Two Rounds are Enough for Reconstructing any Graph (Class) in the Congested Clique Model*

P. Montealegre[1(✉)], S. Perez-Salazar[2], I. Rapaport[3], and I. Todinca[4]

[1] Facultad de Ingeniería y Ciencias, Universidad Adolfo Ibáñez, Santiago, Chile
p.montealegre@edu.uai
[2] ISyE, Georgia Institute of Technology, Atlanta, USA
sperez@gatech.edu
[3] DIM-CMM (UMI 2807 CNRS), Universidad de Chile, Santiago, Chile
rapaport@dim.uchile.cl
[4] Université d'Orléans, INSA Centre Val de Loire, LIFO EA 4022, France
ioan.todinca@univ-orleans.fr

Abstract. In this paper we study the *reconstruction problem* in the congested clique model. In the reconstruction problem nodes are asked to recover *all the edges* of the input graph G. Formally, given a class of graphs $\mathcal{G}$, the problem is defined as follows: if $G \notin \mathcal{G}$, then every node must reject; on the other hand, if $G \in \mathcal{G}$, then every node must end up knowing all the edges of G. It is not difficult to see that the cost Rb of *any algorithm* that solves this problem (even with public coins) is at least $\Omega(\log |\mathcal{G}_n|/n)$, where $\mathcal{G}_n$ is the subclass of all n-node labeled graphs in $\mathcal{G}$, R is the number of rounds and b is the bandwidth.

We prove here that the lower bound above is in fact tight and that it is possible to achieve it with only $R = 2$ rounds and private coins. More precisely, we exhibit (i) a one-round algorithm that achieves this bound for hereditary graph classes; and (ii) a two-round algorithm that achieves this bound for arbitrary graph classes. Later, we show that the bound $\Omega(\log |\mathcal{G}_n|/n)$ cannot be achieved in one round for arbitrary graph classes, and we give tight algorithms for that case.

From (i) we recover all known results concerning the reconstruction of graph classes in one round and bandwidth $\mathcal{O}(\log n)$: forests, planar graphs, cographs, etc. But we also get new one-round algorithms for other hereditary graph classes such as unit-disc graphs, interval graphs, etc. From (ii), we can conclude that *any problem* restricted to a class of graphs of size $2^{\mathcal{O}(n \log n)}$ can be solved in the congested clique model in two rounds, with bandwidth $\mathcal{O}(\log n)$. Moreover, our general two-round algorithm is valid for any set of labeled graphs, not only for graph classes.

Keywords: congested clique · round complexity · reconstruction problem · graph classes · hereditary graphs

* Partially supported by CONICYT PIA/ Apoyo a Centros Científicos y Tecnológicoss de Excelencia AFB 170001 (P.M. and I.R.), Fondecyt 1170021 (I.R.) and CONICYT + PAI + CONVOCATORIA NACIONAL SUBVENCIÓN A INSTALACIÓN EN LA ACADEMIA CONVOCATORIA AÑO 2017 + PAI77170068 (P.M.).

1 Introduction

The *congested clique* model –a message-passing model of distributed computation where the underlying communication network is the complete graph [20]– is receiving increasingly more attention [4, 8–11, 14]. This model allows us to separate and understand the impact of congestion 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. In other words, by isolating the effect of the bandwidth, we intend to understand it. 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 increasingly less exceptional [5, 21, 26]. For instance, in [12], the authors show that fast algorithms in the congested clique model can be translated into fast algorithms in the MapReduce model. Many theoretical models, aiming to bridging the gap between theory and previously mentioned softwares, have emerged [1, 16, 17] (These models are all very similar, but not completely identical, to the congested clique 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 Rb represents 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 subcases 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, $\Pr(\mathcal{A}$ 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 (whp).

Function f defines the problem to be solved. A $0-1$ function corresponds to a decision problem (such as connectivity [11]). For other, more general types of problems, f should be defined, in fact, as a relation. This happens, for instance, when we want to construct a minimum spanning tree [9, 14], a maximal independent set [10], a 3-ruling set [13], all-pairs shortest-paths [4], etc.

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 v has full knowledge of G, then it could answer *any question* concerning G. (This holds because in the congested clique model nodes have unbounded computational power).

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 are going to use an analogous approach here, in the congested clique model. But, as we are going to explain later, surprisingly, the complexity of the reconstruction problem *will only depend on the cardinality* of the subclass of n-node graphs in $\mathcal{G}$.

Formally, for any fixed set of graphs $\mathcal{G}$, we are going to introduce two problems. The first one, the *strong recognition 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 recognition 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

For any positive integer n we define $\mathcal{G}_n$ as the set of n-node graphs in $\mathcal{G}$. There is an obvious lower bound for Rb, even for the weak reconstruction problem $\mathcal{G}$-Weak-Rec and even in the public-coin setting. In fact, $Rb = \Omega(\log |\mathcal{G}_n|/n)$. This can be easily seen if we note 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 this paper we are going to 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).

1.1 Our Results

We start this paper by studying a very natural family of graph classes known as *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, d-degenerate graphs, etc. [3]. Moreover, any intersection class of graphs –such as interval graphs, chordal graphs, unit disc graphs, etc.– is also hereditary [3].

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 is a consequence of the requirement that *all nodes* must compute the output after one round). We also remark that for many hereditary graph classes, including all classes listed above, our algorithm is tight. Moreover, 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, 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 whp. 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 whp, both of them using bandwidth $\mathcal{O}(\sqrt{\log|\mathcal{G}_n| \log n} + \log n)$. We finish Section 5 pointing out that these algorithms, with respect to the bandwidth, are tight.

1.2 Some Remarks

Lenzen's algorithm. Lenzen's algorithm performs a load balancing procedure in the congested clique model [18]. Therefore, if the input graph is sparse, it solves the reconstruction problem very fast (by simply distributing all the edges among the nodes, and then broadcasting everything). For instance, if the input graph G contains $\mathcal{O}(n)$ edges, then it reconstruct G in a constant number of rounds with

bandwidth $\mathcal{O}(\log n)$. Our result is much more general. We do not need the graphs 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 graphs. In Section 4 we prove that, if the class $\mathcal{G}$ is such that $|\mathcal{G}_n| = 2^{\mathcal{O}(n \log n)}$, then there exists a three-round deterministic algorithm that reconstructs $\mathcal{G}$ using bandwidth $\mathcal{O}(\log n)$. Therefore, our three-round deterministic algorithm can be applied to sparse graphs, interval graphs, etc.

Broadcast congested clique. Consider the case where $\mathcal{G}$ is indeed 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 Section 5 we prove 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.

Reconstruction versus recognition. 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 [6], 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.

1.3 Techniques

The main techiques we use in this paper are *fingerprints* and *error correcting codes*. A fingerprint is a small representation of a large vector which statisfies that, if two vectors are different, then their fingerprints, whp, are also different [25]. We define in this paper the *fingerprint of a graph*, which is simply the collection of fingerprints of the rows of its adjacency matrix. Consider two graphs G and H defined on the same set of nodes. The fingerprints of these two graphs 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 the first, hereditary case, we prove that, for any graph 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.

In the second, general case, we use, together with fingerprints, error-corecting codes. More precicely, we use *Reed-Solomon codes* [23]. The idea of these codes consists in mapping a vector into a sligthly larger one, 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 constructions is that it mainly depends on the neighborhoods of the nodes (i.e., rows of the adjacency matrix), and can be implemented efficiently in the congested clique model.

1.4 Related Work

All known results concerning the reconstruction of graphs obtained so far, have been obtained in the context of hereditary graph classes. For instance, let $\mathcal{G}$ be the class of *cograph*, that is, the class of graphs that do not contain the 4-node path as an induced subgraph. This class is obviously hereditary. In [15], the authors presented a one-round public-coin algorithm that solves $\mathcal{G}$-Strong-Rec with bandwidth $\mathcal{O}(\log n)$. Note that $|\mathcal{G}_n| = 2^{\Theta(n \log n)}$. The result we give in this paper is stronger, because our one-round algorithm needs the same bandwidth but uses private coins.

In [2, 22] it is shown that, if $\mathcal{G}$ is the class of *d-degenerate* graphs, then there is a one-round deterministic algorithm that solves $\mathcal{G}$-Strong-Rec with bandwidth $\mathcal{O}(d \log n) = \mathcal{O}(\log n)$. 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. Note that planar graphs (or more generally, bounded genus graphs), bounded tree-width graphs, graphs without a fixed graph H as a minor, are all d-degenerate, for some constant $d > 0$. Since the class of d-degenerate graphs is hereditary and satisfies $|\mathcal{G}_n| = 2^{\Theta(n \log n)}$, it follows, from this paper, the existence of a one-round private-coin randomized algorithm that solves $\mathcal{G}$-Strong-Rec with bandwidth $\mathcal{O}(\log n)$. However, the result of [2] for this particular class is stronger, since their algorithm is deterministic.

Another example of reconstruction with one-round deterministic algorithms can be found in [6]. There, the authors consider the class of graphs defined by one forbidden subgraph H. They show that such classes can be reconstructed deterministically in one round with bandwidth $b = \mathcal{O}((ex(n, H) \log n)/n)$, where $ex(n, H)$ is the *Turán number* of H, defined as be the maximum number of edges

in an n-node graph not containing an isomorphic copy of H as a subgraph. For example, if C_4 is the cycle of length 4, then $ex(n, C_4) = \mathcal{O}(n^{3/2})$. This implies that, if we define $\mathcal{G}$ as the class of graphs not containing C_4 as a subgraph, then there is a one-round *deterministic* algorithm that solves $\mathcal{G}$-Strong-Rec with bandwidth $\mathcal{O}(\sqrt{n}\log n)$.

2 Preliminaries

2.1 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 $f(u)$ and $f(v)$ 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 $G[U]$ 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 $G = (\{v_1, \ldots, v_n\}, E)$, we call $A(G)$ its *adjacency matrix*, i.e., the $0-1$ 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 $i \in [n] = \{1, \ldots, n\}$. 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 $d_r(A(G), A(H)) = k$.

2.2 Fingerprints

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 $FP(a, x) = \sum_{i \in [n]} a_i x^{i-1} \mod p$. (Note that 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. *[19] 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 $i \in [n]$. Moreover, for a graph of size n, and $T \in (\mathbb{F}_p)^n$ we call $FP(G,T)$ the fingerprint of $A(G)$ and T.

3 Reconstructing Hereditary Graph Classes

In this section we start giving the main result. Later we explain the consequence of this result on 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 *whp 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 algorithm consists in: (1) Each node i picks t_i in $\mathbb{F}_p$ uniformly at random (using private coins), and computes $FP(x_i,t_i)$. (2) Each node communicates t_i and $FP(x_i,t_i)$. (3) Every node constructs $T = (t_1,\ldots t_n)$ and $FP(G,T) = (FP(x_1,t_1),\ldots,F(x_n,t_n))$ from the messages sent in the communication round. Finally: (4) Every node looks in $\mathcal{G}_n$ for a graph H such that $FP(H,T) = FP(G,T)$. If such graph H exists, the algorithm outputs H, otherwise it *rejects*.

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

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

Now suppose that $H \neq G$ and let $k > 0$ 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:

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

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 $f(n)/n \leq f(n+1)/(n+1)$, and $\log |\mathcal{G}_n| \leq f(n)$. Therefore, $(|\mathcal{G}_k|)^{1/k} \leq 2^{f(k)/k} \leq 2^{f(n)/n}$. We deduce:

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

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 $F(H,T) \neq F(G,T)$, for every $H \in \mathcal{G}_n$. Hence, the algorithm solves $\mathcal{G}$-Strong-Rec whp.

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

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

□

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 whp, in one-round, with bandwidth $\Theta(\log |\mathcal{G}_n|/n + \log n)$. This matches the lower bound on the cost Rb (which must be satisfied even in the public coin setting).*

In [24], Scheinerman and Zito showed that hereditary graph classes have a very specific growing rate. They showed (Theorem 1 in [24]) 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 $|\mathcal{G}_n| = 2^{\Theta(n \log n)}$.

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 whp 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 error correcting codes.

Definition 1. *Let $0 \leq k \leq n$, and let q be the smallest prime number greater that $n+k$. 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 $|\{i \in [n+k] : C(x)_i \neq C(y)_i\}| \geq k$.

For sake of completeness, we give 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 $C(x) = (P_x(1), \ldots, P_x(n+k))$. 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*

$$(C(x_1)_{n+i}, \ldots, C(x_n)_{n+i}, \mathbf{0}) \in (\mathbb{F}_q)^{n+k},$$

where $\mathbf{0}$ 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:

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

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

Remark 1. 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, $|\{j \in [n+k] : C(A(G))_{i,j} \neq C(A(H))_{i,j}\}| > k$. 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 1, 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

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

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 $\mathcal{O}(\log |\mathcal{G}_n|/n + \log n)$ whp.*

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 $(C(x_1)_{i+n}, \ldots, C(x_n)_{i+n})$, 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 $C(G)$, nodes look for a graph H in $\mathcal{G}_n$ that satisfies $FP(C(G), T) = FP(C(H), T)$. 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. 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 $S = (S_1, \ldots, S_n)$, $FP(G, S) = (FP(x_1, S_1), \ldots, FP(x_n, S_n)$. 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 give the correct answer. Otherwise, it rejects whp. Indeed, if $G \notin \mathcal{G}_n$, then $H \neq G$, and from Lemma 1, $\Pr(FP(G, T) = FP(H, T)) \leq 1/n$. □

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 $1/n$.

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, in terms of the bandwidth, are tight.

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 $\mathcal{O}(\sqrt{\log|\mathcal{G}_n| \log n} + \log n)$ whp.*

Proof. The algorithm 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 $C(x_i)_{n+1}, \ldots, C(x_i)_{n+k} \in (\mathbb{F}_q)^k$, 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 $A(\widetilde{G})$. Therefore, every node can compute $FP(C(x_i), T_i)$, for all $i \in \{n+1, \ldots, n+k\}$, 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 $\mathcal{O}(\sqrt{\log |\mathcal{G}_n| \log n} + \log n)$. □

Now we are going to show that previous algorithms for solving $\mathcal{G}$-Weak-Rec and $\mathcal{G}$-Strong-Rec are in fact tight. For proving this, we are going to exhibit a class of graphs $\mathcal{G}$ satisfying $|\mathcal{G}_n| \leq 2^{\mathcal{O}(n)}$ 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).

Theorem 4. *There exists a class of graphs $\mathcal{G}^+$ satisfying $|\mathcal{G}_n^+| \leq 2^{\mathcal{O}(n)}$ 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 $(n+1), \ldots, n^2$. 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 $(n+1), \ldots, n^2$ 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 $(n+1), \ldots, n^2$.

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 $\Omega(\sqrt{n})$.

□

6 Discussion

Our result gives a straightforward, general strategy to solve arbitrary problems when the input graph belongs to some particular class of graphs. Hence, instead of designing ad-hoc algorithms to solve specific problems, we can reconstruct the input graph G and solve locally *any question* concerning G.

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 run in exponential local time.

However, note 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** [7]. 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.

References

1. P Beame, P Koutris, and D Suciu. Communication steps for parallel query processing. *Journal of the ACM* 64(6), Article 40, 2017.
2. F Becker, M Matamala, N Nisse, I Rapaport, K Suchan, and I Todinca. Adding a referee to an interconnection network: What can(not) be computed in one round. In *IPDPS '11*, 508–514, 2011.
3. A Brandstädt, V B Le, and J P Spinrad. *Graph classes: a survey.* SIAM, 1999.
4. K Censor-Hillel, P Kaski, J H Korhonen, C Lenzen, A Paz, and J Suomela. Algebraic methods in the congested clique. In *PODC '15*, 143–152, 2015.
5. J Dean and S Ghemawat. MapReduce: simplified data processing on large clusters. *Communications of the ACM* 51(1), 107–113, 2008.
6. A Drucker, F Kuhn, and R Oshman. On the power of the congested clique model. In *PODC '14*, 367–376, 2014.
7. M R Garey and D S Johnson. *Computers and Intractability.* W H Freeman, New York, 2002.

8. M Ghaffari. An improved distributed algorithm for maximal independent set. In *SODA '16*, 270–277, 2016.
9. M Ghaffari and M Parter. MST in log-star rounds of congested clique. In *PODC '16*, 19–28, 2016.
10. M Ghaffari. Distributed MIS via All-to-All Communication. In *PODC '17*, 141–149, 2017.
11. J W Hegeman, G Pandurangan, S V Pemmaraju, V Sardeshmukh, and M Scquizzato. Toward optimal bounds in the congested clique: Graph connectivity and MST. In *PODC '15*, 91–100, 2015.
12. J W Hegeman and S V Pemmaraju. Lessons from the congested clique applied to MapReduce. In *SIROCCO '14*, 149–164, 2014.
13. J W Hegeman, S V Pemmaraju, and V Sardeshmukh. Near-constant-time distributed algorithms on a congested clique. In *DISC '14*, 514–530, 2014.
14. T Jurdzinski and K Nowicki. MST in O(1) rounds of the congested clique. In *SODA '18*, 2620–2632, 2018.
15. J Kari, M Matamala, I Rapaport, and V Salo. Solving the induced subgraph problem in the randomized multiparty simultaneous messages model. In *SIROCCO '15*, 370–384, 2015.
16. H Karloff, S Suri, and S Vassilvitskii. A model of computation for MapReduce. In *SODA '10*, 938–948, 2010.
17. H Klauck, D Nanongkai, G Pandurangan, and P Robinson. Distributed computation of large-scale graph problems. In *SODA '15*, 391–410, 2015.
18. C Lenzen. Optimal deterministic routing and sorting on the congested clique. In *PODC '13*, 42–50, 2013.
19. R Lidl and H Niederreiter. *Introduction to finite fields and their applications*. Cambridge University Press, 1994.
20. Z Lotker, B Patt-Shamir, E Pavlov, and D Peleg. Minimum-weight spanning tree construction in O $(\log \log n)$ communication rounds. *SIAM J. Comput.*, 35(1):120–131, 2005.
21. G Malewicz, M H Austern, A J C Bik, J C Dehnert, I Horn, N Leiser, and G Czajkowski. Pregel: a system for large-scale graph processing. In *SIGMOD '10*, 135–146, 2010.
22. P Montealegre and I Todinca. Brief announcement: Deterministic graph connectivity in the broadcast congested clique. In *PODC '16*, 245–247, 2016.
23. I S Reed and G Solomon. Polynomial codes over certain finite fields. *J. Soc. Ind. Appl. Math.*, 8(2):300–304, 1960.
24. E R Scheinerman and J Zito. On the size of hereditary classes of graphs. *Journal of Combinatorial Theory, Series B*, 61(1):16–39, 1994.
25. J T Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *J. of the ACM*, 27(4):701–717, 1980.
26. T White. Hadoop: The definitive guide. O'Reilly Media, Inc., 2012.