every node must reject; if $G \in \mathcal{G}$, then every node must end up knowing all the edges of G. The cost of an algorithm is the total number of bits received by any node through one link. It is not difficult to see that the cost of *any algorithm* that solves this problem is $\Omega(\log|\mathcal{G}_n|/n)$, where $\mathcal{G}_n$ is the subclass of all n-node labeled graphs in $\mathcal{G}$. We prove that the lower bound is tight and that it is possible to achieve it with only 2 rounds.

1. Introduction

The *congested clique* model –a message-passing model of distributed computing introduced by Lotker, Patt-Shamir, Pavlov, and Peleg [26]– is receiving increasingly more attention [5,12–14,16,20]. In the congested clique, the underlying communication network is the complete graph. Therefore, in this network of diameter 1, the issue of *locality* is taken out of the picture. The focus is set on *congestion*, which is, together with locality, the main issue in distributed computing. The point is the following: if the communication network is a complete graph and the cost of local computation is ignored, then the only obstacle to perform any task is due to congestion alone.

Despite the theoretical motivation of the congested clique model, examples of distributed and parallel systems, where the efficiency depends heavily on the bandwidth and therefore might benefit from our results, are becoming increasingly common. For instance, in [17], the authors show that fast algorithms in the congested clique model can be translated into fast algorithms in the MapReduce model. MapReduce is a well-known, popular parallel-programming framework for processing large scale data [6]. Other similar systems are Pregel [27], Spark [35], Hadoop [34], Dryad [19], etc.

Many theoretical models, aiming to bridge the gap between theory and previously mentioned softwares, have emerged: the Massively Parallel Communication model [2], the MapReduce Computation model [22], the k-Machine model [23]. There is also a precursory model: the Bulk-Synchronous Parallel model of Valiant [33]. All these models are very similar, but not completely identical, to the congested clique model.

☆ A preliminary version of part of this work appeared in the proceedings of the 25th International Colloquium on Structural Information and Communication Complexity SIROCCO 2018, held in Ma'ale HaHamisha, Israel.

* Corresponding author.

E-mail addresses: p.montealegre@edu.uai (P. Montealegre), sperez@gatech.edu (S. Perez-Salazar), rapaport@dim.uchile.cl (I. Rapaport), ioan.todinca@univ-orleans.fr (I. Todinca).

https://doi.org/10.1016/j.jcss.2020.04.004
0022-0000/© 2020 Elsevier Inc. All rights reserved.

1.1. The model

The congested clique model is defined as follows. There are n nodes which are given distinct identities (IDs), that we assume for simplicity to be numbers between 1 and n. In this paper we consider the situation where the joint input to the nodes *is a graph* G. More precisely, each node v receives as input an n-bit boolean vector $x_v \in \{0,1\}^n$, which is the indicator function of its neighborhood in G. Note that the input graph G is an arbitrary n-node graph, *a subgraph of the communication network* K_n. Nodes execute an algorithm, communicating with each other in synchronous rounds and their goal is to compute some function f that depends on G. In every round, each of the n nodes may send up to $n-1$ different b-bit messages through each of its $n-1$ communication links. When an algorithm stops *every node must know* $f(G)$. We call $f(G)$ the *output* of the distributed algorithm. The parameter b is known as the *bandwidth* of the algorithm. We denote by R the *number of rounds*. The product $R \cdot b$ corresponds to the total number of bits received by a node through one link, and we call it the *cost* of the algorithm.

An algorithm may be deterministic or randomized. We distinguish two sub-cases of randomized algorithms: the private-coin setting, where each node flips its own coin; and the public-coin setting, where the coin is shared between all nodes. An ε-error algorithm $\mathcal{A}$ that computes a function f is a randomized algorithm such that, for every input graph G, $\Pr(\mathcal{A} \text{ outputs } f(G)) \geq 1-\varepsilon$. In the case where $\varepsilon \to 0$ as $n \to \infty$, we say that $\mathcal{A}$ computes f with high probability (w.h.p.). Recall that in this model all nodes compute the same output; therefore, when we run the algorithm either all nodes compute the right answer or none of them does.

The function f defines the problem to be solved; a 0/1-valued function corresponds to a decision problem (such as connectivity [16]). For other, more general types of problems, f may be more appropriately defined as a relation. This happens, for instance, when we want to construct a minimum spanning tree [14,20], a maximal independent set [13], a 3-ruling set [18], all-pairs shortest-paths [5], etc. We remark that in these references the model does not consider the restriction that all vertices have to know $f(G)$ when the algorithm stops. For instance, in the minimum spanning tree problem each vertex has to output only the subset of its incident edges that belong to the tree.

1.2. The reconstruction problem

The most difficult problem one could attempt to solve is the *reconstruction problem*, where nodes are asked to reconstruct the input graph G. In fact, if at the end of the algorithm every node has full knowledge of G, then, since nodes have unbounded computational power, they could answer *any question* concerning G.

In centralized, classical graph algorithms, a widely used approach to cope with NP-hardness is to restrict the class of graphs where the input G belongs. We use an analogous approach here, in the congested clique model. But, as we explain later, surprisingly, the complexity of the reconstruction problem *will only depend on the cardinality* of the subclass of n-node graphs in $\mathcal{G}$.

We introduce two problems, each defined for a fixed set of graphs $\mathcal{G}$. The first one, the *strong reconstruction problem* $\mathcal{G}$-Strong-Rec, is the following.

$\mathcal{G}$-Strong-Rec

Input: An arbitrary graph G

Output: $\begin{cases} \text{all the edges of } G & \text{if } G \in \mathcal{G}; \\ \text{reject} & \text{otherwise.} \end{cases}$

Recall that the output is computed by *every node* of the network. In other words, every node of an algorithm that solves $\mathcal{G}$-Strong-Rec must end up knowing whether G belongs to $\mathcal{G}$; and, in the positive cases, every node also finishes knowing all the edges of G.

We also define a *weak reconstruction problem* $\mathcal{G}$-Weak-Rec. This is a promise problem, where the input graph G is promised to belong to $\mathcal{G}$. In other words, for graphs that do not belong to $\mathcal{G}$, the behavior of an algorithm that solves $\mathcal{G}$-Weak-Rec does not matter.

$\mathcal{G}$-Weak-Rec

Input: $G \in \mathcal{G}$

Output: all the edges of G

What is already known about the (strong) reconstruction problem? The answer to this question is that there exist mainly three different types of algorithms for tackling this problem.

First, we have Lenzen's algorithm, which performs a load balancing procedure [24]. It simply distributes all the edges among the nodes, and then broadcasts everything. Therefore, if the class of input graphs $\mathcal{G}$ consists of a subset of *sparse graphs* then such algorithm solves the reconstruction problem very quickly. For instance, if the input graph G contains $\mathcal{O}(n)$ edges, then Lenzen's algorithm reconstruct G in a constant number of rounds with bandwidth $\mathcal{O}(\log n)$.

The second approach has been devised for graphs with bounded degeneracy. Recall that a graph G is d-degenerate if one can remove from G a vertex r of degree at most d, and then proceed recursively on the resulting graph $G' = G - r$, until obtaining the empty graph. Bounded genus graphs such as planar graphs, bounded tree-width graphs, graphs without a fixed graph H as a minor, are all examples of classes of graphs with bounded degeneracy. In [3,29] the authors exhibit a one-round deterministic algorithm that solves $\mathcal{G}$-Strong-Rec using bandwidth $\mathcal{O}(d \log n) = \mathcal{O}(\log n)$, where $\mathcal{G}$ is the class of d-degenerate graphs.

Note that the class of d-degenerate graphs is simultaneously sparse and *hereditary*. A class $\mathcal{G}$ is hereditary if, for every graph $G \in \mathcal{G}$, every induced subgraph of G also belongs to $\mathcal{G}$. Many graph classes are hereditary: forests, planar graphs, bipartite graphs, k-colorable graphs, bounded tree-width graphs, etc. [4]. Moreover, any intersection class of graphs –such as interval graphs, chordal graphs, unit disc graphs, etc.– is also hereditary [4].

The third type of result is related to the reconstruction of particular hereditary graph classes. Authors have studied hereditary classes defined by one fixed forbidden graph H. A graph G is called H-free if H is not an induced subgraph of G. In [21] the authors studied the (one-round) reconstruction problem of H-free classes, for all possible graphs H. In particular, they consider the case when H is the path of 4 nodes P_4. They exhibited a one-round, public coin algorithm for solving $\mathcal{G}$-Strong-Rec using bandwidth $\mathcal{O}(\log n)$, where $\mathcal{G}$ is the class of P_4-free graphs, known as *cographs*.

Finally, one can also define a class of graph by forbidding H as a subgraph (instead of forbidding it as an *induced subgraph)*. Any class defined like this is also hereditary. In [7], the authors prove that the degeneracy of the class of graphs defined by forbidding the subgraph H is upper bounded by $4 \cdot ex(n, H)/n$, where $ex(n, H)$ is the *Turán number* of H, defined as the maximum number of edges in an n-node graph not containing an isomorphic copy of H as a subgraph. Therefore, the class of graphs defined by a forbidden subgraph H can be reconstructed deterministically in one round with bandwidth $\mathcal{O}((ex(n, H) \log n)/n)$.

1.3. A key remark

In this short subsection we explain a key aspect of this paper. For any positive integer n, let us define the set $\mathcal{G}_n$ as the set of all n-node graphs in $\mathcal{G}$.

There is an obvious lower bound for $R \cdot b$, even for the weak reconstruction problem $\mathcal{G}$-Weak-Rec and even in the public-coin setting. In fact,

$$R \cdot b = \Omega(\log |\mathcal{G}_n|/n).$$

This can be seen by noting that, in the randomized case, there must be at least one outcome of the coin tosses for which the correct algorithm reconstructs the input graph in at least $(1 - \varepsilon)$ of the cases. In fact, if this was not true, then we would be contradicting the correctness of the algorithm, which states that for every input graph, in at least $(1 - \varepsilon)$ of the coin tosses, the answer is correct. A useful way to see this is to visualize a matrix where the rows correspond to coin tosses, the columns to input graphs, and each entry is the output given by the algorithm to the corresponding graph and the corresponding sequence of coin tosses.

Now we count the number of bits received by any node. By previous argument, it must be $\Omega(\log((1 - \varepsilon) \cdot |\mathcal{G}_n|)) = \Omega(\log |\mathcal{G}_n|)$. This holds because in at least one outcome of the coin tosses, the nodes must reconstruct that many different graphs.

On the other hand, the total number of bits received by any node v of the network is $(n-1) \cdot R \cdot b + n$. In fact, $(n-1) \cdot R \cdot b$ bits are received from the other nodes and n bits are known by v at the beginning of the algorithm. Therefore, putting all together, we have $n + (n - 1) \cdot R \cdot b = \Omega(\log |\mathcal{G}_n|)$. This implies that $R \cdot b = \Omega(\log |\mathcal{G}_n|/n)$.

In this paper we prove that this bound is essentially tight, even with $R = 1$ (if $\mathcal{G}$ is an hereditary class of graphs) and $R = 2$ (in the general case).

1.4. Our results

In Section 3 we give, for every hereditary class of graphs $\mathcal{G}$, a one-round private-coin randomized algorithm that solves $\mathcal{G}$-Strong-Rec with bandwidth $\mathcal{O}(\max_{k \in [n]} \log |\mathcal{G}_k|/k + \log n)$.

We emphasize that our algorithm runs in one round, and therefore it runs in the *broadcast congested clique*, a restricted version of the congested clique model where, in every round, the $n - 1$ messages sent by a node must be the same. This equivalence –which is a consequence of the requirement that *all nodes* must compute the output after one round– is proved in Section 2. We also remark that for many hereditary graph classes our algorithm is tight. More precisely, our result implies that $\mathcal{G}$-Strong-Rec can be solved in one round with bandwidth $\mathcal{O}(\log n)$ when $\mathcal{G}$ is the class of forests, planar graphs, d-degenerate graphs, interval graphs, unit-circle graphs, or any other hereditary graph class $\mathcal{G}$ such that $|\mathcal{G}_n| = 2^{\mathcal{O}(n \log n)}$.

In Section 4 we give a very general result, showing that two rounds are sufficient to solve $\mathcal{G}$-Strong-Rec in the congested clique model, for any set of graphs $\mathcal{G}$. More precisely, we provide a two-round deterministic algorithm that solves $\mathcal{G}$-Weak-Rec and a two-round private-coin randomized algorithm that solves $\mathcal{G}$-Strong-Rec w.h.p. We also give a three-round deterministic algorithm solving $\mathcal{G}$-Strong-Rec. All algorithms run using bandwidth $\mathcal{O}(\log |\mathcal{G}_n|/n + \log n)$, so they are asymptotically optimal when $|\mathcal{G}_n| = 2^{\Omega(n \log n)}$.

Our result implies, in particular, that $\mathcal{G}$-Strong-Rec can be solved in two rounds with bandwidth $\mathcal{O}(\log n)$, when $\mathcal{G}$ is *any set* of graphs of size $2^{\mathcal{O}(n\log n)}$. The only property of the set of graphs $\mathcal{G}$ used by our algorithm is the cardinality of $\mathcal{G}_n$. Our algorithm does not require $\mathcal{G}$ to be closed under isomorphisms.

In Section 5 we revisit the one-round case. Our general algorithm can be adapted to run in one round (i.e., in the broadcast congested clique model) by allowing a larger bandwidth. We show that, for every set of graphs $\mathcal{G}$, there is a one-round deterministic algorithm that solves $\mathcal{G}$-Weak-Rec, and a one-round private-coin algorithm that solves $\mathcal{G}$-Strong-Rec w.h.p., both of them using bandwidth $\mathcal{O}(\sqrt{\log|\mathcal{G}_n|\log n}+\log n)$. We finish Section 5 pointing out that our algorithms are tight with respect to the bandwidth as well as the requirement of randomness.

Even though in the congested clique model, by definition, the only complexity measure taken into account is communication, it is important to point out that the general algorithms we present in this paper *might* run in exponential local time.

Note, however, that unless $P=NP$ (or even if stronger conjectures in computational complexity are false), this difficulty can not be overcome. In fact, for many graph classes $\mathcal{G}$, solving $\mathcal{G}$-Strong-Rec in polynomial local time is impossible.

Let us illustrate this with an example. Consider the *hereditary, sparse* class of 3-colorable planar graphs, that we denote **3-col-plan**. It is NP-complete to decide whether an arbitrary graph belongs to **3-col-plan** [11]. Any algorithm in the congested clique model that runs in polynomial local time can be simulated by a sequential algorithm that also runs in polynomial time: simply run the computation of each node one by one at each round. Therefore, unless $P=NP$, there is no algorithm running in polynomial local time solving **3-col-planar**-Strong-Rec. Nevertheless, when the class of graphs is decidable in (centralized) polynomial time, there is no reason, a priori, preventing us from finding one-round, polynomial local time reconstruction algorithms in the congested clique model.

The remark above motivates the study of the following question: for what graph classes $\mathcal{G}$ the reconstruction problem $\mathcal{G}$-Strong-Rec can be solved in one round, bandwidth $\mathcal{O}(\log|\mathcal{G}_n|/n)$ and *polynomial* local time? In Section 6 we address this question by devising one-round, public-coin algorithms for reconstructing two hereditary classes of graphs: distance-hereditary graphs and graphs of bounded modular width. Both algorithms use bandwidth $\mathcal{O}(\log n)$, run in polynomial local time, and give the correct answer w.h.p.

We point out that, with respect to the local computation time, there exists an example of a conditional lower-bound. In fact, Drucker et al. [7] show that, assuming the Exponential Time Hypothesis, any randomized algorithm solving triangle-freeness in the broadcast congested clique model such that the local computation time is sub-exponential, has communication cost $\Omega(n/e^{\sqrt{\log n}})$. The existence of similar lower-bounds for the reconstruction of graph classes is an interesting and challenging open question.

2. Preliminaries

2.1. One-round algorithms

The *broadcast congested clique* model is a restricted version of the congested clique model where each node is forced, in each round, to send the *same* message through its $n-1$ communication links. But, if we consider one-round algorithms, the two models are the same. In fact, suppose that there is a one-round algorithm $\mathcal{A}$ (deterministic or randomized) in the congested clique with bandwidth b. We can transform it into an algorithm $\mathcal{B}$ in the broadcast version with bandwidth $b+1$ as follows. We fix a vertex, say the one with ID 1, and every node j broadcasts the message it would send to node 1 on algorithm $\mathcal{A}$, plus one bit indicating whether node j and node 1 are adjacent in G. After this communication round of $\mathcal{B}$, every node knows the messages node 1 would have received after the communication round of algorithm $\mathcal{A}$. Moreover, every node knows the neighborhood of node 1. The result follows from the fact that, with this information, node 1 knows the output. Obviously, as we will see in this paper, when multi-round algorithms are considered, the broadcast congested clique model is much less powerful than the congested clique model.

2.2. Some graph terminology

Two graphs G and H are *isomorphic* if there exists a bijection $\varphi: V(G)\to V(H)$ such that any pair of vertices u,v are adjacent in G if and only if $\varphi(u)$ and $\varphi(v)$ are adjacent in H. A *class of graphs* $\mathcal{G}$ is a set of graphs which is *closed under isomorphisms*, i.e., if G belongs to $\mathcal{G}$ and H is isomorphic to G, then H also belongs to $\mathcal{G}$. For a class of graphs $\mathcal{G}$ and $n>0$, we call $\mathcal{G}_n$ the subclass of n-node graphs in $\mathcal{G}$. For a graph $G=(V,E)$ and $U\subseteq V$ we denote by $G[U]$ the subgraph of G induced by U. More precisely, the vertex set of $G[U]$ is U and the edge set consists of all of the edges in E that have both endpoints in U. A class $\mathcal{G}$ is *hereditary* if it is closed under taking induced subgraphs, i.e., for every $G=(V,E)\in\mathcal{G}$ and every $U\subseteq V$, the induced subgraph $G[U]\in\mathcal{G}$.

For a graph $G=(\{v_1,\dots,v_n\},E)$, we call $A(G)$ its *adjacency matrix*, i.e., the $0-1$ symmetric square matrix of dimension n where $[A(G)]_{ij}=1$ if and only if v_i is adjacent to v_j. Let M be a square matrix of dimension n, and let $i\in[n]=\{1,\dots,n\}$. We call M_i the i-th row of M. Let N be another square matrix of dimension n. We denote by $d_r(M,N)$ the *row-distance* between M and N, that is, the number of rows that are different between M and N. In other words, $d_r(M,N)=|\{i\in[n]: M_i\neq N_i\}|$. For $k>0$ and $G=(V,E)$, we denote by $D(G,k)$ the set of all graphs $H=(V,E')$ such that $d_r(A(G),A(H))=k$.

2.3. Fingerprints

A fingerprint is a small representation of a large vector which satisfies that, if two vectors are different, then their fingerprints, w.h.p., are also different [32]. We define here the *fingerprint of a graph*, which is simply the collection of fingerprints of the rows of its adjacency matrix.

Let n be a positive integer and p be a prime number. In the following, we denote by $\mathbb{F}_p$ the finite field of size p and by $\mathbb{F}_p[X]$ the polynomial ring on $\mathbb{F}_p$. A polynomial $P \in \mathbb{F}_p[X]$ is an expression of the form $P(x) = \sum_{i=1}^{d} a_i x^{i-1}$, where $a_i \in \mathbb{F}_p$.

Let q be a prime number such that $q < n < p$. For each $a \in \mathbb{F}_q^n$, consider the polynomial $FP(a, \cdot) \in \mathbb{F}_p[X]$ defined as $FP(a, x) = \sum_{i \in [n]} a_i x^{i-1} \mod p$, where we interpret the coordinates of a as elements of $\mathbb{F}_p$.

For $t \in \mathbb{F}_p$, we call $FP(a, t)$ the *fingerprint* of a and t. The following lemma is direct.

Lemma 1. *[25] Let n be a positive integer, p and q be two prime numbers such that $q < n < p$. Let $a, b \in (\mathbb{F}_q)^n$ such that $a \neq b$. Then, $|\{t \in \mathbb{F}_p : P(a, t) = P(b, t)\}| \leq n - 1$.*

We extend the definition of fingerprints to matrices. Let M be a square matrix of dimension n and coordinates in $\mathbb{F}_q$, and let T be an element of $(\mathbb{F}_p)^n$. We call $FP(M, T) \in (\mathbb{F}_p)^n$ the *fingerprint of M and T*, defined as $FP(M, T) = (FP(M_1, T_1), \ldots, FP(M_n, T_n))$, where M_i is the i-th row of M, for each $i \in [n]$.

For a graph of size n, and $T \in (\mathbb{F}_p)^n$ we denote by $FP(G, T)$ the fingerprint of $A(G)$ and T.

3. Reconstructing hereditary graph classes

Suppose that $T = (T_1, \ldots, T_n) \in (\mathbb{F}_p)^n$ is such that every T_i is picked uniformly at random. In that case, the fingerprints of two n-node graphs H and G, that correspond to $F(G, T)$ and $F(H, T)$, are random vectors. These fingerprints are different with a probability that grows exponentially with respect to the number of nodes having different neighborhoods in G and H. Therefore, roughly speaking, if $\mathcal{G}$ is a set of graphs where all graphs are *very different*, then each graph in $\mathcal{G}$ will have a different fingerprint.

What happens when G differs from H only in a few nodes? We have two different answers, depending on whether: (i) the graphs belong to some hereditary class of graphs $\mathcal{G}$; (ii) the graphs are arbitrary.

In this section we address the first, hereditary case. More precisely, we prove that, for any graph G that belongs to some hereditary graph class $\mathcal{G}$, the number of graphs $H \in \mathcal{G}$ which are *close* to G (in terms of the number different rows in the corresponding adjacency matrices) is *small*. Therefore, the fingerprints will be different even for graphs which are close between themselves.

For tackling the second, general case, together with fingerprints, we use error-correcting codes (see Section 4).

Now we give the main result. Later we explain the consequences of this result for well-known hereditary graph classes.

Theorem 1. *Let $\mathcal{G}$ be an hereditary class of graphs. There exists a one-round private-coin algorithm that solves $\mathcal{G}$-Strong-Rec w.h.p. and bandwidth*

$$\mathcal{O}(\max_{k \in [n]}(\log(|\mathcal{G}_k|)/k) + \log n).$$

Proof. In the algorithm, nodes use a prime number p, whose value will be chosen later. The description of the algorithm is given in Algorithm 1.

Let T in $(\mathbb{F}_p)^n$ be picked uniformly at random. We show that, for every G, if some $H \in \mathcal{G}_n$ satisfies $FP(H, T) = FP(G, T)$, then $G = H$ w.h.p. First, note that

$$\begin{gathered}\Pr(\exists H \in \mathcal{G}_n \text{ s.t. } H \neq G \text{ and } FP(G, T) = FP(H, T)) \\ \leq \\ \textstyle\sum_{k \in [n]} \Pr(\exists H \in \mathcal{G}_n \cap D(G, k) \text{ s.t. } FP(G, T) = FP(H, T)),\end{gathered}$$

where $D(G, k)$ is the set of all graphs H such that the number of rows where the adjacency matrices of H and G differ is k.

Now suppose that $H \neq G$ and let $0 < k \leq n$ be such that H belongs to $D(G, k) \cap \mathcal{G}_n$. From Lemma 1 we deduce that $Pr(FP(G, T) = FP(H, T)) \leq \left(\frac{n}{p}\right)^k$. It follows that

$$\Pr(\exists H \in \mathcal{G}_n \cap D(G, k) \text{ s.t. } FP(G, T) = FP(H, T)) \leq \left(\frac{n}{p}\right)^k \cdot |\mathcal{G}_n \cap D(G, k)|.$$

We now claim that $|\mathcal{G}_n \cap D(G, k)| \leq \binom{n}{k}|\mathcal{G}_k|$. Indeed, we can interpret a graph H in $D(G, k)$ as a graph built by picking k vertices $\{v_1, \ldots v_k\}$ of G and then adding or removing edges between those vertices. Since $\mathcal{G}$ is hereditary, the graph induced by $\{v_1, \ldots, v_k\}$ must belong to $\mathcal{G}_k$. Therefore, $|\mathcal{G}_n \cap D(G, k)| \leq \binom{n}{k}|\mathcal{G}_k|$. This claim implies:

$$\Pr(\exists H \in \mathcal{G}_n \cap D(G,k) \text{ s.t. } FP(G,T) = FP(H,T)) \le \left(\frac{n}{p}\right)^k \cdot \left(\frac{ne}{k}\right)^k \cdot |\mathcal{G}_k| \le \left(\frac{n^2 \cdot e \cdot (|\mathcal{G}_k|)^{1/k}}{p}\right)^k.$$

Let $f : \mathbb{N} \to \mathbb{R}$ be defined as $f(n) = n \cdot \max_{k\in[n]} \frac{\log|\mathcal{G}_k|}{k}$. Note that this function is increasing, satisfies $f(n)/n \le f(n+1)/(n+1)$, and $\log|\mathcal{G}_n| \le f(n)$. Therefore, $(|\mathcal{G}_k|)^{1/k} \le 2^{f(k)/k} \le 2^{f(n)/n}$. We deduce:

$$\Pr(\exists H \in \mathcal{G}_n \text{ s.t. } H \ne G \text{ and } FP(G,T) = FP(H,T)) \le \sum_{k\in[n]} \left(\frac{n^2 \cdot e \cdot 2^{f(n)/n}}{p}\right)^k.$$

We now fix p as the smallest prime number greater than $n^4 \cdot e \cdot 2^{f(n)/n}$, and we get that with probability at least $1 - 1/n$, either $G = H$ or $F(H,T) \ne F(G,T)$, for every $H \in \mathcal{G}_n$. Hence, the algorithm solves $\mathcal{G}$-Strong-Rec w.h.p.

Note that the bandwidth required by node i in the algorithm equals the number of bits required to represent the pair $(t_i, F(x_i,t_i))$, which are two integers in $[p]$. Therefore, the bandwidth of the algorithm is

$$2\lceil \log p \rceil = \mathcal{O}(f(n)/n + \log n) = \mathcal{O}\left(\max_{k\in[n]}(\log(|\mathcal{G}_k|)/k) + \log n\right). \quad \square$$

Algorithm 1: $\mathcal{G}$-Strong-Rec when $\mathcal{G}$ is hereditary. Algorithm executed by node i.

```
1 Compute $p$, the smallest prime greater than $n^4 \cdot e \cdot 2^{f(n)/n}$, where $f(n) = n \cdot \max_{k\in[n]} \frac{\log|\mathcal{G}_k|}{k}$ ;
2 Pick $T_i \in \mathbb{F}_p$ uniformly at random using private coins ;
3 Compute $FP(x_i, T_i)$ ;
4 Communicate $FP(x_i, T_i)$ and $T_i$ ;
5 Receive $T = (T_1, \dots, T_n)$ and $FP(G, T)$ ;
6 Look for $H \in \mathcal{G}_n$ such that $FP(H,T) = FP(G,T)$;
7 If $H$ exists and is unique, output $H$. Otherwise, reject.
```

We deduce the following corollary.

Corollary 1. *Let $\mathcal{G}$ be an hereditary class of graphs, and h be an increasing function such that $|\mathcal{G}_n| = 2^{\theta(nh(n))}$. Then, our private-coin algorithm solves $\mathcal{G}$-Strong-Rec w.h.p., in one-round, with bandwidth $\Theta(\log|\mathcal{G}_n|/n + \log n)$. This matches the lower bound on the cost $R \cdot b$ (which must be satisfied even in the public coin setting).*

Proof. We simply note the existence of constants $c_1, c_2 > 0$ such that:

$$\max_{k\in[n]}(\log(|\mathcal{G}_k|)/k) \le c_2 \cdot \max_{k\in[n]} h(k) \le c_2 \cdot h(n) \le (c_2/c_1) \cdot (\log(|\mathcal{G}_n|)/n).$$

Therefore, the algorithm of Theorem 1 uses bandwidth $\mathcal{O}(\log(|\mathcal{G}_n|)/n)$. $\square$

Remark 1. In [31], Scheinerman and Zito showed that hereditary graph classes have highly-constrained growth rates. They showed (Theorem 1 in [31]) that, for any hereditary class of graphs $\mathcal{G}$, one of the following behaviors must hold: $|\mathcal{G}_n| \in \{\mathcal{O}(1), n^{\Theta(1)}, 2^{\Theta(n)}, 2^{\Theta(n\log n)}, 2^{\omega(n\log n)}\}$. Corollary 1 implies that our algorithm is tight for any hereditary class of graphs such that $|\mathcal{G}_n| = 2^{\Theta(n\log n)}$.

Remark 2. In this section we have shown that, if a class $\mathcal{G}$ is hereditary and we want to reconstruct it, we do not need the graphs in $\mathcal{G}$ to be sparse. We just need the class to be small. For example, the class of interval graphs contains very dense graphs (including the clique), but it is small, since it contains $2^{\mathcal{O}(n\log n)}$ different labeled n-node graphs. Therefore, there exists a one-round algorithm that reconstructs the class of interval graphs using bandwidth $\mathcal{O}(\log n)$.

Remark 3. Consider the class $\mathcal{G}$ of d-degenerate graphs. This class is hereditary and is such that $|\mathcal{G}_n| = 2^{\mathcal{O}(n\log n)}$. Therefore, we may conclude, from Corollary 1, the existence of a private-coin algorithm that solves $\mathcal{G}$-Strong-Rec w.h.p., in one-round, with bandwidth $\Theta(\log|\mathcal{G}_n|/n + \log n)$. Nevertheless, the reconstruction algorithm for d-degenerate graphs given in [3,29] is deterministic, and therefore that result is stronger than the general one we present here.

4. Reconstructing arbitrary graph classes

In this section we show that there exists a two-round private-coin algorithm in the congested clique model that solves $\mathcal{G}$-Strong-Rec w.h.p. and bandwidth $\mathcal{O}(\log|\mathcal{G}_n|/n + \log n)$. Our algorithm is based, roughly, on the same ideas used to reconstruct hereditary classes of graphs. But the problem we encounter is the following: while in the case of hereditary classes of graphs, we had for every graph G and $k > 0$, a bound on the number of graphs contained in $D(G,k) \cap \mathcal{G}_n$, this is not the case in an arbitrary family $\mathcal{G}$. Therefore, fingerprints alone are not enough to differentiate graphs. To cope with this obstacle, we use *Reed-Solomon error-correcting codes* [30]. Error-correcting codes map vectors into slightly larger ones, satisfying that the mapping of two different vectors differ in *many* coordinates. With this, we define *error-correcting-graphs* where, instead of vectors, we map any graph into a slightly larger one. The mapping of two different graphs will have *many* nodes with different neighborhoods. We show that the fingerprint of such mapping uniquely identifies the graphs in $\mathcal{G}$, for any $\mathcal{G}$. The advantage of our construction is that it can be computed locally by each node as a function of its neighborhood (i.e., as a function of the corresponding row of the adjacency matrix).

Definition 1. Let $0 \le k \le n$, and let q be the smallest prime number greater than $n+k$. An *error correcting code with parameters* (n,k) is a mapping $C : \{0,1\}^n \to (\mathbb{F}_q)^{n+k}$, satisfying:

1) For every $x \in \{0,1\}^n$ and $i \in [n]$, $C(x)_i = x_i$.
2) For each $x, y \in \{0,1\}^n$, $x \neq y$ implies $|\{i \in [n+k] : C(x)_i \neq C(y)_i\}| \ge k$.

For sake of completeness, we recall the construction of an error correcting code with parameters (n,k). For $x \in \{0,1\}^n$, let P_x be the unique polynomial of degree (at most) n in $\mathbb{F}_q[X]$ satisfying $P_x(i) = x_i$ for each $i \in [n]$. The function C is then defined as $C(x) = (P_x(1), \dots, P_x(n+k))$. This function satisfies properties (1) and (2). We now adapt the definition of error correcting codes to graphs.

Definition 2. For a graph G, we call $C(G)$ the square matrix of dimension $n+k$ with elements in $\mathbb{F}_q$ defined as follows.

- For each $i \in [n]$, the i-th row of $C(G)$ is $C(A(G)_i) \in (\mathbb{F}_q)^{n+k}$ (recall that $A(G)_i$ is the i-th row of the adjacency matrix of G).
- For each $i \in [k]$, the $(n+i)$-th row of $C(G)$ is the vector

  $$(C(A(G)_1)_{n+i}, \dots, C(A(G)_n)_{n+i}, \vec{0}) \in (\mathbb{F}_q)^{n+k},$$

  where $\vec{0}$ is the zero-vector of $\mathbb{F}_q^d$, and $C(x)_j \in \mathbb{F}_q$ is the j-th element of $C(x)$.

We can represent $C(x)$ as a pair $(x, \tilde{x})$, where $\tilde{x}$ belongs to $(\mathbb{F}_q)^k$. Similarly, for a graph G, we can represent $C(G)$ as the symmetric matrix:

$$C(G) = \begin{bmatrix} A(G) & \widetilde{A(G)} \\ \widetilde{A(G)}^T & 0 \end{bmatrix},$$

where $\widetilde{A(G)}$ is the matrix with rows $C(A(G)_i)_{n+1}, \dots, C(A(G)_i)_{n+k}$, with $i \in [n]$.

Remark 4. Note that $d_r(C(G), C(H)) > k$, for every two different n-node graphs H and G. Indeed, if $G \neq H$, there exists $i \in [n]$ such that $A(G)_i$ is different than $A(H)_i$. Then, by definition of C (Property 2 of Definition 1), $|\{j \in [n+k] : C(A(G))_{i,j} \neq C(A(H))_{i,j}\}| > k$. This means that $d_r(C(G), C(H)) > k$, because $C(G)$ and $C(H)$ are symmetric matrices.

Lemma 2. *Let $\mathcal{G}$ be a set of graphs, C the error correcting code with parameters (n,k), and let p be the smallest prime number greater than $(n+k)\cdot|\mathcal{G}_n|^{2/k}$. Then, there exists $T \in (\mathbb{F}_p)^{n+k}$ depending only on $\mathcal{G}$, satisfying $FP(C(G),T) \neq FP(C(H),T)$ for all different $G, H \in \mathcal{G}_n$.*

Proof. From Remark 4, we know that $d_r(C(G), C(H)) > k$, for every two different n-node graphs H and G. Then, if we pick $T \in (\mathbb{F}_p)^{n+k}$ uniformly at random we have, from Lemma 1:

$$\Pr(FP(C(G),T) = FP(C(H),T)) < \left(\frac{n+k}{p}\right)^k.$$

Then, by the union bound

$$\Pr(\exists G, H \in \mathcal{G}_n \text{ s.t. } G \neq H \text{ and } FP(C(G),T) = FP(C(H),T)) < \left(\frac{n+k}{p}\right)^k \cdot |\mathcal{G}_n|^2 \leq 1.$$

The last inequality follows from the choice of p. Therefore, there must exist a $T \in (\mathbb{F}_p)^{n+k}$ such that $FP(C(G),T) \neq FP(C(H),T)$, for all different $G, H \in \mathcal{G}_n$. □

Theorem 2. *Let $\mathcal{G}$ be a set of graphs. The following holds:*

1) *There exists a two-round deterministic algorithm in the congested clique model that solves $\mathcal{G}$-Weak-Rec with bandwidth $\mathcal{O}(\log|\mathcal{G}_n|/n + \log n)$.*
2) *There exists a three-round deterministic algorithm in the congested clique model that solves $\mathcal{G}$-Strong-Rec with bandwidth $\mathcal{O}(\log|\mathcal{G}_n|/n + \log n)$.*
3) *There exists a two-round private-coin algorithm in the congested clique model that solves $\mathcal{G}$-Strong-Rec with bandwidth $\mathcal{O}(\log|\mathcal{G}_n|/n + \log n)$ w.h.p.*

Proof. The first algorithm we explain here, Algorithm 2, is deterministic and solves $\mathcal{G}$-Weak-Rec with bandwidth $\mathcal{O}(\log|\mathcal{G}_n|/n + \log n)$. The algorithms for (2) and (3) are slight modifications of Algorithm 2 and will also be explained in this proof.

1) Let p be the first prime greater than $2n \cdot |\mathcal{G}_n|^{2/n}$ (then $p \leq 4n \cdot |\mathcal{G}_n|^{2/n}$), and let q be the smallest prime number greater than $2n$. In the algorithm, node i first computes $C(x_i)$, where C is the error correcting code with parameters (n,n). Then, for each $j \in [n]$ node i communicates $C(x_i)_{j+n}$ to node j. This communication round requires bandwidth $\lceil \log q \rceil = \mathcal{O}(\log n)$. After the first communication round, node i knows $C(x_i)$ and $(C(x_1)_{i+n}, \dots, C(x_n)_{i+n})$, i.e., it knows rows i and $i+n$ of matrix $C(G)$. Each node computes a vector $T \in (\mathbb{F}_p)^{2n}$ such that $FP(C(G),T) \neq FP(C(H),T)$, for all different $G, H \in \mathcal{G}_n$ (each node computes the same T). The existence of T is given by Lemma 2. Then, node i communicates (broadcasts) $FP(C(G)_i, T_i)$ and $FP(C(G)_{i+n}, T_{i+n})$. This communication round requires bandwidth $2\lceil \log p \rceil = \mathcal{O}((\log|\mathcal{G}_n|)/n + \log n)$. After the second communication round, each node knows $FP(C(G),T)$. Then, they locally compute the unique $H \in \mathcal{G}_n$ such that $FP(C(H),T) = FP(C(G),T)$. Since G belongs to $\mathcal{G}_n$, then necessarily $G = H$.

2) Suppose now that we are solving $\mathcal{G}$-Strong-Rec. In this case G does not necessarily belong to $\mathcal{G}_n$. After receiving the fingerprints of $C(G)$, the nodes look for a graph H in $\mathcal{G}_n$ that satisfies $FP(C(G),T) = FP(C(H),T)$ (line 9 in Algorithm 2). If such a graph exists, we call it a *candidate*. Otherwise, every node decides that G is not in $\mathcal{G}_n$, so they *reject*. Note that, if the candidate exists, then it is unique, since $FP(C(H_1),T) \neq FP(C(H_2),T)$ for all different H_1, H_2 in $\mathcal{G}_n$. So, if the candidate H exists, each node i checks whether the neighborhood of vertex i on G and H are equal, and announces the answer in the third round (communicating one bit). If every node announces affirmatively, then they output $G = H$. Otherwise, it means that G is not in $\mathcal{G}_n$, so every node *rejects*.

3) We now show that, if we allow the algorithm to be randomized, then we can spare the third round. In fact, nodes only need to run Algorithm 3 after the first round of Algorithm 2. Let us explain this now. Let $p' \in [n^2, 2n^2]$ be a prime number. In the second round, node i picks $S_i \in \mathbb{F}_p$, and it communicates, together with $FP(C(G)_i, T_i)$ and $FP(C(G)_{i+n}, T_{i+n})$, also $FP(x_i, S_i)$. After the second round of communication, if a candidate $H \in \mathcal{G}_n$ exists, each node computes $S = (S_1, \dots, S_n)$, $FP(G,S) = (FP(x_1,S_1), \dots, FP(x_n,S_n))$. If $FP(G,S) = FP(H,S)$, then nodes deduce that $G = H$. Otherwise, they deduce that $G \notin \mathcal{G}_n$ and *rejects*. Note that if G belongs to $\mathcal{G}_n$, then the algorithm always gives the correct answer. Otherwise, it rejects w.h.p. Indeed, if $G \notin \mathcal{G}_n$, then $H \neq G$, and from Lemma 1, $\Pr(FP(G,T) = FP(H,T)) \leq 1/n$. □

Algorithm 2: $\mathcal{G}$-Weak-Rec. Algorithm executed by node i.

```
1  Compute $C(x_i)$, where $C$ is the error-correcting-code with parameters $(n,n)$;
2  Communicate the element $n+j$ of $C(x_i)$ to player $j$ ;
3  Receive $C(x_1)_{n+i}, \dots, C(x_n)_{n+i}$;
4  Call $C(x_{i+n}) = (C(x_1)_{n+i}, \dots, C(x_n)_{n+i}, \vec{0})$, where $\vec{0}$ is the zero vector of $(\mathbb{F}_p)^n$;
5  Compute $p$ as the smallest prime greater than $2n \cdot |\mathcal{G}_n|^{2/n}$;
6  Compute $T$, the vector in $\mathbb{F}_p^{2n}$, given by Lemma 2 ;
7  Compute and communicate (broadcast) $FP(C(x_i), T_i)$ and $FP(C(x_{n+i}), T_{n+i})$;
8  Receive $FP(C(G), T)$;
9  Look for $H \in \mathcal{G}_n$ such that $FP(C(H), T) = FP(C(G), T)$;
10 Output $H$.
```

Our private-coin algorithm for $\mathcal{G}$-Strong-Rec has one-sided error. In fact, if the input graph belongs to $\mathcal{G}$, then our algorithm reconstructs it with probability 1. On the other hand, if G does not belong to $\mathcal{G}$, then our algorithm fails to discard the candidate with probability at most $1/n$.

Algorithm 3: Checking a candidate H. Algorithm executed by node i.

```
1 Compute $p'$, the smallest prime number such that $p' > n^2$;
2 Pick $T_i \in \mathbb{F}_p$ uniformly at random using private coins ;
3 Compute $FP(x_i, T_i)$ ;
4 Communicate $FP(x_i, T_i)$ and $T_i$ ;
5 Receive $T = (T_1, \dots, T_n)$ and $FP(G, T)$ ;
6 Output $H$ if $FP(H, T) = FP(G, T)$, otherwise reject.
```

5. Revisiting the one-round case

In this section we revisit the one-round case (and therefore the broadcast congested clique model). But, instead of studying hereditary graph classes, we study arbitrary graph classes, and we show that for this general case we need a larger bandwith. Our results are tight, not only in terms of the bandwidth, but also in the necessity of using randomness.

Theorem 3. *Let $\mathcal{G}$ be a set of graphs. The following holds:*

1) *There exists a one-round deterministic algorithm in the congested clique model that solves $\mathcal{G}$-*Weak-Rec *with bandwidth $\mathcal{O}(\sqrt{\log|\mathcal{G}_n|\log n} + \log n)$.*
2) *There exists a one-round private-coin algorithm in the congested clique model that solves $\mathcal{G}$-*Strong-Rec *with bandwidth $\mathcal{O}(\sqrt{\log|\mathcal{G}_n|\log n} + \log n)$ w.h.p.*

Proof. The algorithms in this case are very similar to the algorithms we provided in the proof of Theorem 2. Let k be a parameter whose value will be chosen at the end of the proof, and let C be the error-correcting-code with parameters (n, k). Let p be the smallest prime number greater than $2n \cdot |\mathcal{G}|^{2/k}$. Let $T \in (\mathbb{F}_p)^{n+k}$ be the vector given by Lemma 2, corresponding to $\mathcal{G}$. In the algorithm, every node i computes $C(x_i)$, and communicates $FP(C(x_i), T_i)$ together with $C(x_i)_{n+1}, \dots, C(x_i)_{n+k} \in (\mathbb{F}_q)^k$, where q is the smallest prime greater than $k + n$. Note that the communication round requires bandwidth

$$\mathcal{O}(\log p + k \cdot \log(n+k)) = \mathcal{O}(\log|\mathcal{G}_n|/k + (k+1) \cdot \log n).$$

After the communication round, every node knows $FP(C(x_i), T_i)$, for all $i \in [n]$, and also knows the matrix $A(\tilde{G})$. Therefore, every node can compute $FP(C(x_i), T_i)$, for all $i \in \{n+1, \dots, n+k\}$, and, moreover, compute $FP(C(G), T)$.

From the construction of T, there is at most one graph $H \in \mathcal{G}_n$ such that $FP(C(G), T) = FP(C(H), T)$. Therefore, if G belongs to $\mathcal{G}$, every node can reconstruct it.

On the other hand, if we are solving $\mathcal{G}$-Strong-Rec, then we proceed as in the algorithm of Theorem 2, either testing whether $H = G$ in one more round, or sending a fingerprint of G to check with high probability if a candidate $H \in \mathcal{G}_n$ such that $FP(C(G), T) = FP(C(H), T)$ is indeed equal to G. This verification requires to send $\mathcal{O}(\log n)$ more bits, which fits in the asymptotic bound of the bandwidth. The optimal value of k, that is, the one which minimizes the expression $\mathcal{O}(\log|\mathcal{G}_n|/k + (k+1) \cdot \log n)$, is such that $k = \mathcal{O}\left(\sqrt{\frac{\log|\mathcal{G}_n|}{\log n}}\right)$. Therefore, the bandwidth is $\mathcal{O}(\sqrt{\log|\mathcal{G}_n|\log n} + \log n)$. □

Remark 5. Consider the case where $\mathcal{G}$ is sparse but we want to reconstruct it using the *broadcast* congested clique model (and therefore we can not use Lenzen's algorithm). Suppose, for instance, that the number of edges of graphs in $\mathcal{G}$ is $\mathcal{O}(n)$. The naive algorithm, where every node broadcasts its incident edges, may take $\Omega(n/b)$ rounds, because some nodes may have $\Omega(n)$ neighbors (recall that b is the bandwidth). In Theorem 3 above, we proved that, in the broadcast congested clique model, we can reconstruct any class of graphs $\mathcal{G}$ in one round using bandwidth $b = \mathcal{O}(\sqrt{\log|\mathcal{G}_n|\log n} + \log n)$. The class of graphs having $\mathcal{O}(n)$ edges satisfies that $\log|\mathcal{G}_n| = \mathcal{O}(n \log n)$. Hence, we can reconstruct it in one round using bandwidth $b = \mathcal{O}(\sqrt{n}\log n)$. This algorithm is much faster than the naive one, that would take, for the same bandwidth, $\Omega(\sqrt{n}/\log n)$ rounds.

Our algorithms for solving $\mathcal{G}$-Weak-Rec and $\mathcal{G}$-Strong-Rec are tight, from two different perspectives. First, from the point of view of the bandwidth. In fact, in Theorem 4 we exhibit a class of graphs $\mathcal{G}$ satisfying $|\mathcal{G}_n| \le 2^{\mathcal{O}(n)}$ such that every algorithm (deterministic or randomized) solving $\mathcal{G}$-Weak-Rec in the broadcast congested clique model has cost $Rb = \Omega(\sqrt{\log|\mathcal{G}_n|})$. This lower bound matches the upper *one-round* bound given in Theorem 3 (up to logarithmic factors).

Second, if we want to solve $\mathcal{G}$-Strong-Rec with non-trivial, general one-round algorithms, we are forced to use randomness. In fact, we exhibit in Theorem 5 a set of graphs $\mathcal{G}^*$ satisfying $|\mathcal{G}^*_n| \le 2^n$ such that, every one-round deterministic algorithm that solves $\mathcal{G}^*$-Strong-Rec, requires bandwidth $\Omega(n)$.

Theorem 4. *There exists a class of graphs $\mathcal{G}^+$ satisfying $|\mathcal{G}^+_n| \le 2^{\mathcal{O}(n)}$ such that, any ϵ-error public-coin algorithm in the broadcast congested clique model that solves $\mathcal{G}^+$-*Weak-Rec*, has cost $Rb = \Omega(\sqrt{n}) = \Omega(\sqrt{\log|\mathcal{G}^+_n|})$.*

Proof. Let $\mathcal{G}^+$ be the class of graphs defined as follows: G belongs to $\mathcal{G}^+_n$ if and only if G is the disjoint union of a graph H of $\lceil\sqrt{n}\rceil$ nodes and $n-|H|$ isolated nodes. Note that $|\mathcal{G}^+_n| = \binom{n}{\lceil\sqrt{n}\rceil}\cdot 2^{\binom{\lceil\sqrt{n}\rceil}{2}} \leq 2^{\mathcal{O}(n)}$. Indeed, there are $2^{\binom{\lceil\sqrt{n}\rceil}{2}} = 2^{\mathcal{O}(n)}$ labeled graphs of size $\lceil\sqrt{n}\rceil$, and at most $\binom{n}{\lceil\sqrt{n}\rceil} = 2^{\mathcal{O}(\sqrt{n}\log n)}$ different labelings of a graph of $\sqrt{n}$ nodes using n labels (so $\mathcal{G}^+$ is closed under isomorphisms).

Let $\mathcal{A}$ be an ϵ-error public-coin algorithm solving $\mathcal{G}^+$-Weak-Rec in $R(n)$ rounds and bandwidth $b(n)$, on input graphs of size n.

Consider now the following algorithm $\mathcal{B}$ that solves $\mathcal{U}$-Weak-Rec, where $\mathcal{U}$ is the set of all graphs: on input graph G of size n, each node $i\in[n]$ supposes that it is contained in a graph G^+ formed by G plus n^2-n isolated vertices with identifiers $(n+1),\dots,n^2$. Note that G^+ belongs to $\mathcal{G}^+$. Then, node i simulates $\mathcal{A}$ as follows: at each round, node $i\in[n]$ produces the message of node i in G^+ according to $\mathcal{A}$. Note that the messages produced by nodes labeled $(n+1),\dots,n^2$ do not depend on G, so they can be produced by any node of G without any extra communication. Since $\mathcal{A}$ solves $\mathcal{G}^+$-Weak-Rec, when the algorithm halts every node knows all the edges of G^+, so they reconstruct G ignoring vertices labeled $(n+1),\dots,n^2$.

We deduce that algorithm $\mathcal{B}$ solves $\mathcal{U}$-Weak-Rec. Note that the cost of $\mathcal{B}$ is $R(n^2)b(n^2)$ on input graphs of size n. We deduce that $R(n^2)b(n^2)=\Omega(n)$, i.e., the cost of $\mathcal{A}$ is $\Omega(\sqrt{n})$. $\square$

We now show that when restricted to one-round algorithms, the use of randomness is necessary in order to have non-trivial general algorithms solving $\mathcal{G}$-Strong-Rec.

Definition 3. We say that an algorithm *recognizes* $\mathcal{G}$ if the algorithm decides whether an input graph G belongs to $\mathcal{G}$. We call $\mathcal{G}$-Recognition the problem of recognizing $\mathcal{G}$.

Theorem 5. *There exists a set of graphs $\mathcal{G}^*$ satisfying $|\mathcal{G}^*_n|\leq 2^n$ such that, any one-round deterministic algorithm in the congested clique model that solves $\mathcal{G}^*$-Recognition, requires bandwidth $\Omega(n)$.*

Proof. We prove this theorem by a counting argument. Our goal is to show that there are more *small* sets of graphs than one-round deterministic algorithms capable to recognize them. We first count the number of sets of graphs (not necessarily closed under taking isomorphism) containing 2^n different graphs of size n. We call the family of these sets $\mathcal{C}$. There are $2^{\binom{n}{2}}$ possible graphs of size n, so $\binom{2^{\binom{n}{2}}}{2^n}$ possible choices for graphs in $\mathcal{C}$. We deduce that there exists $c_1>0$ such that $|\mathcal{C}|\geq 2^{c_1\cdot n^2\cdot 2^n}$.

On the other hand, we count the number of one-round deterministic algorithms that recognize a set of graphs in $\mathcal{C}$ with bandwidth at most β. A one-round deterministic algorithm is composed of two parts: the algorithm before the communication round, and the algorithm after the communication. The first part of the algorithm is defined by the messages that a node sends on each input. The input of a node is its neighborhood represented by a Boolean vector of size n, and an integer representing its label. Therefore, the first part of the algorithm is defined by the messages corresponding to all the $n2^n$ possible inputs. Since the bandwidth is β, we obtain that there are $2^{n\beta 2^n}$ possible choices for the first part of the algorithm.

The second part of the algorithm is defined by a function $f_{\mathcal{G}}:(\{0,1\}^\beta)^n\to\{0,1\}$, such that if $m=(m_1,\dots,m_n)$ are the messages sent by the nodes in the communication round, then $f(m)=1$ if and only if m was produced from an input graph belonging to $\mathcal{G}$. The crucial observation is that this implies that f can output 1 in at most 2^n inputs. Therefore, the number of possible second parts of the algorithm is $\sum_{i\in[2^n]}\binom{2^{n\beta}}{i}\leq(1+2^{n\beta})^{2^n}\leq 2^{c_2\cdot n\beta 2^n}$, where $c_2>0$ is a constant.

We deduce that the number of one-round deterministic algorithms with bandwidth β that are capable to recognize a set of graphs in $\mathcal{C}$ is at most $2^{c_3 n\beta 2^n}$, with $c_3>0$. Since we are considering only deterministic algorithms, two different sets must be recognized by two different algorithms. This implies that $2^{c_3 n\beta 2^n}$ must be greater than $2^{c_1 n^2 2^n}$, so $\beta=\Omega(n)$. Finally, we construct the set $\mathcal{G}^*$ by picking, for each n, one set of graphs contained in $\mathcal{C}$ that can not be recognized by any algorithm of bandwidth $o(n)$. $\square$

Remark 6. Note that for any set of graphs $\mathcal{G}$, problem $\mathcal{G}$-Strong-Rec is at least as hard as $\mathcal{G}$-Recognition. We conclude that there exists a set of graphs $\mathcal{G}^*$ satisfying $|\mathcal{G}^*_n|\leq 2^n$ such that, any one-round deterministic algorithm that solves $\mathcal{G}^*$-Strong-Rec, requires bandwidth $\Omega(n)$. Note that, from Theorem 3, we know that $\mathcal{G}^*$-Strong-Rec can be solved using a one-round private-coin algorithm with bandwidth $\mathcal{O}(\sqrt{n\log n})$ w.h.p.

Unfortunately, we do not know any explicit set of graphs $\mathcal{G}$ satisfying the above remark.

6. Local time complexity

Even though in the congested clique model, by definition, the only complexity measure taken into account is communication, it is important to point out that the general algorithms we presented in this paper might run in exponential local time. The reason behind this is that nodes, at some point, need to look, in $\mathcal{G}_n$, for a graph having the same fingerprint as the fingerprint of G. The search space is obviously super-polynomial when $|\mathcal{G}_n|$ is super-polynomial (see, for instance, line 6 of Algorithm 1).

In this section we initiate the study of the following question: for what graph classes $\mathcal{G}$ can the reconstruction problem $\mathcal{G}$-Strong-Rec be solved in one round, bandwidth $\mathcal{O}(\log|\mathcal{G}_n|/n)$ and *polynomial* local time? We already know that this can be achieved for d-degenerate graphs (deterministically) and for cographs (with public coins). Using ideas from the cograph reconstruction algorithm [21] we devise here one-round public-coin algorithms for reconstructing *distance-hereditary graphs* and *graphs of bounded modular width*. Both algorithms use bandwidth $\mathcal{O}(\log n)$, run in polynomial local time, and give the correct answer w.h.p. (Distance-hereditary graphs and graphs of bounded modular width are well studied hereditary graph classes in the context of sequential algorithms [8,9].)

6.1. Distance hereditary graphs

A graph $G=(V,E)$ is distance-hereditary if the distance between any two nodes of a same connected component is preserved in any induced subgraph that contains them. Distance-hereditary graphs can be characterized by the existence of a *twin-pendant vertex decomposition*. To define this decomposition, we must introduce some concepts. For more information about distance-hereditary graphs, we recommend the book on graph classes written by Brandstädt, Le and Spinrad [4].

Let G be a graph. A vertex v in G is called *pendant* if v has only one neighbor. Two vertices u, v in G are called *true twins* if u and v are adjacent, and share the same neighborhood, i.e., $N(u)=N(v)$. Similarly, u and v are called *false twins* when u and v are non-adjacent and share the same neighborhood.

A *twin-pendant vertex decomposition* of an n-node graph G is a sequence $(v_1,\dots,v_n)$ of vertices of G, such that, for each $i\in\{1,\dots,n\}$, one of the following conditions is true:

1. v_i is a pendant vertex in $G[\{v_{i+1},\dots,v_n\}]$,
2. v_i has a true-twin in $G[\{v_{i+1},\dots,v_n\}]$,
3. v_i has a false-twin in $G[\{v_{i+1},\dots,v_n\}]$.

Theorem 6 *([1]). A graph G is distance-hereditary if and only if G has a twin-pendant vertex decomposition.*

We use previous proposition to devise a one-round, public-coin algorithm to reconstruct the class of distance-hereditary graphs.

Definition 4. Let p be a prime number, and let $G=(V,E)$ be a graph. A vector $m=((a_v,b_v))_{v\in V}\in\mathbb{F}_p^{2n}$ is *valid* for G at $t\in\mathbb{F}_p$ if there is a linearly independent family of polynomials $\Phi=(\phi_v)_{v\in V}$ in $\mathbb{F}_p[X]$ such that, for each $v\in V$,

$$a_v=\phi_v(t) \mod p \text{ and } b_v=\sum_{w\in N_G(v)}\phi_w(t) \mod p.$$

Note that, if node w in G has degree 1, and its neighbor is node u, then $b_w=a_u$. Moreover, two twin nodes u,w in G satisfy either $b_u=b_w$ (if they are non-adjacent) or $b_u+a_u=b_w+a_w$ (if they are adjacent). In the following two lemmas, we show how to compute, from a valid vector of a graph G, a valid vector of the graph obtained by the deletion of some specific nodes.

Lemma 3. *Let $m=((a_v,b_v))_{v\in V}\in(\mathbb{F}_p)^{2n}$ be valid for $G=(V,E)$ at t. Let w be a node of degree 1 in G, and let u be its unique neighbor in G. Then, the vector $m'=((a'_v,b'_v))_{v\in V\setminus\{w\}}\in(\mathbb{F}_p)^{2n-2}$ is valid for $G-w$ at t, where*

$$a'_v=a_v \text{ for all } v\in V\backslash\{w\},\qquad b'_v=\begin{cases} b_v & \text{if } v\neq u\\ b_u-a_w & \text{if } v=u\end{cases}$$

Proof. Let $\Phi=(\phi_v)_{v\in V}$ be a linearly independent family of polynomials associated to m. We simply note that m' is a valid vector of $G-w$ in t for the family of polynomials $\Phi'=(\phi'_v)_{v\in V\setminus\{w\}}$ such that $\phi'_v=\phi_v$ for all $v\in V\setminus\{w\}$. Indeed, for all $v\in V\setminus\{w\}$, $a'_v=\phi'_v(t)=\phi_v(t)=a_v$. On the other hand, for all $v\neq u,w$,

$$b'_v=\sum_{x\in N_G(v)\setminus\{w\}}\phi'_x(t)=\sum_{x\in N_G(v)\setminus\{w\}}\phi_x(t)=b_v.$$

Finally, $b'_u=\sum_{x\in N_G(v)\setminus\{w\}}\phi'_x(t)=\sum_{x\in N_G(v)}\phi_x(t)-\phi_w(t)=b_u-a_w$. □

Lemma 4. *Let $m=((a_v,b_v))_{v\in V}\in(\mathbb{F}_p)^{2n}$ be valid for $G=(V,E)$ at $t\in\mathbb{F}_p$. Let u,w be (true or false) twins in G such that $a_u\neq a_w$. Then, a vector*

$$m'=((a'_v,b'_v))_{v\in V\setminus\{w\}}\in(\mathbb{F}_p)^{2n-2}$$

is valid for $G-w$ at t, where

$$a'_v = \begin{cases} a_v & \text{if } v \neq u \\ a_u + a_w & \text{if } v = u \end{cases} \qquad b'_v = \begin{cases} b_v & \text{if } v \neq u \\ b_u - a_w\delta_{uw} & \text{if } v = u \end{cases}$$

and $\delta_{uw} = 1$ if $a_u + b_u = a_w + b_w$ and 0 otherwise.

Proof. Let $\Phi = (\phi_v)_{v\in V}$ be a linearly independent family of polynomials associated to m.

Consider the family of polynomials $\Phi' = (\phi'_v)_{v\in V\setminus\{w\}}$, given by $\phi'_v = \phi_v$ for each $v \neq u$ and $\phi'_u = \phi_u + \phi_w$, which is trivially linearly independent because Φ is. We will show that $m' = ((a'_v, b'_v))_{v\in V\setminus\{w\}} \in (\mathbb{F}_p)^{2n-2}$ is valid for $G - w$ at t for the family of polynomials Φ', i.e., $a'_v = \phi'_v(t)$ and $b'_v = \sum_{x\in N_G(v)\setminus\{w\}} \phi'_x(t)$. First, if $v \neq u$ then $a'_v = \phi'_v(t) = \phi_v(t) = a_v$. On the other hand, $a'_u = \phi'_u(t) = \phi_u(t) + \phi_w(t) = a_u + a_w$.

Now fix $v \neq u$, and note that either both u and w are neighbors of v, or none of them, because u and w are twins. If u and w are neighbors of v, then,

$$\begin{aligned} b'_v &= \textstyle\sum_{x\in N_G(v)\setminus\{w\}} \phi'_x(t) \\ &= \phi'_u(t) + \textstyle\sum_{x\in N_G(v)\setminus\{u,w\}} \phi'_x(t) \\ &= \phi_u(t) + \phi_w(t) + \textstyle\sum_{x\in N_G(v)\setminus\{u,w\}} \phi_x(t) \\ &= b_v \end{aligned}$$

On the other hand, if u and w are not neighbors of v, then

$$b'_v = \sum_{x\in N_G(v)\setminus\{w\}} \phi'_x(t) = \sum_{x\in N_G(v)\setminus\{u,w\}} \phi'_x(t) = \sum_{x\in N_G(v)\setminus\{u,w\}} \phi_x(t) = b_v.$$

Finally, remember that u and w are adjacent if and only if $a_u + b_u = a_w + b_w$. Then, δ_{uw} equals 1 when u and w are adjacent, and 0 otherwise. Therefore

$$\begin{aligned} b'_u &= \textstyle\sum_{x\in N_G(v)\setminus\{w\}} \phi'_x(t) \\ &= \textstyle\sum_{x\in N_G(v)\setminus\{w\}} \phi_x(t) \\ &= \textstyle\sum_{x\in N_G(v)} \phi_x(t) - \delta_{uw}\phi_w(t) \\ &= b_u - \delta_{uw}a_w. \quad \square \end{aligned}$$

Our distance-hereditary reconstruction algorithm consists of two parts: first, using the public random bits, the nodes jointly produce a valid vector for the input graph, which is communicated to all nodes. Then, each node uses the information in the valid vector to produce a decomposition (this part does not require any extra communication). Each vertex will look for either a pendant node w or a pair of twins u, w, and then delete node w from the graph updating the valid vector according to Lemma 3 and Lemma 4. The process is iterated until the whole graph is deleted (if the graph is distance-hereditary), or the process is blocked (so the graph is not distance-hereditary).

Theorem 7. *Let $\mathcal{G}$ be the class of distance-hereditary graphs. There is a one-round, public-coin algorithm in the congested clique model that solves problem $\mathcal{G}$-Strong-Rec w.h.p. using bandwidth $\mathcal{O}(\log n)$ and $\mathcal{O}(n^3)$ local-time computation.*

Proof. Let $G = (V, E)$ the input graph of size n. For simplicity we assume that $V = \{1, \dots, n\}$. In other words, the nodes and their identities are the same. Let p be a prime number to be fixed later and $\Phi = (x^v)_{v\in V}$ as a family of polynomials in $\mathbb{F}_p[X]$. Before the communication round of the algorithm, nodes pick $t \in \mathbb{F}_p$ uniformly at random using the public randomness, and communicate m_v, where $m = (m_v)_{v\in V}$ is valid for G at t for the family of polynomials $\Phi = (x^v)_{v\in V}$.

Upon receiving all messages, every vertex executes an iterative algorithm for $i \in \{1, \dots, n\}$. In each step i it receives as input a vector m^i and a set V^i and outputs a vector m^{i+1} of and a set V^{i+1}, where $V^1 = V$ and $m^1 = m$. We call $m^i_v = (a^i_v, b^i_v)$, for each $i \in \{1, \dots n\}$. The vector m^{i+1} and the set V^{i+1} is generated as follows

1. (*Look for a pendant vertex*) Look for a pair of vertices u, w in V^i satisfying $a^i_u = b^i_w$. If such pair exists, according to Lemma 3, we deduce that w has degree 1 and his unique neighbor is u. Then m^{i+1} is computed according to Lemma 3, and V^{i+1} is defined as $V^i - \{w\}$. Then, the algorithm starts step $i + 1$. If no such pair exists, continue to step 2.
2. (*Look for twins*) Look for a pair of vertices u, w in V^i such that $a_u \neq a_w$ and either $b_u = b_w$ or $a_u + b_u = a_w + b_w$. Is such pair exists, we deduce that u and w are (true or false) twins. In either case, m^{i+1} is computed according to Lemma 4, and V^{i+1} is defined as $V^i - \{w\}$. Then, the algorithm starts step $i + 1$. If no such pair exists, continue to step 3.
3. Deduce that G is not distance hereditary and all nodes *reject*.

We conclude by proving the correctness of the algorithm. Let ϕ^i be the linearly independent family of polynomials corresponding to the valid vector of $G^i = G[V^i]$. For each $u, v \in V(G^i)$, $u \neq v$, consider the polynomials $\alpha^i_{u,v} = \phi^i_u - \phi^i_v$, $\beta^i_{u,v} = \sum_{w\in N_{G^i}(u)} \phi^i_w - \sum_{w\in N_{G^i}(v)} \phi^i_w$, $\gamma^i_{u,v} = \sum_{w\in N_{G^i}[u]} \phi^i_w - \sum_{w\in N_{G^i}[v]} \phi^i_w$ and $\sigma^i_{u,v} = \phi^i_u - \sum_{w\in N_{G^i}(v)} \phi^i_w$.

The algorithm errs when a pair of nodes u, w satisfies the conditions 1 or 2 (without being twins or without one pending from the other). The probability of this event is at most the probability that, at some step i, t is a root of a nonzero polynomial $\alpha^i_{u,v}, \beta^i_{u,v}, \gamma^i_{u,v}, \sigma^i_{u,v}$ for some pair u, v in any iteration. The union of the families $\alpha^i = (\alpha^i_{u,v})_{u,v\in V}$, $\beta^i = (\beta^i_{u,v})_{u,v\in V}$, $\gamma^i = (\gamma^i_{u,v})_{u,v\in V}$ and $\sigma^i = (\sigma^i_{u,v})_{u,v\in V}$, have at most $4n^2$ polynomials, there are at most n steps and each polynomial has degree at most n. This implies that the protocol fails with probability at most $4n^4/p$. If we pick $p \in [4n^{c+4}, 4n^{c+5}]$ for $c > 1$ we obtain that the protocol fails with probability at most $1/n^c$. □

6.2. *k-Modular-width graphs*

Graphs of bounded modular-width were introduced by Gajarský et al. [10] in the context of parameterized graph algorithms and parameterized complexity, with the aim of solving efficiently a large family of problems when this parameter is fixed. The modular-width generalizes other natural parameters such as *vertex cover*, *neighborhood diversity* and *twin-cover*. We refer the reader to [28] and [15] for surveys on structural and algorithmic aspects of modular decompositions (in the first reference, they are called *substitution decompositions*).

To define k-modular-width graphs we must define the notion of *module*. A module M in a graph G is a set of vertices such that all members in M have the same neighborhood outside M. Formally, for all $u, v \in M$: $N(u) \setminus M = N(v) \setminus M$. We say that a graph G is a *k-modular-width graph* if one of the following conditions holds:

1. G has at most one node (the base case).
2. G is a disjoint union of two k-modular-width graphs.
3. G is a *join* of two k-modular-width graphs, i.e., G is obtained from two disjoint k-modular-width graphs by taking their disjoint union and then adding all possible edges between these two graphs.
4. The node set of G can be partitioned into $\ell \le k$ modules $V_1, \dots, V_\ell$ such that $G[V_\ell]$ is a k-modular-width graph, for all $1 \le i \le \ell$.

We say that a class of graphs $\mathcal{G}$ is of *bounded modular width* if there exists some fixed $k > 0$ such that every $G \in \mathcal{G}$ is a k-modular-width graph.

Given a graph G of modular-width at most k, one can associate to G a rooted decomposition tree T following the rules above. Each internal node of the tree corresponds to operations 2 to 4, and leaves correspond to single vertices (operation 1). The following observations are crucial for our reconstruction algorithm.

Lemma 5 *([9]). Let G be a k-modular-width graph having strictly more than k vertices. Then, G has a module of* $2 \le \ell \le k$ *vertices.*

Proof. Consider a decomposition tree T of G. Let u be one of the internal nodes such that all sons of u are leaves. Then, the corresponding module $V[u]$ is formed by at most k leaves of the subtree rooted in u, and satisfies the lemma. □

Lemma 6. *Let G be a graph and $M = \{u_1, \dots, u_l\}$ be a module of G, with $2 \le l \le k$. G is a k-modular-width graph if and only if $G[u_1 \leftarrow M] := G - \{u_2, \dots, u_l\}$ is a k-modular-width graph.*

Proof. Let $G = (V, E)$ be a k-modular-width graph. Actually, any induced subgraph $G[W]$ is of modular width at most k. Indeed, consider a decomposition tree for G, remove all leaves contained in $V \setminus W$, and shortcut all nodes with at most one son. We obtain a decomposition tree of $G[W]$, proving that it is of modular width at most k.

Conversely, assume that $\{u_1, \dots, u_l\}$ is a module of G, and $G[u_1 \leftarrow M]$ is a k-modular-width graph. Replace, in a decomposition tree of $G[u_1 \leftarrow M]$ the leaf u_1 with a decomposition tree of $G[\{u_1, \dots, u_l\}]$. We obtain a decomposition tree of G, certifying that G is a k-modular-width graph. □

The idea behind our reconstruction algorithm for k-modular-width graphs consists in iteratively find a module of size at most k in the graph, then *compress* the module into a single node, and repeat the procedure in the resulting graph. The procedure stops when the initial graph is reduced to a single vertex (and by Lemmas 5 and 6 we deduce that the graph is a k-modular-with graph), or no module can be found (and we deduce that the graph was not a k-modular-width graph). Following lemmas show how valid vectors can be used to identify the modules on this iterative algorithm.

Lemma 7. *Let $m = ((a_v, b_v))_{v\in V} \in (\mathbb{F}_p)^{2n}$ be a valid vector for $G = (V, E)$ at $t \in \mathbb{F}_p$. Let $M = \{u_1, \dots, u_k\}$ be a module in G of size k, and call $H = G[M]$ the subgraph induced by M. The vector*

$$m[u_1 \leftarrow M] = ((a'_v, b'_v))_{v\in V\setminus\{u_2,\dots,u_k\}} \in (\mathbb{F}_p)^{2n-2k}$$

defined by:

$$a'_v = \begin{cases} a_v & \text{if } v \neq u \\ \sum_{i=1}^k a_{u_k} & \text{if } v = u_1 \end{cases} \qquad b'_v = \begin{cases} b_v & \text{if } v \neq u \\ b_{u_1} - \sum_{u\in N_H(u_1)} a_u & \text{if } v = u_1 \end{cases}$$

is valid for $G[u_1 \leftarrow M]$ at t.

Proof. Let $\Phi = (\phi_v)_{v \in V}$ be a linearly independent family of polynomials associated to m, and let us call $G' = G[u_1 \leftarrow M]$ and $m' = m[u_1 \leftarrow M]$.

We show that m' is valid for G' at t for the family of polynomials $\Phi' = (\phi'_v)_{v \in V \setminus \{u_2,\dots,u_n\}}$ given by $\phi'_v = \phi_v$ for each $v \neq u_1$ and $\phi'_{u_1} = \sum_{i=1}^{k} \phi_{u_i}$, which is trivially linearly independent. For $w \neq u_1$, we have that

$$a'_w = a_w = \phi_w(t) = \phi'_w(t).$$

Since M is a module in G, any node $w \in V \setminus M$ is either adjacent to $\{u_1, \dots, u_k\}$ or to none of them. In both cases

$$b'_w = b_w = \sum_{v \in N(w)} \phi'_v(t).$$

With respect to u_1, by definition,

$$a'_{u_1} = \phi'_{u_1}(t) = \sum_{i=1}^{k} \phi_{u_i} = \sum_{i=1}^{k} a_{u_i}.$$

Finally,

$$b'_{u_1} = \sum_{u \in N_{G'}(u_1)} \phi'_u = \sum_{u \in N_G(u_1) \setminus M} \phi_u = b_{u_1} - \sum_{u \in N_H(u_1)} a_u. \quad \square$$

The following lemma shows how to use a valid vector of G to find a module. Let $G = (V, E)$ be a graph, let $m = ((a_v, b_v))_{v \in V}$ be a valid vector of G at $t \in \mathbb{F}_p$, and let $0 < k \leq n$. Let $M = \{u_1, \dots, u_k\}$ be a set of nodes and let H be k-node graph such that $V(H) = M$. Let us call, for $u \in M$,

$$s(u, m, H) = b_u - \sum_{w \in N_H(u)} a_w$$

Lemma 8. *If M is a module of G such that $G[M] = H$ then*

$$s(u, m, H) = s(v, m, H) \text{ for all } u, v \in M.$$

Proof. Let M be a module of G. Then, $N_G(u) \setminus M = N_G(v) \setminus M$, for all $u, v \in M$. Note that $s(u, m, H)$ is exactly $\sum_{w \in N_G(u) \setminus M} a_w$. We deduce that $s(u, m, H) = s(v, m, H)$ for all $u, v \in M$. $\square$

The algorithm for reconstructing k-modular-width graphs is very similar to the graph reconstructing distance-hereditary graphs. Using the public random bits the nodes will jointly produce a valid vector for the input graph, which is communicated to all nodes. Then, each node will use the information in the valid vector to produce a modular decomposition. Each node *guesses* a non-trivial module by testing every set M of at most k nodes, and for each such set, every possible induced graph H of $|M|$ vertices. Then, it suppresses all vertices of the module except one, as in Lemma 7, and repeats the procedure in the remaining graph.

Theorem 8. *Let $\mathcal{G}$ be the class of k-modular-width graphs. There is a one-round, public-coin algorithm in the congested clique model that solves problem $\mathcal{G}$-STRONG-REC w.h.p. using bandwidth $\mathcal{O}(\log n)$ and $\mathcal{O}(k^2 2^{k^2} n^{k+4})$ local-time computation.*

Proof. Let $G = (V, E)$ the input graph of size n. Let p be a prime number to be fixed later and $\Phi = (x^v)_{v \in V}$ be a family of polynomials in $\mathbb{F}_p[X]$. Before the communication round of the algorithm, nodes pick $t \in \mathbb{F}_p$ uniformly at random using the public randomness, and communicate m_v, where $m = (m_v)_{v \in V}$ is valid for G at t for the family of polynomials $\Phi = (x^v)_{v \in V}$.

Upon receiving all messages, every vertex executes an iterative algorithm (the algorithm is detailed in Algorithm 4). The algorithm starts defining a set $E^{out} = \emptyset$ which at the end contains the set of edges of the input graph, when the input graph is a k-modular-width graph. In step $i \in \{1, \dots, n\}$ the algorithm receives as input a vector m^i, a set V^i and a map $\varphi^i : V \to V$, and outputs a vector m^{i+1}, a set V^{i+1} and map φ^{i+1}. Initially $m^1 = m$, $V^1 = V$ and φ^1 is the identity, and for $i > 1$:

- V^i is the set of vertices obtained after the i-th time a module of size at most k is identified and reduced to a single vertex (see Lemma 8).
- m^i is the valid vector of $G[V^i]$ in t.
- φ^i maps each vertex v into the vertex u into which v is compressed when a module is reduced to u. In other words, $\varphi^i(v) = u$ then u is an ancestor of v in the decomposition tree T.

On input (m^i, V^i, φ^i), the values of m^{i+1}, V^{i+1} and φ^i are computed as follows:

1. Look for a set of at most k nodes that induce module M. To do this, the algorithm picks every possible set $M = \{u_1, \dots, u_\ell\} \subseteq V^i$ of $2 \le \ell \le k$ nodes and every possible subgraph $H = (M, E')$, and test whether $s(u, m^i, H) = s(u_v, m^i, H)$ for all $u, v \in M$. If one such a set M and graph H are found, the algorithm deduces that M is a module and H is the subgraph of $G[V^i]$ induced by M, and continues with step 2. Otherwise, the algorithm deduces that the input graph is not a k-modular-width graph and rejects. Otherwise it continues to step 2.

2. When a module is identified, the algorithm deduces that the connections between the vertices in the module are given by H. Let v_1 and v_2 be two vertices in V such that $\varphi^i(v_1), \varphi^i(v_2) \in M$. Suppose now that $\{\varphi^i(v_1), \varphi^i(v_2)\} \in E(H)$. The fact that v_1 and v_2 have adjacent ancestors in the decomposition tree means that v_1 and v_2 are also adjacent. Therefore, the algorithm adds to E^{out} all edges $\{v, w\}$ satisfying that $\varphi^i(v), \varphi^i(w) \in M$ and $\{\varphi^i(v), \varphi^i(w)\} \in E(H)$.

3. Finally, the algorithm defines $V^{i+1} = V^i - \{u_2, \dots, u_\ell\}$. If $V^{i+1} = \{u_1\}$, return the set of edges E^{out}. Otherwise, the algorithm computes then the vector $m^{i+1} = m^i[u_1 \leftarrow M]$ according to Lemma 7. The algorithm has all the information to compute m^{i+1} because it knows H. Then, for each vertex $v \in V$ it defines $\varphi^{i+1}(v) = u_1$ if $\varphi^{i+1}(v) \in M$, and $\varphi^{i+1}(v) = \varphi^i(v)$ otherwise.

Algorithm 4: Iterative algorithm executed by every vertex after the communication round.

```
1  Receive from each $v \in V$ the bits of $m_v$, and define $m = (m_v)_{v \in V}$;
2  Initialize $E^{out} = \emptyset$, $V^1 = V$, $m^1 = m$ and $\varphi^1(v) = v$ for all $v \in V$;
3  for $i = 1$ to $n$ do
4      Compute the set $\Sigma$ of all tuples $(\ell, M = \{u_1, \dots, u_\ell\}, H)$ such that $2 \le \ell \le k$, $M \subseteq V^i$, $H$ is a graph with vertex set $M$, and
         $s(u, m^i, H) = s(v, m^i, H)$ for all $u, v \in M$ (see Lemma 8);
5      if $\Sigma = \emptyset$ then
6          return reject;
7      else
8          Pick any $(\ell, M = \{u_1, \dots, u_\ell\}, H) \in \Sigma$ ;
9          forall $v, w \in V$ do
10             if $\varphi^i(v) \neq \varphi^i(w)$ and $\{\varphi^i(v), \varphi^i(w)\} \in E(H)$ then
11                 $E^{out} = E^{out} \cup \{v, w\}$
12         forall $v \in V$ do
13             if $\varphi(v) \in M$ then
14                 $\varphi^{i+1}(v) = \varphi^i(u_1)$
15             else
16                 $\varphi^{i+1}(v) = \varphi^i(v)$
17         $m^{i+1} = m^i[u_1 \leftarrow M]$ (see Lemma 7) ;
18         $V^{i+1} = V^i \setminus \{u_2, \dots, u_\ell\}$;
19         if $V^i = \{u_1\}$ then
20             return $E^{out}$ ;
```

Lemmas 5, 6, 7 and 8 ensure that the Algorithm 4 is correct as long as that at each iteration a module M and the corresponding induced subgraph H are correctly identified.

Consider the i-th iteration of the algorithm, and let ϕ^i be the linearly independent family of polynomials corresponding to the valid vector of the graph $G^i = G[V^i]$ at t. For each $u, v \in V(G^i)$ and ℓ-vertex graph H with $V(H) \subseteq V(G^i)$, let us define the polynomials $\rho^i_{H,v} = \sum_{w \in N_{G^i}(v)} \phi^i_w - \sum_{w \in N_H(v)} \phi^i_w$, and $\tau^i_{H,u,v} = \rho^i_{H,u} - \rho^i_{H,v}$. Observe that $\rho^i_{H,v}(t) = s(v, m^i, H)$.

The algorithm errs if at some iteration i there is a set of at most k nodes M such that $\tau^i_{H,u,v} = 0$ for each pair of vertices $u, v \in M$, but at the same time M is not a module of $G[V^i]$ or H is not the subgraph induced by M in $G[V^i]$. Let us suppose that M is not a module of $G[V^i]$ or H is not the subgraph induced by M in $G[V^i]$. Then there must exist a pair of vertices $u, v \in M$ such that $\rho^i_{H,v}$ and $\rho^i_{H,u}$ are defined by different linear combinations of polynomials in Φ^i. Since Φ^i is a linearly independent family of polynomials, $\tau^i_{H,u,v}$ is a non-zero polynomial of degree at most n. Therefore, the probability that the algorithm fails for M and H is at most n/p.

We deduce that the probability that the algorithm misidentifies a module is at most the probability that, at any iteration, t is a root of a nonzero polynomial $\tau^i_{H,u,v}$ for some choice of set M of at most k nodes, graph $H(M, E')$, and nodes $u, v \in M$. The union of these families has at most $k^2 \cdot 2^{\mathcal{O}(k^2)} \cdot n^{k+2}$ polynomials at any iteration. Since there are at most n iterations, we conclude that the protocol fails with probability at most $n^{f(k)}(n/p)$, with $f(k) = \mathcal{O}(k^2)$. Picking $p \in [n^{f(k)+c}, n^{f(k)+c+1}]$ we obtain that the protocol fails with probability at most $1/n^c$. $\square$

7. Discussion

A natural question is to try to understand the relation between the recognition problem and the reconstruction problem. (The recognition problem is the classical decision problem, where we simply want to decide whether the input graph belongs to some class $\mathcal{G}$.) It is clear that finding a formal proof showing some type of equivalence between the reconstruction and the recognition problems would yield a non-trivial lower bound on the recognition problem. However, in [7], the authors show that any non-trivial unconditional lower bound on a decision problem in the congested clique model would imply novel Boolean circuit complexity lower bounds. Nevertheless, proving lower bounds for explicit Boolean functions in the theory of circuit complexity has been an elusive goal for decades. Therefore, even though for some graph classes $\mathcal{G}$, *it seems that the only strategy to decide whether $G \in \mathcal{G}$ is to reconstruct G*, proving this is as difficult as proving fundamental conjectures in circuit complexity, a notoriously difficult challenge.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work has been partially supported by CONICYT via PIA/Apoyo a Centros Científicos y Tecnológicos de Excelencia AFB 170001 (P.M. and I.R.), FONDECYT 1170021 (I.R.), FONDECYT 11190482 (P.M.) and PAI + Convocatoria Nacional Subvención a la Incorporación en la Academia Año 2017 + PAI77170068 (P.M.).

References

[1] Hans-Jürgen Bandelt, Henry Martyn Mulder, Distance-hereditary graphs, J. Comb. Theory, Ser. B 41 (2) (1986) 182–208.
[2] Paul Beame, Paraschos Koutris, Dan Suciu, Communication steps for parallel query processing, J. ACM 64 (6) (2017) 40.
[3] Florent Becker, Martín Matamala, Nicolas Nisse, Ivan Rapaport, Karol Suchan, Ioan Todinca, Adding a referee to an interconnection network: what can(not) be computed in one round, in: 25th IEEE International Parallel and Distributed Processing Symposium, IPDPS, 2011, pp. 508–514.
[4] Andreas Brandstädt, Van Bang Le, Jeremy P. Spinrad, Graph Classes: A Survey, SIAM, 1999.
[5] Keren Censor-Hillel, Petteri Kaski, Janne H. Korhonen, Christoph Lenzen, Ami Paz, Jukka Suomela, Algebraic methods in the congested clique, in: ACM Symposium on Principles of Distributed Computing, PODC, 2015, pp. 143–152.
[6] Jeffrey Dean, Sanjay Ghemawat, Mapreduce: simplified data processing on large clusters, Commun. ACM 51 (1) (2008) 107–113.
[7] Andrew Drucker, Fabian Kuhn, Rotem Oshman, On the power of the congested clique model, in: ACM Symposium on Principles of Distributed Computing, PODC, 2014, pp. 367–376.
[8] Eduard Eiben, Robert Ganian, O-joung Kwon, A single-exponential fixed-parameter algorithm for distance-hereditary vertex deletion, J. Comput. Syst. Sci. 97 (2018) 121–146.
[9] Fedor V. Fomin, Mathieu Liedloff, Pedro Montealegre, Ioan Todinca, Algorithms parameterized by vertex cover and modular width, through potential maximal cliques, Algorithmica 80 (4) (2018) 1146–1169.
[10] Jakub Gajarský, Michael Lampis, Sebastian Ordyniak, Parameterized algorithms for modular-width, in: Gregory Gutin, Stefan Szeider (Eds.), Parameterized and Exact Computation, Springer International Publishing, Cham, 2013, pp. 163–176.
[11] Michael R. Garey, David S. Johnson, Computers and Intractability, vol. 29, WH Freeman, New York, 2002.
[12] Mohsen Ghaffari, An improved distributed algorithm for maximal independent set, in: 23rd Annual ACM-SIAM Symposium on Discrete Algorithms, SODA, Society for Industrial and Applied Mathematics, 2016, pp. 270–277.
[13] Mohsen Ghaffari, Distributed mis via all-to-all communication, in: Proceedings of the ACM Symposium on Principles of Distributed Computing, ACM, 2017, pp. 141–149.
[14] Mohsen Ghaffari, Merav Parter, MST in log-star rounds of congested clique, in: ACM Symposium on Principles of Distributed Computing, PODC, ACM, 2016, pp. 19–28.
[15] Michel Habib, Christophe Paul, A survey of the algorithmic aspects of modular decomposition, Comput. Sci. Rev. 4 (1) (2010) 41–59.
[16] James W. Hegeman, Gopal Pandurangan, Sriram V. Pemmaraju, Vivek B. Sardeshmukh, Michele Scquizzato, Toward optimal bounds in the congested clique: graph connectivity and MST, in: ACM Symposium on Principles of Distributed Computing, PODC, ACM, 2015, pp. 91–100.
[17] James W. Hegeman, Siriam V. Pemmaraju, Lessons from the congested clique applied to MapReduce, in: Int. Colloquium on Structural Information and Communication Complexity, SIROCCO, in: LNCS, vol. 8576, 2014, pp. 149–164.
[18] James W. Hegeman, Sriram V. Pemmaraju, Vivek Sardeshmukh, Near-constant-time distributed algorithms on a congested clique, in: 28th International Symposium on Distributed Computing, DISC, 2014, pp. 514–530.
[19] Michael Isard, Mihai Budiu, Yuan Yu, Andrew Birrell, Dennis Fetterly, Dryad: distributed data-parallel programs from sequential building blocks, in: Proceedings of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems 2007, 2007, pp. 59–72.
[20] Tomasz Jurdziński, Krzysztof Nowicki, MST in O(1) rounds of congested clique, in: Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SIAM, 2018, pp. 2620–2632.
[21] Jarkko Kari, Martín Matamala, Ivan Rapaport, Ville Salo, Solving the induced subgraph problem in the randomized multiparty simultaneous messages model, in: International Colloquium on Structural Information and Communication Complexity, Springer, 2015, pp. 370–384.
[22] Howard Karloff, Siddharth Suri, Sergei Vassilvitskii, A model of computation for mapreduce, in: Proceedings of the 21st ACM-SIAM Symposium on Discrete Algorithms, Society for Industrial and Applied Mathematics, 2010, pp. 938–948.
[23] Hartmut Klauck, Danupon Nanongkai, Gopal Pandurangan, Peter Robinson, Distributed computation of large-scale graph problems, in: 26th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA, SIAM, 2015, pp. 391–410.
[24] Christoph Lenzen, Optimal deterministic routing and sorting on the congested clique, in: ACM Symposium on Principles of Distributed Computing, PODC, 2013, pp. 42–50.
[25] Rudolf Lidl, Harald Niederreiter, Introduction to Finite Fields and Their Applications, Cambridge University Press, 1994.
[26] Z. Lotker, B. Patt-Shamir, E. Pavlov, D. Peleg, Minimum-weight spanning tree construction in O (log log n) communication rounds, SIAM J. Comput. 35 (1) (2005) 120–131.
[27] Grzegorz Malewicz, Matthew H. Austern, Aart J.C. Bik, James C. Dehnert, Ilan Horn, Naty Leiser, Grzegorz Czajkowski, Pregel: a system for large-scale graph processing, in: Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data, ACM, 2010, pp. 135–146.
[28] Rolf H. Möhring, Franz J. Radermacher, Substitution decomposition for discrete structures and connections with combinatorial optimization, in: North-Holland Mathematics Studies, vol. 95, Elsevier, 1984, pp. 257–355.

[29] Pedro Montealegre, Ioan Todinca, Brief announcement: deterministic graph connectivity in the broadcast congested clique, in: ACM Symposium on Principles of Distributed Computing, PODC, ACM, 2016, pp. 245–247.
[30] Irving S. Reed, Gustave Solomon, Polynomial codes over certain finite fields, J. Soc. Ind. Appl. Math. 8 (2) (1960) 300–304.
[31] Edward R. Scheinerman, Jennifer Zito, On the size of hereditary classes of graphs, J. Comb. Theory, Ser. B 61 (1) (1994) 16–39.
[32] Jacob T. Schwartz, Fast probabilistic algorithms for verification of polynomial identities, J. ACM 27 (4) (1980) 701–717.
[33] Leslie G. Valiant, A bridging model for parallel computation, Commun. ACM 33 (8) (1990) 103–111.
[34] Tom White, Hadoop: The Definitive Guide, O'Reilly Media, Inc., 2012.
[35] Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, Ion Stoica, Spark: cluster computing with working sets, HotCloud 10 (10-10) (2010) 95.