The Role of Randomness in the Broadcast Congested Clique Model ☆

Florent Becker[a], Pedro Montealegre[b], Ivan Rapaport[c,*], Ioan Todinca[a]

[a] *Université d'Orléans, INSA Centre Val de Loire, LIFO EA 4022, Orléans , France*
[b] *Facultad de Ingeniería y Ciencias, Universidad Adolfo Ibáñez, Santiago, Chile*
[c] *DIM and CMM (UMI 2807 CNRS), Universidad de Chile, Santiago, Chile*

Abstract

We study the role of randomness in the broadcast congested clique model. This is message-passing model of distributed computation where the nodes of a network know their local neighborhoods and they broadcast, in synchronous rounds, messages that are visible to every other node.

This works aims to separate three different settings: deterministic protocols, randomized protocols with private coins, and randomized protocols with public coins. We obtain the following results:

- If more than one round is allowed, public randomness is as powerful as private randomness.
- One-round public-coin algorithms can be exponentially more powerful than deterministic algorithms running in several rounds.
- One-round public-coin algorithms can be exponentially more powerful than one-round private-coin algorithms.
- One-round private-coin algorithms can be exponentially more powerful than one-round deterministic algorithms.

☆ A preliminary version of (part of) this work appeared in the proceedings of the 21st International Colloquium on Structural Information and Communication Complexity SIROCCO 2014, held in Hida Takayama, Japan.

* Corresponding author. Addresss: Beauchef 851, Edificio Norte, Santiago, Chile.

Email addresses: florent.becker@univ-orleans.fr (Florent Becker), p.montealegre@uai.cl (Pedro Montealegre), rapaport@dim.uchile.cl (Ivan Rapaport), ioan.todinca@univ-orleans.fr (Ioan Todinca)

Keywords: Distributed computing, broadcast congested clique, message size complexity, private and public coins, simultaneous multi-party communication

1. Introduction

The *congested clique* model is a message-passing model of distributed computation introduced by Lotker, Patt-Shamir, Pavlov, and Peleg [1]. 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, we intend to understand the effect of the bandwidth by isolating it.

Despite the theoretical motivation of the congested clique model, examples of distributed and parallel systems where the efficiency depends heavily on the bandwidth, are increasingly less exceptional (Mapreduce [2], Pregel [3], Spark [4], Hadoop [5], Dryad [6], etc.).

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, which is the complete graph (communication is all-to-all).

Nodes execute an algorithm, communicating with each other in synchronous rounds with the objective of collectively computing some function f that depends on G. In this paper we study a variant of the congested clique model, namely, the *broadcast congested clique* model, where at each round each vertex broadcasts only one (the same) b-bit message through all of its $n-1$ incident communication links [7, 8]. In each round of the algorithm, each of the n nodes has to decide whether to *stop* or *continue*. An algorithm *halts* when every vertex stops. At that moment, *every node must know* $f(G)$. Hence, $f(G)$ is called the *output* of the distributed algorithm. The parameter b (the maximum size of a message sent by a node) 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.

Function f defines the problem to be solved. In this paper we are interested in *decision problems*. A decision problem is a function where the output is either 0 or 1, or *reject-accept*, respectively. In such a case, we say that G is *accepted* if every node outputs *accept* (or 1) and G is *rejected* if every vertex outputs *reject* (or 0). For example, in this paper we study problem Twins, which consists in accepting input graphs containing at least two vertices having the same neighborhood.

In the literature it is common to define the rejection of a distributed decision problem in a slightly different way. Indeed, when G is rejected, instead of asking that every vertex rejects, the most common convention is to ask that *at least one vertex* rejects. We remark that in the broadcast congested clique model, both definitions are equivalent, since communication is all-to-all, and therefore every vertex can learn in one round (and communicating one bit) whether every vertex accepts or not.

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 among 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 $\varepsilon = 1/n^{\mathcal{O}(1)}$, we say that $\mathcal{A}$ computes f *with high probability* (whp).

In this paper we are particularly interested in the *one-round* broadcast congested clique model. One can view this particular case as follows: nodes need to send, simultaneously, a b-bit message to some referee allowing him/her to answer, typically, questions of the form "Does the input graph G belong to the graph class $\mathcal{C}$?"

The goal of this work is to explore the role of randomness. Specifically, the difference between deterministic algorithms, public coin algorithms and private coin algorithms.

1.1. Our results

In Section 2 we study multi-round randomized algorithms. We start proving that there is essentially no difference between the public and the private

coin settings. More precisely, we extend a classic result of Newmann [9] and show that any public coin algorithm in the broadcast congested clique model can be simulated by a private coin algorithm (slightly increasing the number of rounds, the bandwidth and the error probability). In order to separate multi-round deterministic and randomized algorithms, we introduce function GraphEq(G) defined over labeled graphs of size $2n$. This function equals 1 if and only if the adjacency matrix of $G[\{1,\ldots,n\}]$ is equal to the adjacency matrix of $G[\{n+1,\ldots,2n\}]$. In other words , if for every $i,j \in [n]$, vertex j belongs to $N(i)$ implies that vertex $j+n$ belongs to $N(i+n)$ and vice versa. We prove that, in the public coin setting, the cost $R \cdot b$ of this function is $\mathcal{O}(\log n)$, even for one-round algorithms, while the cost in the deterministic setting is $\Omega(n)$. (The proof of the upper bound $\mathcal{O}(\log n)$ corresponding to the public coin algorithm appears in Section 3).

The previous results indicate that relevant differences between public and private randomness can only be found in one-round algorithms. In Section 3 we separate the deterministic, the private coin and the public coin settings of the one-round broadcast congested clique model. For achieving this we use problem Twins and some variants. The boolean function Twins(G) returns *accept* if and only if graph G has at least two twins (that is, two nodes having the same neighborhood). We also consider function Twin$_x(G)$, where x is the identifier of a node, and the result is 1 if and only if there is some other node having the same neighborhood of node identified with x. We prove that the deterministic message size complexity of Twins and Twins$_x(G)$ is $\Theta(n)$. Also, both functions can be computed with one-round public coin and one-round private coin algorithms of message size $\mathcal{O}(\log n)$. These algorithms, based on the classical fingerprint technique, have one-sided error $\mathcal{O}(1/n^c)$ for any constant $c > 0$. In order to separate the private and public coin settings we use a boolean function called Translated-Twins (see Section 1.3 for details). We prove that the message size complexity of this function is $\Omega(\sqrt{n})$ in the private coin setting, while it is $\mathcal{O}(\log n)$ in the public coin setting. The main results of this paper are summarized in Table 1. Observe in Table 1 the exponential gap between private coins and determinism in the message size complexity of Twin$_x$. This situation is very different from what happens in the simultaneous 2-player case, where the gap between private coins and determinism is at most the square root of n [10] (see Section 1.3 for details).

There are several natural problems that cannot be solved with randomized algorithms using $o(n)$ bits, even with public coins. This is the case of deciding whether the input graph contains a clique of size 4, or deciding

	TWINS	TWINS$_x$	TRANSLATED-TWINS
Det	$\Theta(n)$ [Thm 3]	$\Theta(n)$ [Thm 3]	$\Theta(n)$ [Thm 3]
Priv	$\mathcal{O}(\sqrt{n}\log n)$ [Thm 6]	$\mathcal{O}(\log n)$ [Thm 4]	$\Omega(\sqrt{n})$ [Thm 5] $\mathcal{O}(\sqrt{n}\log n)$ [Thm 6]
Pub	$\mathcal{O}(\log n)$ [Thm 4]	$\mathcal{O}(\log n)$ [Thm 4]	$\mathcal{O}(\log n)$ [Thm 4]

Table 1: Results regarding one-round algorithms in the broadcast congested clique model.

whether the input graph has a cycle of length 5. In both cases the cost $R \cdot b = \Omega(n)$ [7]. In Section 4 we complement those results, proving that the one-round randomized public coin message size complexity of the boolean functions TRIANGLE(G) (that outputs 1 if and only if G has a triangle) and DIAM3(G) (that outputs 1 if and only if G has diameter at most 3) is $\Omega(n)$. We obtain those results extending the arguments of [11], from the deterministic setting to the randomized one.

Finally, in Section 5, we revisit problem TWINS, but in the framework of the *unicast congested clique* model. In the unicast congested clique model, nodes are allowed to send different messages in each round (i.e., sending up to $n-1$ different messages each round). The unicast congested clique model is much more powerful than the broadcast one. In particular, we make use of the algebraic methods given in [12] in order to solve TWINS with cost $\mathcal{O}(n^{0.158}\log n)$.

1.2. Related work

1.2.1. The simultaneous multi-party model

The simultaneous two-player model was already present in Yao's seminal communication complexity paper of 1979 [13]. He proved that the message size complexity of EQ, which simply tests whether two n-bit inputs are equal, is $\Theta(n)$ in the deterministic case (in fact he proved that this is also true even if players can communicate back-and-forth). Later, clear separations have been proved between deterministic, private coin and public coin algorithms. In the public coin setting with constant one-sided error, the message size complexity of EQ is $\mathcal{O}(1)$ [14]. On the other hand, for private coin algorithms of constant one-sided error, the message size complexity is much higher, $\Theta(\sqrt{n})$ [14, 15] (see Section 1.3 for details). More generally, Babai and Kimmel [14] proved

that, for any function f, the gap between the deterministic and private coin message size complexities is at most the square root of n.

There are (at least) two natural ways to extend problem EQ to more than two players. This issue is addressed by Fischer, Oshman and Zwick [16] in the context of the *number-in-hand* model (where each player only knows its own input; players broadcast messages but they are not nodes of some input graph G). First, they define problem ALLEQ. In this simultaneous multi-party problem, each player receives a boolean vector $\{0,1\}^n$ and they have to decide whether all the k vectors are equal. In problem EXISTSEQ, the k players have to decide whether there exist *at least* two players with the same input. It is not difficult to see that in both the deterministic case and the public coin case the results for 2 players can be extended to k players (for EXISTSEQ this holds for $k \leq 2^n$, otherwise the problem is trivial). The private coin case is more involved. With respect to private coin algorithms of constant error, the authors of [16] prove, for problem ALLEQ, an upper bound of $\mathcal{O}(\sqrt{n/k} + \log(\min(n,k)))$ and a lower bound of $\Omega(\log n)$. In the case of EXISTSEQ the upper bound they show is $\mathcal{O}(\log k\sqrt{n})$ while the lower bound is $\Omega(\sqrt{n})$.

1.2.2. The broadcast congested clique model

An interesting example of a problem defined in the broadcast congested clique model where randomness helps is CONNECTIVITY, which is the decision problem of determining whether the input graph is connected. This problem can be easily solved in $\mathcal{O}(\log n)$ rounds and bandwidth $\mathcal{O}(\log n)$ by a deterministic algorithm simulating the classical algorithm of Boruvka [17]. A more careful analysis shows that the same argument can be used to solve the problem deterministically in two rounds and bandwidth $\mathcal{O}(\sqrt[2]{n}\log n)$ [18]. In [19] Jurdziński and Nowicki gave a deterministic algorithm for CONNECTIVITY that runs in $\mathcal{O}\left(\frac{\log n}{\log\log n}\right)$ rounds and bandwidth $\mathcal{O}(\log n)$. Not much is known with respect to lower bounds. In fact, in [20] Pai and Pemmaraju show that any algorithm solving CONNECTIVITY in the broadcast congested clique has cost $\Omega(\log n)$, far from the cost $\mathcal{O}\left(\frac{\log^2 n}{\log\log n}\right)$ of the algorithm of Jurdziński and Nowicki.

In [21, 22] Ahn et. al presented an extremely elegant one-round public coin algorithm that solves connectivity with high probability and bandwidth $\mathcal{O}(\log^4 n)$. The algorithm uses a technique called *linear sketches*, which roughly consists in compressing vectors through a linear function. This tech-

nique allows to sample (with high probability) a random edge from the cut induced by any set of vertices. The authors use this capability to simulate in only one round of communication the whole Boruvka's algorithm. It is not known whether private randomness helps to solve CONNECTIVITY.

Another example where randomness helps is RECONSTRUCTION, the problem consisting in computing the function $f(G) = E$, i.e., every vertex ends up knowing the adjacency matrix of G. Roughly speaking, this problem is the *hardest* problem that one might attempt to solve. Indeed, since in this model nodes have unbounded computational power, any vertex knowing the adjacency matrix of G can compute any property of G without any further communication. Of course any algorithm (randomized or deterministic) solving RECONSTRUCTION has cost $\Theta(n)$. For that reason a restricted version of RECONSTRUCTION is considered, where a set of graphs $\mathcal{G}$ is fixed. $\mathcal{G}$-RECONSTRUCTION is the function that outputs the adjacency matrix of G when G belongs to $\mathcal{G}$, and outputs *reject* when G does not belong to $\mathcal{G}$.

In [23] it is shown that $\mathcal{G}$-RECONSTRUCTION can be solved with a deterministic one-round algorithm with bandwidth $\mathcal{O}(\log n)$, when $\mathcal{G}$ is the class of trees, planar graphs, or any class defined by a finite set of excluded minors. In fact, the algorithm of [23] solves the problem with bandwidth $\mathcal{O}(d^2 \log n)$ for every set of graphs of degeneracy at most d (trees have degeneracy 1, planar graphs have degeneracy 5). The upper bound on the bandwidth was improved to $\mathcal{O}(d \log n)$ in [18].

In [24] it is shown that any algorithm (deterministic or randomized) solving $\mathcal{G}$-RECONSTRUCTION has cost $\Omega(\frac{\log |\mathcal{G}_n|}{n})$, where $\mathcal{G}_n$ is the set of n-node graphs in $\mathcal{G}$. A common property of d-degenerate sets of graphs is that $\mathcal{G}_n = \mathcal{O}(2^{dn \log n})$, so the lower bound can be reached by a deterministic one-round algorithm. This is not the case for any set of graphs. There are sets of graphs $\mathcal{G}$ such that $\mathcal{G}_n = 2^{\mathcal{O}(n)}$, but any one-round deterministic algorithm solving $\mathcal{G}$-RECONSTRUCTION has cost $\Omega(n)$ [24].

The use of randomness improves the upper bounds on the bandwidth with respect to deterministic algorithms. Indeed, for any set of graphs $\mathcal{G}$, there is a two-round public coin algorithm that solves $\mathcal{G}$-RECONSTRUCTION with bandwidth $\mathcal{O}(\sqrt{\log |\mathcal{G}_n| \log n})$. Moreover, if G is an hereditary class of graphs, $\mathcal{G}$-RECONSTRUCTION can be solved with a one-round private coin algorithm with bandwidth $\mathcal{O}(\frac{\log |\mathcal{G}_n|}{n})$, matching the lower-bound given above [24].

1.3. Preliminaries

All our graphs are undirected. So, for any pair i, j of nodes, the bit number i of node (with identifier) j equals the bit number j of node i. In full words, each edge of the graph is known by the two players corresponding to its end-nodes. All our algorithms use $\Omega(\log n)$ bits. We assume, w.l.o.g., that in each round each node sends its own number in the message transmitted to the referee (or broadcasted to the other nodes, or written in a whiteboard visible to every node; all these definitions are equivalent).

In this paper we study, among others, the following three boolean functions on graphs.

- TWINS(G) outputs 1 if and only if G has two vertices u and v with the same neighborhood, i.e., such that $N(u) = N(v)$.
- TWINS$_x(G)$ is a "pointed" version of previous function. Its output is 1 if and only if there is a vertex $y \neq x$ such that $N(y) = N(x)$.
- TRANSLATED-TWINS is defined on input graphs G of size $2n$, labeled from 1 to $2n$. Its output is 1 if and only if G has a vertex $i \in [n]$ such that, for any vertex $j \in [n]$, $j \in N(i) \iff j + n \in N(i + n)$. In other words, the output is 1 if and only if there exists i such that $N(i) + n = N(i + n)$.

For reductions we also use function RECONSTRUCTION(G), whose output is G itself, i.e., the adjacency matrix of G. Note that if a deterministic algorithm computes RECONSTRUCTION on some family of n-vertex graphs $\mathcal{G}_n$, then the cost $R \cdot b$ of such algorithm must be $\Omega(\log(|\mathcal{G}_n|)/n)$ [11].

The *message cost* of a one-round algorithm $\mathcal{P}$, denoted by $C(\mathcal{P})$, is the length of the longest message sent to the referee. The *deterministic message size complexity*, denoted $C^{\det}(f)$, is the minimum message of any one-round deterministic algorithm computing f. Analogously, we denote $C^{\text{priv}}_{\epsilon}(f)$, $C^{\text{pub}}_{\epsilon}(f)$, the message size complexity for ϵ-error public coin and ϵ-error private coin algorithms, respectively.

The *simultaneous k-player model* can be seen as a generalization of the one-round broadcast congested clique (despite having been introduced much earlier). It is defined as follows. Let f be a function having as input k boolean vectors of length n. There are k players $\{p_1, \ldots, p_k\}$ who wish to compute the value of f on input $(x_1, \ldots, x_k) \in (\{0,1\}^n)^k$. Player p_i only sees

the input x_i, and also knows his/her own index i. All the k players simultaneously send a message to a *referee*. After that, the referee (another player who sees none of the inputs) announces the value $f(x_1, \ldots, x_k)$ using only the information contained in the k messages. The mesage size complexities $C^{\det}(f)$, $C^{\mathrm{priv}}_{\epsilon}(f)$, $C^{\mathrm{pub}}_{\epsilon}(f)$, are defined as above. (Note that, in the one-round broadcast congested clique model, there is some shared information between the nodes or players: in fact, each edge of the input graph is known *by two nodes*).

Let us recall some classical results on the simultaneous 2-player model. Babai and Kimmel [14] have shown that the order of magnitude of the private coin randomized message size complexity of any function f is at least the square root of the deterministic message size complexity of f. They also characterize completely the function $\textsc{Eq} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, where $\textsc{Eq}(x, y) = 1$ iff $x = y$.

Proposition 1 ([14]). *Consider the simultaneous 2-player model and a constant $\epsilon > 0$. The function* Eq *on two n-bit boolean vectors has the following message size complexities: $C^{\det}(\textsc{Eq}) = n$, $C^{\mathrm{priv}}_{\epsilon}(\textsc{EQ}) = \Theta(\sqrt{n})$ and $C^{\mathrm{pub}}_{\epsilon}(\textsc{Eq}) = \mathcal{O}(1)$. For any boolean function f, $C^{\mathrm{priv}}_{\epsilon}(f) = \Omega(\sqrt{C^{\det}(f)})$.*

We also use the following result of Chakrabarti *et al.* [10] for private-coin algorithms (the deterministic part is a matter of exercise).

Proposition 2 ([10]). *Consider the boolean function* OrEq *that takes as input two boolean $n \times n$ matrices, and the output is 1 if and only if there is some $1 \leq i \leq n$ such that the i-th rows of the two matrices are equal. Then, for any $\epsilon < 1/2$, $C^{\mathrm{priv}}_{\epsilon}(\textsc{OrEq}) = \Omega(n\sqrt{n})$. Also, $C^{\det}(\textsc{OrEq}) = \Theta(n^2)$.*

2. Public versus private randomness for multi-round algorithms

We start this section by showing that, in the broadcast congested clique model, private and public coin multi-round algorithms are equivalent (up to a logarithmic factor).

Theorem 1. *Let $\epsilon, \delta > 0$ be such that $\epsilon < 1/3$, and let f be a function $f : \mathcal{G}_n \to \{0,1\}$, where $\mathcal{G}_n$ is the set of all graphs of size n. For every R-round ϵ-error public coin algorithm computing f in the broadcast congested clique model with bandwidth b, there exists an $(R+1)$-round $(\epsilon+\delta)$-error private coin algorithm computing f in the broadcast congested clique model with bandwidth $b + \log(n) + \log(\delta^{-1})$.*

To show this theorem we follow the same arguments of Newman for the two-player case [9]. Intuitively, to construct a private coin algorithm from a public coin algorithm, we may ask to an arbitrary node to privately generate the number of random bits required to run the public coin algorithm, and communicate those bits to the other nodes in a first round. However, in our definition of randomized algorithms, we did not bound the number of random bits that an algorithm is able to use, so this first round may require a very big bandwidth (or too many rounds). In the next lemma, we show that by slightly increasing the error, we can transform any public coin algorithm that uses an arbitrary number of random bits, into another public coin algorithm that uses a number of random bits that is logarithmic in the size of the input.

Lemma 1. *Let $\epsilon, \delta > 0$ be such that $\epsilon < 1/3$, and let f be a function $f : \mathcal{G}_n \to \{0,1\}$, where $\mathcal{G}_n$ is the set of all graphs of size n. Suppose that there exists an R-round ϵ-error public coin algorithm $\mathcal{P}$ computing f in the broadcast congested clique model with bandwidth b. Then, there exists an R-round $(\epsilon+\delta)$-error public coins algorithm $\mathcal{P}'$ computing f in the broadcast congested clique model with bandwidth b that uses $\mathcal{O}(\log n + \log \delta^{-1})$ public random bits.*

Proof Roughly, the proof shows that it is possible to fix a *small* sample of the set of all possible random bis produced by $\mathcal{P}$. When the algorithm picks vectors uniformly at random from the sample, there is only a *small* increase on the error of the algorithm. This sample is represented by a vector $\vec{q}$, which we are assuming to be public (it is a part of algorithm $\mathcal{P}'$). In $\mathcal{P}'$ the nodes collectively choose a vector from $\vec{q}$ uniformly at random using the public coins, and then simulate $\mathcal{P}$.

Let G be the input graph and denote by x_i the input of node i. Let $\mathbf{q}$ be the random variable corresponding to the public random bits produced by $\mathcal{P}$ on input G. We call $\mathcal{P}(G, \mathbf{q})$ the random variable corresponding to the output of the algorithm $\mathcal{P}$ on input G and public coins $\mathbf{q}$. Now denote by $Z(G, \mathbf{q})$ the random variable indicating whether $\mathcal{P}(G, \mathbf{q})$ equals $f(G)$. Formally:

$$Z(G, \mathbf{q}) = \begin{cases} 1 & \text{if } \mathcal{P}(G, \mathbf{q}) \neq f(G), \\ 0 & \text{otherwise.} \end{cases}$$

Since $\mathcal{P}$ has error probability ϵ, then for every fixed graph G, the expected value of $Z(G, \mathbf{q})$ is ϵ. Indeed, $E_q(Z(G, \mathbf{q})) = Pr(Z(G, \mathbf{q}) = 1) = Pr(\mathcal{P}(G, \mathbf{q}) \neq f(G)) = \epsilon$.

For $t = (2n^2)/(3\delta^2)$, let $\vec{q} = q_1, \ldots q_t$ be t realizations of the public coins of $\mathcal{P}$. Define the following public coin algorithm $\mathcal{P}_{\vec{q}}$ as follows. On input G, each node picks (using the public coin) uniformly at random an integer $i \in \{1, \ldots, t\}$, and then runs $\mathcal{P}$ on input G with random string q_i. In other words, they pick $i \in \{1, \ldots t\}$ and, compute $\mathcal{P}(G, q_i)$.

Suppose we pick $\vec{q} = q_1, \ldots, q_t$ independently and uniformly at random, and fix a graph G. For each $i \in [t]$, let $\eta_i(G)$ be a random variable defined as $\eta_i(G) = Z(G, q_i) - \epsilon$. Note that $\{\eta_1(G), \ldots, \eta_t(G)\}$ are independent, $E_{\vec{q}}(\eta_i(G)) = E_{\vec{q}}(Z(G, q_i)) - \epsilon = 0$, and $|\eta_i(G)| \leq 1$ (since $\epsilon < 1/2$). Therefore, we can use a Chernoff bound to obtain:

$$Pr_{\vec{q}}\left(\sum_{i\in[t]} \eta_i(G) > \delta \cdot t\right) \quad \leq \quad e^{-\delta^2 \cdot t/2} = 2^{-\delta^2 \cdot t/2 \cdot \log(e)} = 2^{-n^2 \cdot \log(e)/3} < 2^{-n^2}.$$

Then,

$$\begin{aligned} Pr_{\vec{q}}\left(\exists G \in \mathcal{G}_n, \textstyle\sum_{i\in[t]} \eta_i(G) > \delta \cdot t\right) \quad &\leq \textstyle\sum_{G\in\mathcal{G}_n} Pr_{\vec{q}}\left(\sum_{i\in[t]} \eta_i(G) > \delta \cdot t\right) \\ &< \textstyle\sum_{G\in\mathcal{G}_n} 2^{-n^2} = 2^{\binom{n}{2}} \cdot 2^{-n^2} < 1 \end{aligned}$$

Thus

$$Pr_{\vec{q}}\left(\forall G \in \mathcal{G}_n, \sum_{i\in[t]} \eta_i(G) \leq \delta \cdot t\right) > 0.$$

We deduce that there exists a choice $\overline{q} = (q_1, \ldots, q_t)$ such that

$$\frac{1}{t}\sum_{i\in[t]} Z(G, q_i) \leq \epsilon + \delta \quad \forall G \in \mathcal{G}_n$$

We define $\mathcal{P}' = \mathcal{P}_{\overline{q}}$, and we obtain that, $\forall G \in \mathcal{G}_n$:

$$\begin{aligned} Pr_i(\mathcal{P}'(G, i) \neq f(G)) \quad &= \quad \sum_{i\in[t]} Pr([\text{the algorithm chooses } i] \wedge [\mathcal{P}(G, q_i) \neq f(G)]) \\ &\leq \quad \frac{1}{t}\sum_{i\in[t]} Z(G, q_i) \leq \epsilon + \delta \end{aligned}$$

Therefore $\mathcal{P}'$ is an R-round $(\epsilon + \delta)$-error public coin algorithm for f in the broadcast congested clique model, that uses $\lceil \log t \rceil = \mathcal{O}(\log n + \log \delta^{-1})$ random bits. $\square$

We are now ready to show Theorem 1.

Proof of Theorem 1. Let $\mathcal{P}$ be an algorithm as in the statement of the theorem, and for $\delta > 0$ let $\mathcal{P}'$ be the algorithm obtained from $\mathcal{P}$ using Lemma 1, i.e., $\mathcal{P}'$ is an R-round $(\epsilon+\delta)$-error public coin algorithm in the broadcast congested clique model, that uses $\lceil \log t \rceil = \mathcal{O}(\log n + \log \delta^{-1})$ random bits. We simply construct an $(\epsilon+\delta)$-error private coin algorithm $\mathcal{P}''$ on the broadcast congested clique model as follows. First, node 1 produces (using its private coins) a Boolean random vector q of size $\lceil \log t \rceil$, and communicates it in the first round. Then, all the nodes simulate $\mathcal{P}'$ on their inputs considering q as a public random vector. Clearly, $\mathcal{P}''$ runs in $R+1$ rounds and has bandwidth $b + \lceil \log t \rceil = \mathcal{O}(b + \log n + \log \delta^{-1})$. □

We end this section giving a deterministic lower bound for multi-round protocols solving GraphEq. This problem separates deterministic from randomized multi-round algorithms, because, as we will show in Theorem 4, there is a one-round public coin algorithm solving GraphEq with high probability and bandwidth $\mathcal{O}(\log n)$. (Recall that GraphEq(G) is defined over labeled graphs of size $2n$: it equals 1 if and only if the adjacency matrix of $G[\{1,\ldots,n\}]$ is equal to the adjacency matrix of $G[\{n+1,\ldots,2n\}]$).

Theorem 2. *Any deterministic algorithm that computes* GraphEq *in the broadcast congested clique model has cost* $R \cdot b = \Omega(n)$.

Proof Let $\mathcal{P}$ be a deterministic algorithm that computes GraphEq in the broadcast congested clique model, with cost $g(n)$ on input graphs of size $2n$. Call $m = \binom{n}{2}$, and let $x, y \in \{0,1\}^m$ be an input of problem Eq_m. Let G_x and G_y be graphs of size n, where the vertex set is $[n]$ and the edge sets are given by x and y, interpreted as their adjacency matrices. Consider now the graph of size $2n$ called $G_{x,y}$, which is the union of G_x and G_y, after relabeling each vertex i of G_y by $i+n$ (see Figure 1).

Note that GraphEq$(G_{x,y})$ equals 1 if and only $G_x = G_y$, i.e., if $x = y$. We define a two-player algorithm $\mathcal{P}'$ for Eq_m as follows. At each round, Alice (resp. Bob) simulates algorithm $\mathcal{P}$ on each vertex of G_x (resp. G_y), and communicates the obtained messages. Since the cost of $\mathcal{P}$ in $G_{x,y}$ is $g(n)$, the cost of algorithm $\mathcal{P}'$ is $n \cdot g(n)$. Recall that the deterministic cost of Eq_m is $\Omega(m)$. We obtain that $n \cdot g(n) = \Omega(\binom{n}{2})$, so $g(n) = \Omega(n)$. □

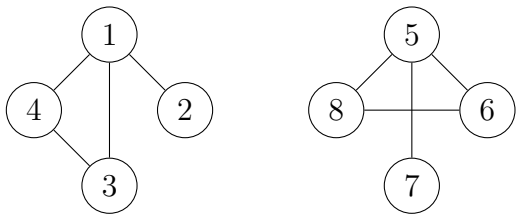


Figure 1: Reduction of EQ_m to GRAPHEQ where $n = 4$, and the inputs of EQ_m are $x = (1, 1, 1, 0, 0, 1)$ and $y = (1, 1, 1, 0, 1, 0)$ (the coordinates 1 (resp. 0) in x represent the presence (absence) of edges $(\{1, 2\}, \{1, 3\}, \{1, 4\}, \{2, 3\}, \{2, 4\}, \{3, 4\})$ in $G_{x,y}$, while the coordinates of y represent edges $(\{5, 6\}, \{5, 7\}, \{5, 8\}, \{6, 7\}, \{6, 8\}, \{7, 8\})$.

3. One-round algorithms

3.1. Deterministic algorithms

Theorem 3. *The one-round deterministic message size complexity of functions* TWINS, TWIN_x *and* TRANSLATED-TWINS *is* $\Theta(n)$.

The upper bounds of $\mathcal{O}(n)$ are trivial so we only need to prove the lower bounds. For the first two problems, we use the following reduction.

Lemma 2. *Assume that there is a one-round deterministic algorithm solving* TWINS *(resp.* TWINS_{2n+1}*) on* $(2n + 1)$*-node graphs using messages of size* $g(n)$*. Then one can solve* RECONSTRUCTION *in one round on* n*-node graphs using messages of size* $2g(n)$*.*

Proof Let G be an arbitrary n-node graph, i an integer between 1 and n and S a subset of $\{1, \ldots, n\}$ not containing i. Denote by $H(i, S)$ the graph on $2n + 1$ nodes obtained as follows (see Figure 2):

1. $H[\{1, \ldots, n\}] = G$.
2. For each $n + 1 \leq j \leq 2n$, its unique neighbor with identifier at most n is $j - n$.
3. Node $2n+1$ is adjacent to the nodes of S and to $i+n$ (and to no other node).

Claim. We claim that $\text{TWINS}(H(i, S))$ (resp. $\text{TWINS}_{2n+1}(H(i, S))$ is true if and only if $N_G(i) = S$.

Clearly, if $N_G(i) = S$ then node i is a twin of $2n + 1$ in graph H. Conversely, we prove that if $H(i, S)$ has two twins u and v then one of them

is $2n+1$. This comes from the fact that the edges between $\{1,\ldots,n\}$ and $\{n+1,\ldots,2n\}$ in $H(i,S)$ form a matching. So, no two nodes of $\{1,\ldots,2n\}$ may be twins. Now, assume that $2n+1$ has a twin u. Since $N_{H(i,S)}(2n+1)\cap\{n+1,\ldots,2n\}=\{i+n\}$, the only possibility is that $u=i$. Eventually, i and $2n+1$ are twins if and only if $N_G(i)=S$, which proves our claim.

Now, assume that we have a distributed algorithm for TWINS (or for TWINS_{2n+1}) on graphs with $2n+1$ nodes. We construct an algorithm for RECONSTRUCTION on an arbitrary n-node graph G.

The players construct their messages as follows. Each node i sends the message m_i that it would send in the TWINS algorithm if it had neighborhood $N_G(i)\cup\{i+n\}$ and the message m_i^+ that it would send in the same algorithm with neighborhood $N_G(i)\cup\{i+n,2n+1\}$. That makes messages of size $2g(n)$.

The referee needs to retrieve the neighborhood $N_G(i)$ for each i, from the set of messages. For each i and each subset S of $\{1,\ldots,n\}$ not containing i, she simulates the behavior of the referee for TWINS on graph $H(i,S)$. For this purpose, for each $j\leq n$, she uses message m_j if $j\notin S$ and message m_j^+ if $j\in S$. The messages for nodes $k>n$ can be constructed directly by the referee. Note that $\text{TWINS}(H(i,S))$ is true iff $N_G(i)=S$. Thus, she can reconstruct $N_G(i)$. Eventually, this allows to solve RECONSTRUCTION on graph G. The same argument works if we replace the TWINS algorithm by TWINS_{2n+1}. □

Remark 1. *Since problem* RECONSTRUCTION *on n-node graphs requires messages of size $\Omega(n)$, we conclude that any deterministic algorithm for either* TWINS *or* TWINS_{2n+1} *also requires messages of size $\Omega(n)$.*

For problem TRANSLATED-TWINS, we provide a reduction from OREQ (see Proposition 2 in Section 1.3). This reduction will be used in both the deterministic and private coin settings. (Recall that OREQ takes as input two boolean matrices, and the output is 1 if and only if some row i of the two matrices are equal).

Lemma 3. *Assume that there is a one-round algorithm solving* TRANSLATED-TWINS *for $6n$-node graphs using messages of size $g(n)$, in any of our three settings. Then, there is a one-round algorithm for function* OREQ*, in the same setting, using messages of size $3ng(n)$.*

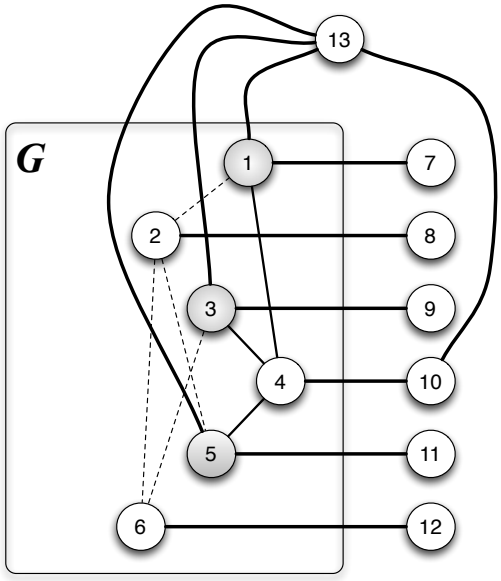


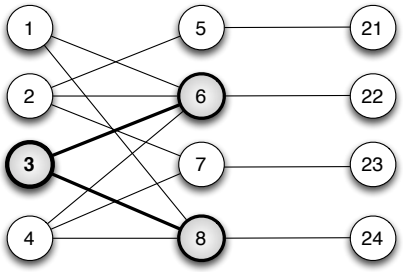
Figure 2: $H(4, S)$, when $S = \{1, 3, 5\}$.

Proof Let x and y be two $n \times n$ boolean matrices. We construct a graph $G_{x,y}$ with $6n$ nodes such that Translated-Twins$(G_{x,y})$ = OrEq(x, y). The graph G is formed by two connected components G_x^1 and G_y^2 of $3n$ nodes each, encoding the two matrices as follows (see Figure 3 for an example).

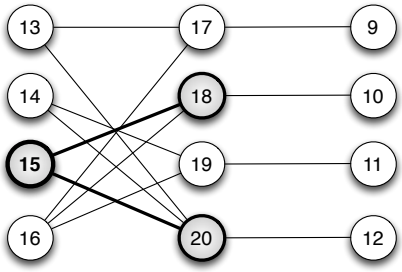
G_x^1 has $3n$ nodes numbered from 1 to $2n$ and from $5n + 1$ to $6n$. For any $i, j \in \{1, \ldots, n\}$ we put an edge between node i and node $j + n$ if and only if $x_{i,j} = 1$. Also, for any $i \in \{1, \ldots, n\}$ we put an edge between node $i + n$ and node $i + 4n$. In other words, the node subsets $\{1, \ldots, n\}$ and $\{n+1, \ldots, 2n\}$ induce a bipartite graph representing matrix x, and the node subsets $\{n + 1, \ldots, 2n\}$ and $\{5n + 1, \ldots, 6n\}$ induce a perfect matching.

The construction of G_y^2, with nodes numbered from $2n+1$ to $5n$ is similar. For any $i, j \in \{1, \ldots, n\}$ we put an edge between node $i+3n$ and node $j+4n$ if and only if $y_{i,j} = 1$. Also, for any $i \in \{1, \ldots, n\}$, we put an edge between node $4n + i$ and node $2n + i$. Thus the node subsets $\{3n + 1, \ldots, 4n\}$ and $\{4n + 1, \ldots, 5n\}$ form a bipartite graph corresponding to matrix y. The subsets $\{4n + 1, \ldots, 5n\}$ and $\{2n + 1, \ldots, 3n\}$ induce a matching.

We claim that Translated-Twins$(G_{x,y})$ = OrEq(x, y). Assume that OrEq$(x, y) = 1$. Then, there is an index i such that the i-th row of x equals the i-th row of y. Then, by construction, the neighborhood of node $i+3n$ in



$$x = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix}$$



$$y = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

Figure 3: Examples of graphs G_x^1 (top) and G_y^2 (bottom), for a given input (x, y). This is a *yes* instance since $x_3 = y_3$.

$G_{x,y}$ is the neighborhood of node i, translated by an additive term $3n$.

Conversely, assume that there is some node $u \in \{1, \ldots, 3n\}$ such that the neighborhood of u is the translated neighborhood of $u+3n$. By construction, the only possibility is that $u \leq n$ (because of the numberings of the matchings the other nodes cannot have translated twins). Thus, the u-th rows are the same in both matrices.

To complete the proof of our lemma, assume that we have a algorithm for Translated-Twins for graphs with $3n$ nodes, with $g(n)$ bits per message. We design the following algorithm for OrEq. (Recall that, in OrEq, each player has a matrix, say x for the first one and y for the second one). The first player constructs graph $G_{x,0} = (G_x^1, G_0^2)$, the second constructs $G_{0,y} = (G_0^1, G_y^2)$ (here 0 denotes the $n \times n$ boolean matrix whose elements are all 0). The first player sends the $3n$ messages corresponding to the nodes of G_x^1 in the Translated-Twins algorithm for graph $G_{x,0}$. The second player sends the $3n$ messages corresponding to the nodes of G_y^2 in algorithm Translated-Twins for $G_{0,y}$. The referee collects these $6n$ mes-

sages; observe that they are exactly those sent by algorithm TRANSLATED-TWINS for the graph $G_{x,y}$. She applies the same algorithm as the referee of TRANSLATED-TWINS would apply on these messages. By the claim above, its output is TRANSLATED-TWINS$(G_{x,y})$. Thus, OREQ(x, y). Note that the messages used here are of size $\mathcal{O}(3ng(n))$ and that our arguments hold for any type of algorithm. □

Remark 2. *Since, in the deterministic framework, problem* OREQ *requires messages of size* $\Omega(n^2)$*, we conclude that any one-round deterministic algorithm for problem* TRANSLATED-TWINS *requires messages of size* $\Omega(n)$*. This completes the proof of Theorem 3.*

Remark 3. *The lower-bound construction given in the proof of Lemma 3 also holds for connected graphs. Indeed, it is enough to add an edge between vertices numbered* $5n + 1$ *and* $2n + 1$ *to obtain a connected graph without creating unwanted translated-twins.*

3.2. Randomized algorithms

Theorem 4. *For any constant* $c > 0$*,* TWINS*,* TWIN$_x$*,* TRANSLATED-TWINS *and* GRAPHEQ *can be solved with one-round public coin algorithms using messages of size* $O(\log n)$ *and* $1/n^c$ *one-sided error. Problem* TWIN$_x$ *can also be solved with a one-round private coin algorithm using messages of size* $O(\log n)$ *and* $1/n^c$ *one-sided error.*

Proof Let $n^{c+3} < p \leq 2n^{c+3}$ be a prime number. A random $t \in \mathbb{Z}_p$ is chosen uniformly at random using $\mathcal{O}(\log(n))$ public random bits (the prime field of order p is represented by $\mathbb{Z}_p$, the set of integers modulo p). Given an n-bit vector $a = (a_1, \ldots, a_n)$, consider the polynomial $P_a = a_1 + a_2X + a_3X^2 + \ldots a_nX^{n-1}$ in $Z_p[X]$ and let $FP(a, t) = P_a(t)$. $FP(a, t)$ is sometimes called the "fingerprint" of vector a. Clearly, two equal vectors have equal fingerprints, and, more important, for any two different vectors a and b, the probability that $FP(a, t) = FP(b, t)$ is at most $1/n^{c+2}$ (because the polynomial $P_a - P_b$ has at most n roots and t was chosen uniformly at random, thus the probability that t is a root of $P_a - P_b$ is at most $1/n^{c+2}$, see e.g., [25]).

Let x_i be the input vector of player (node) i, i.e., the characteristic function of its neighborhood $N(i)$. A one-round algorithm for TWINS consists in each node sending the message $m_i = FP(x_i, t)$. The referee outputs 1 if

and only if $m_i = m_j$ for some pair $i \neq j$. A one-round algorithm for $\textsc{Twins}_x$ sends the same messages, but this time the referee checks whether $m_x = m_i$ for some $i \neq x$.

The algorithm for Translated-Twins on n-node graphs is slightly different. If a node $i \leq n/2$ has a neighbor $j > n/2$, it sends a special "no" message specifying that it cannot be a candidate for having a translated twin. Otherwise, let y_i^1 be the $n/2$-bit vector formed by the $n/2$ first bits of x_i. Thus, y_i^1 is the characteristic vector of $N(i) \cap \{1, \ldots, n/2\}$. Player i sends the message $m_i = FP(y_i^1, t)$. Symmetrically, for nodes labelled $i > n/2$, if i has some neighbor $j \leq n/2$, it sends the "no" message. Otherwise, let y_i^2 be the $n/2$-bit vector formed by the last $n/2$ bits of x_i. Hence, y_i^2 corresponds to $N(i) \cap \{n/2, \ldots, n\}$, "translated" by $-n/2$. Player i sends the message $m_i = FP(y_i^2, t)$. Then, the referee returns 1 if $m_i = m_{i+n/2}$ for some $i \leq n/2$.

Clearly, for Twins (resp. $\textsc{Twin}_x$, Translated-Twins), if the input graph is a yes-instance, then the algorithm outputs 1. The probability that Twins answers 1 on a no-instance is the probability that $FP(m_i, t) = FP(m_j, t)$ for two nodes i and j with different neighborhoods. For each fixed pair of nodes this probability is at most $1/n^{c+2}$, so altogether the probability of a wrong answer is at most $1/n^c$. With similar arguments for $\textsc{Twins}_x$ and Translated-Twins the probability of a wrong answer is at most $1/n^{c+1}$, since the referee makes n tests and each may be a false positive with probability at most $1/n^{c+2}$.

For GraphEq on graphs of size $2n$, each node $i \in [n]$ calls y_i the first n coordinates of x_i, and each node $i \in [2n] \setminus [n]$, calls y_i the last n coordinates of x_i. Then, each node i communicates $m_i = FP(y_i, t)$, the fingerprint of y_i produced using t. The referee then checks whether $m_i = m_{i+n}$, for all $i \in [n]$. Again, the probability of a wrong answer is at most $1/n^{c+1}$, since the referee makes n tests and each may be a false positive with probability at most $1/n^{c+2}$.

For $\textsc{Twins}_x$ with private coins, each node i sends a bit stating if it sees x, a number t_i chosen uniformly at random in the interval $n^{c+2} < p \leq 2n^{c+2}$ and also $FP(x_i, t_i)$. The referee retrieves the neighborhood of node x (which was sent bit by bit by all the others) and then, for each $i \neq x$, it constructs $FP(x_x, t_i)$ and compares it to $FP(x_i, t_i)$. If the values are equal for some i, the referee outputs 1, otherwise it outputs 0. Again, any yes-instance will answer 1, and the probability that a no-instance (wrongly) answers 1 is at most $1/n^c$. □

The fact that TRANSLATED-TWINS requires $\Omega(\sqrt{n})$ bits per node for any one-round private coin, ϵ-error algorithm follows directly from Lemma 3 and Proposition 2.

Theorem 5. *For any $\epsilon < 1/2$, $C_\epsilon^{\text{priv}}(\text{TRANSLATED-TWINS}) = \Omega(\sqrt{n})$.*

Remark 4. *Theorems 4 and 5 show that problem* TRANSLATED-TWINS *separates private-coin from public-coin algorithms.*

In order to complete the Table 1, it only remains to prove that problems TWINS_x and TRANSLATED-TWINS can be solved with one-round private coin algorithms using $\mathcal{O}(\sqrt{n}\log n)$ bits.

Theorem 6. *For any $c > 0$, there is a one-round private coin algorithm for* TWINS *and* TRANSLATED-TWINS *using messages of size $\mathcal{O}(\sqrt{n}\log n)$ and having $1/n^c$ one-sided error.*

Proof Babai and Kimmel in [14] propose, for solving problem EQ in the simultaneous 2-player model, a private coin algorithm with 1/3 one sided error and $\mathcal{O}(\sqrt{n})$ message size complexity (see Proposition 1). Let us call this algorithm $\mathcal{P}_0$. As the authors point out, this algorithm is symmetrical, in the sense that both players compute the same function on their own input. We can refer to any of these two indistiguishable player as Alice. We define the algorithm $\mathcal{P}$ as one obtained by simulating $(c+2)\log_3 n$ calls to algorithm $\mathcal{P}_0$. More formally, in $\mathcal{P}$ each player creates $(c+2)\log_3 n$ times the message that it would create in $\mathcal{P}_0$, using at each time independent tosses of private coins. The referee answers 1 if and only if the referee of $\mathcal{P}_0$ would have answered 1 on each of the $(c+2)\log_3 n$ pairs of messages. Therefore, $\mathcal{P}$ is a private coin algorithm for EQ with one sided error smaller than $1/n^{c+2}$, and message size $\mathcal{O}(\sqrt{n}\log n)$.

The one sided private-coin algorithm $\mathcal{P}'$ for TWINS is the one where each node plays the role of Alice in $\mathcal{P}$ taking as input the characteristic function of its neighborhood, and then the referee simulates the role of the referee in $\mathcal{P}$ for each pair of messages. Similarily, an algorithm $\mathcal{P}''$ for TRANSLATED-TWINS works as follows: each node i sends "*no*" in the same cases described in the proof of Theorem 4, and otherwise it simulates the role of Alice on input y_i^1 formed by the first $n/2$ bits of x_i, if $i \leq n/2$ or on input y_i^2 formed by the $n/2$ last bits of x_i if $i > n/2$, where x_i is the characteristic function

of $N(i)$. The referee then simulates the behavior of the referee of $\mathcal{P}$ on the messages sent by i and $i+n$ (as long as none of them is "*no*").

Since $\mathcal{P}$ has just one sided error, if TWINS (resp. TRANSLATED-TWINS) is *true*, $\mathcal{P}'$ (resp. $\mathcal{P}''$) will always accept. On the other hand, if TWINS (resp. TRANSLATED-TWINS) is *false*, then the probability that $\mathcal{P}'$ (resp. $\mathcal{P}''$) accepts is the probability that $\mathcal{P}$ accepts in at least one pair of vertices, and then the error of $\mathcal{P}'$ (resp. $\mathcal{P}''$) is at most n^2 times (resp. n times) the error of $\mathcal{P}$. We obtain that $\mathcal{P}'$ and $\mathcal{P}''$ have at most $1/n^c$ one sided error, and message size complexity $\mathcal{O}(\sqrt{n}\log n)$. □

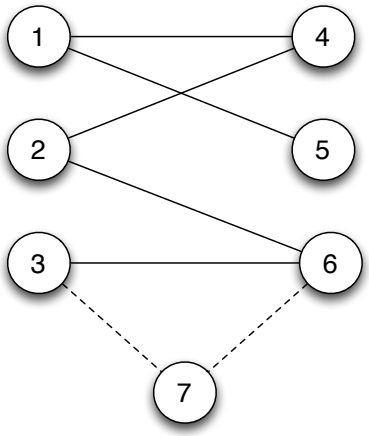
4. Hard problems for public coin algorithms

Consider the boolean function TRIANGLE(G) that outputs 1 if and only if G has a triangle, and the function DIAM3(G), that outputs 1 if and only if G has diameter at most 3. In [11] it is shown that the one-round deterministic message size complexity of these problems are lower-bounded by $\Omega(n)$, using a reduction from RECONSTRUCTION. However, as seen in Theorem 3, a reduction from RECONSTRUCTION does not imply lower bounds on the message size of randomized algorithms.

In the following theorem, we extend the techniques of [11], by constructing a reduction from INDEX to TRIANGLE(G) and DIAM3(G), showing that the message size complexity of these problems are also $\Omega(n)$, even in the public coin setting. We will show that we can push further the lower bounds for TRIANGLE(G) and DIAM3(G), showing that almost the same reduction in [11] can be used to decide INDEX, and therefore the message sizes of one-round randomized algorithms for these problems are also lower bounded by $\Omega(n)$. This explains why, in order to separate the models, we were forced to find new problems (such as TWINS).

Theorem 7. *For any $\epsilon < 1/2$, any one-round public coin algorithm computing* TRIANGLE(G) *(resp.* DIAM3(G)*) with ϵ two-sided error uses messages of size $\Omega(n)$.*

Proof Consider the INDEX function in the simultaneous 2-player model: the first player, say Alice, has as input an m-bit boolean vector x and the second player, Bob, has an integer $q, 1 \leq i \leq m$. INDEX(x, q) is defined as x_q, the qth coordinate of Alice's vector. We use the fact that for any $\epsilon < 1/2$, any



$$x = (1, 1, 0, 1, 0, 1, 0, 0, 1)$$

$$q = 9$$

Figure 4: An illustration of $H_x(3, 6)$ when $x = (1, 1, 0, 1, 0, 1, 0, 0, 1)$ and $q = 9$.

public-coin algorithm for INDEX requires $\Omega(m)$ bits (see, e.g., [26, 27] for a proof). We may assume, w.l.o.g., that $m = n^2$.

In [11], Becker *et al.* show that the deterministic message size complexity of TRIANGLE and DIAM3 is $\Theta(n)$. The idea of the proof is the following: if there is an algorithm $\mathcal{P}$ of message size $c(n)$ for TRIANGLE or DIAM3, then there is an algorithm for RECONSTRUCTION in bipartite graphs of cost $2c(n)$. We slightly modify that proof in order to obtain a reduction from INDEX.

Let $\epsilon < 1/2$, and $\mathcal{P}$ be an ϵ-error public coin algorithm for TRIANGLE on n-nodes graphs, using $c(n)$ bits. We give an algorithm for INDEX using $2n \cdot c(2n + 1)$ bits.

Let x be an $m = n^2$-bit vector. Let H_x be the bipartite graph with vertex set $\{1, \ldots, 2n\}$, such that for any $1 \leq k, l \leq n$, if $x_{(k-1)n+l} = 1$ then H_x has an edge between nodes k and $l + n$ (see Figure 4). Consider the family of graphs $H_x(i, j)$ obtained from H_x by adding a node $2n + 1$ whose neighbors are nodes i and $j + n$ (for any $1 \leq i, j \leq n$). Observe that $H_x(i, j)$ has a triangle if and only if $x_{(i-1)n+j} = 1$, in which case the triangle is formed by the nodes $\{i, j+n, 2n+1\}$. To simplify the notation we also define the graph $H_x(0, 0)$ obtained from H_x by adding an isolated node $2n + 1$.

The algorithm for INDEX is as follows. Bob sends its input q, which only has $\mathcal{O}(\log n)$ bits. Alice constructs the family of graphs $H_x(i, j)$, for all pairs $1 \leq i, j \leq n$ and for $(i, j) = (0, 0)$. Any node $k \leq 2n$ has exactly two possible neighborhoods, depending on whether it is adjacent to $2n + 1$ or not. For each $k \leq 2n$, Alice creates the message $m^+(k)$ that the algorithm

for TRIANGLE would send for node k in the graph $H_x(k,1)$ (if $k \leq n$) or in the graph $H(1,k-n)$ (if $k > n$). It also creates the message $m^-(k)$ that TRIANGLE would construct for node k in the graph $H_x(0,0)$. In full words, $m^-(k)$ corresponds to the case when the neighborhood of k is the same as in H_x, and $m^+(k)$ to the case when this neighborhood is the neighborhood in H_x, plus node $2n+1$. Then Alice sends, for each k, $1 \leq k \leq 2n$, the pair of messages $(m^-(k), m^+(k))$. Therefore, Alice uses $2n \cdot c(2n+1)$ bits.

It remains to explain how the referee retrieves the bit x_q. Let i, j be such that $q = (i-1)n + j$. Observe that $x_q = 1$ if and only if graph $H_x(i,j)$ has a triangle, therefore the referee must simulate the behavior of the referee for TRIANGLE on $H_x(i,j)$. For this purpose, the referee computes the message that node $2n+1$ would have sent on this graph (it only depends on i and j) and observes that algorithm $\mathcal{P}$ on $H_x(i,j)$ would have sent message $m^+(i)$, $m^+(j+n)$ and $m^-(k)$ for any $k \leq 2n$ different from i and j. Therefore, the referee can give the same output as $\mathcal{P}$ on $H_x(i,j)$. That is, it outputs x_q. The algorithm for INDEX will have ϵ error and will use $2n \cdot c(2n+1)$ bits. Thus, $\mathcal{P}$ requires $\Omega(n)$ bits.

The proof for DIAM3 is based on a similar reduction. Let $D_x(i,j)$ be the graph obtained from H_x by adding three nodes: node $2n+1$ connected to all nodes $k \leq 2n$, node $2n+2$ connected to node i and node $2n+3$ connected to $j+n$. Graph $D_x(0,0)$ is similar with the difference that nodes $2n+2$ and $2n+3$ are isolated. Observe (see also [11]) that $D_x(i,j)$ has diameter 3 if and only if $x_{(i-1)n+j} = 1$. The rest of the proof follows as before, swiching the roles of $H_x(0,0)$ for $D_x(0,0)$, and $H_x(i,j)$ for $D_x(i,j)$, for all $1 \leq i, j \leq n$. □

5. Solving Twins deterministically in the unicast congested clique model

In this section we show that, when we remove the restriction of one-round algorithms,we can solve TWINS in $o(n)$ rounds with bandwidth $\mathcal{O}(\log n)$ in the unicast congested clique model. Let G be a graph, and let A be the adjacency matrix of G. Call A^2 the matrix multiplication of A with itself, over the ring of integer square matrices of dimension n.

Lemma 4. *Let u and v be two vertices of G such that $deg(u) = deg(v) = d$. Then u and v are twins if and only if $A^2_{uv} = d$.*

Proof Observe that $deg(u) = \sum_{w\in[n]} a_{uw} = \sum_{w\in[n]} a_{wu}$. Suppose that u and v are twins, then $a_{uw} = a_{wv}$ for every $w \in [n]$. Therefore,

$$A^2_{uv} = \sum_{w\in[n]} a_{uw}a_{wv} = \sum_{w\in[n]} a_{uw} = d.$$

Conversely, suppose that

$$A^2_{uv} = \sum_{w\in[n]} a_{uw}a_{wv} = d = \sum_{w\in[n]} a_{uw}$$

it means that $a_{wv} = a_{uw}$ each time that $a_{uw} = 1$. Since $deg(u) = deg(v)$, necessarily u and v are twins. □

Consider the following problem, that we call Matrix-Multiplication, which is studied in [12]. The input are two square matrices S and T of dimension n, distributed over n players, labeled from 1 to n. Player labeled i receives as input the i-th row of S and the i-th row of T, and wants to compute the i-th row of matrix $P = ST$.

Proposition 3. *[12] There is a deterministic algorithm in the unicast congested clique model that solves* Matrix-Multiplication *in $\mathcal{O}(n^{1-2/\omega}\log n)$ rounds, where ω is the matrix multiplication exponent.*

Combining Lemma 4 and Proposition 3 we obtain the following algorithm for Twins. First, each vertex broadcasts its degree. Then, each vertex simulates the Matrix-Multiplication algorithm of Proposition 3 taking S and T equal to the adjacency matrix of G. Finally, each vertex verifies if it has some twin (by just looking for a vertex satisfying the condition of Lemma 4). Each vertex broadcasts the answer (one bit). Finally, every vertex accepts if at least one pair of twins was detected. We deduce the following result.

Theorem 8. *There is a deterministic algorithm in the unicast congested clique model that solves* Twins *with $\mathcal{O}(n^{1-2/\omega}\log)$ rounds and bandwidth $\mathcal{O}(\log n)$, where ω is the matrix multiplication exponent.*

The upperbound on the round complexity of previous algorithm depends on a constant ω known as the *matrix multiplication exponent*. This constant corresponds roughly to the exponent of n in the running time of the best

sequential algorithm performing matrix multiplication of square matrices of dimension n over a ring. Current best upper bound is $\omega < 2.3728639$ due to Le Gall [28]. This implies that the number of rounds of the algorithm of Proposition 3 is $\mathcal{O}(n^{0.158} \log n)$. For the same reason the algorithm of Theorem 8 solves TWINS in $\mathcal{O}(n^{0.158} \log n)$ rounds.

6. Open problems

The future challenge is to determine the one-round message size complexity of TWINS for private coin algorithms. Techniques for proving lower bounds in the similar problem EXISTSEQ, defined in the simultaneous n-player model, can not be used directly: in the broadcast congested clique model nodes share information. This subtle difference makes the problem of finding lower bounds elusive. This seems to be the case of TWINS. More generally, as stated by Fischer, Oshman an Zwick [16], "the power of private randomness in this model remains a fascinating open question".

Another interesting problem is CONNECTIVITY. Its one-round message size complexity is also open. Recall that, in the randomized, public coins setting, there exists an algorithm that uses $\mathcal{O}(\log^3 n)$ bits, due to Ahn, Guha and McGregor [21]. Can this upper bound be improved to $\mathcal{O}(\log n)$? Is it possible to find, in the private coin case and/or in the deterministic case, better lower bounds?

Aknowledgements

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

References

[1] Z. Lotker, B. Patt-Shamir, E. Pavlov, D. Peleg, Minimum-weight spanning tree construction in $O(\log \log n)$ communication rounds, SIAM Journal on Computing 35 (1) (2005) 120–131.

[2] J. Dean, S. Ghemawat, Mapreduce: simplified data processing on large clusters, Communications of the ACM 51 (1) (2008) 107–113.

[3] G. Malewicz, M. H. Austern, A. J. Bik, J. C. Dehnert, I. Horn, N. Leiser, G. 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.

[4] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, I. Stoica, Spark: Cluster computing with working sets., HotCloud 10 (10-10) (2010) 95.

[5] T. White, Hadoop: The definitive guide, " O'Reilly Media, Inc.", 2012.

[6] M. Isard, M. Budiu, Y. Yu, A. Birrell, D. Fetterly, Dryad: distributed data-parallel programs from sequential building blocks, in: ACM SIGOPS operating systems review, Vol. 41, ACM, 2007, pp. 59–72.

[7] A. Drucker, F. Kuhn, R. Oshman, The communication complexity of distributed task allocation, in: Proc. of the 2012 ACM Symposium on Principles of Distributed Computing, PODC '12, 2012, pp. 67–76.

[8] A. Drucker, F. Kuhn, R. Oshman, On the power of the congested clique model, in: Proc. of the 2014 ACM Symposium on Principles of Distributed Computing, PODC '14, 2014, pp. 367–376.

[9] E. Kushilevitz, N. Nisan, Communication Complexity, Cambridge University Press, New York, NY, USA, 1997.

[10] A. Chakrabarti, S. Yaoyun, A. Wirth, A. Yao, Informational complexity and the direct sum problem for simultaneous message complexity, in: Proc. of the 42nd IEEE Symposium on Foundations of Computer Science, FOCS '01, 2001, pp. 270–278.

[11] F. Becker, M. Matamala, N. Nisse, I. Rapaport, K. Suchan, I. Todinca, Adding a referee to an interconnection network: What can(not) be computed in one round, in: Proc. of the 25th IEEE International Parallel and Distributed Processing Symposium, IPDPS '11, 2011, pp. 508–514.

[12] K. Censor-Hillel, P. Kaski, J. H. Korhonen, C. Lenzen, A. Paz, J. Suomela, Algebraic methods in the congested clique, in: Proc. of the 2015 ACM Symposium on Principles of Distributed Computing, PODC '15, 2015, pp. 143–152.

[13] A. C.-C. Yao, Some complexity questions related to distributive computing (preliminary report), in: Proc. of the 11th ACM Symposium on Theory of Computing, STOC '79, 1979, pp. 209–213.

[14] L. Babai, P. G. Kimmel, Randomized simultaneous messages: Solution of a problem of Yao in communication complexity, in: Proc. of the 12th IEEE Conference on Computational Complexity, 1997, pp. 239–246.

[15] I. Newman, M. Szegedy, Public vs. private coin flips in one round communication games, in: Proc. of the 28th ACM Symposium on Theory of Computing, STOC '09, 1996, pp. 561–570.

[16] O. Fischer, R. Oshman, U. Zwick, Public vs. private randomness in simultaneous multi-party communication complexity, in: Proc. of the International Colloquium on Structural Information and Communication Complexity, SIROCCO '16, Vol. 9988 of Lecture Notes in Computer Science, 2016, pp. 60–74.

[17] J. Nešetřil, E. Milková, H. Nešetřilová, Otakar Boruvka on minimum spanning tree problem: Translation of both the 1926 papers, comments, history, Discrete Mathematics 233 (1-3) (2001) 3–36.

[18] P. Montealegre, I. Todinca, Brief announcement: Deterministic graph connectivity in the broadcast congested clique, in: Proc. of the 2016 ACM Symposium on Principles of Distributed Computing, PODC '16, 2016, pp. 245–247.

[19] T. Jurdzínski, K. Nowicki, Brief announcement: On connectivity in the broadcast congested clique, in: Proc. of the 31st International Symposium on Distributed Computing, DISC '17, Vol. 91 of LIPIcs, 2017, pp. 1868–1869.

[20] S. Pai, S. V. Pemmaraju, Connectivity lower bounds in broadcast congested clique, arXiv:1905.09016, 2019.

[21] K. J. Ahn, S. Guha, A. McGregor, Analyzing graph structure via linear measurements, in: Proc. of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '12, 2012, pp. 459–467.

[22] K. J. Ahn, S. Guha, A. McGregor, Graph sketches: Sparsification, spanners, and subgraphs, in: Proc. of the 31st Symposium on Principles of Database Systems, PODS '12, 2012, pp. 5–14.

[23] F. Becker, A. Kosowski, M. Matamala, N. Nisse, I. Rapaport, K. Suchan, I. Todinca, Allowing each node to communicate only once in a distributed system: shared whiteboard models (2015).

[24] P. Montealegre, S. Perez-Salazar, I. Rapaport, I. Todinca, Two rounds are enough for reconstructing any graph (class) in the congested clique model, in: Proc. pf the International Colloquium on Structural Information and Communication Complexity, SIROCCO '18, Vol. 11085 of Lecture Notes in Computer Science, 2018, pp. 134–148.

[25] E. Kushilevitz, Communication complexity, Advances in Computers 44 (1997) 331–360.

[26] I. Kremer, N. Nisan, D. Ron, On randomized one-round communication complexity, Computational Complexity 8 (1) (1999) 21–49.

[27] I. Kremer, N. Nisan, D. Ron, Errata for: "On randomized one-round communication complexity", Computational Complexity 10 (4) (2001) 314–315.

[28] F. Le Gall, Powers of tensors and fast matrix multiplication, in: Proc. of the 39th International Symposium on Symbolic and Algebraic Computation, ISSAC '14, 2014, pp. 296–303.