

Networks Cannot Compute Their Diameter in Sublinear Time

PRELIMINARY VERSION - PLEASE CHECK FOR UPDATES

Silvio Frischknecht Stephan Holzer Roger Wattenhofer
{fsilvio, stholzer, wattenhofer}@tik.ee.ethz.ch
Computer Engineering and Networks Laboratory (TIK), ETH Zurich, Switzerland

September 13, 2011

Abstract

We study the problem of computing the diameter of a network in a distributed way. The model of distributed computation we consider is: in each synchronous round, each node can transmit a different (but short) message to each of its neighbors. We provide an $\tilde{\Omega}(n)$ lower bound for the number of communication rounds needed, where n denotes the number of nodes in the network. This lower bound is valid even if the diameter of the network is a small constant. We also show that a $(3/2 - \varepsilon)$ -approximation of the diameter requires $\tilde{\Omega}(\sqrt{n})$ rounds. Furthermore we use our new technique to prove an $\tilde{\Omega}(\sqrt{n})$ lower bound on approximating the girth of a graph by a factor $2 - \varepsilon$.

Contact author:

Stephan Holzer, ETZ G64.2, ETH Zurich, CH-8092 Zurich, Switzerland, +41 446327065

1 Introduction

Without doubt, a fundamental property of a network is its diameter. In distributed computing, the network diameter plays a prominent role as the boundary between so-called local and global problems. In distributed complexity, some problems are *global* (“difficult”), in the sense that far-apart nodes must communicate with each other in order to solve the problem. Typical examples of such global problems include counting the total number of nodes in the network, or constructing a spanning tree. Global problems need at least time $\Omega(D)$, where D is the diameter of the network. Then again, many of these global problems *can* be computed in $O(D)$ time, usually by a simple flooding/echo procedure. If a problem is not global, it is *local*, as it can be computed in time independent of the diameter. Typical examples of local problems include combinatorial graph problems such as matching or vertex cover.¹

Computing the diameter of a network is surely a global problem as one cannot hope to compute the diameter D in $o(D)$ time. However, can the nodes of a network find its diameter in $O(D)$ time? In this paper, we answer this question negatively, by presenting a lower bound of $\Omega(n)$, where n is the number of nodes of the network. Our lower bound can be proved by means of communication complexity, and even holds for networks of constant diameter. Moreover, we show that a $(3/2 - \varepsilon)$ -approximation of the diameter, and a $(2 - \varepsilon)$ -approximation of the girth also have a $\Omega(\sqrt{n})$ distributed complexity.

All our bounds hold in the so-called congest model, where each node can send a different but short message to each of its neighbors in each round. Since our lower bounds are substantial, we hope that some of our techniques might be of use also in a non-distributed (sequential) setting. Some of the best sequential techniques for computing the diameter use fast matrix multiplication, resulting in time $o(n \cdot m)$. In contrast, we present a lower bound of $\Omega(n)$ in a system that is m -parallel: a speedup of m is possible due to communicating over m edges of the network at the same time. This implies that the diameter problem does not allow for a high parallel speedup of sequential algorithms.

Usually the fastest known algorithms (e.g. [2, 3, 25]) use the fast matrix multiplication algorithm by Coppersmith and Winograd [5] and run in time $O(n^{2.376})$. For sparse graphs well thought specialized algorithms such as [4] will be faster. In the light of all this progress that was made on the upper bounds it is somehow surprising that almost nothing is known about lower bounds for this problem (even for any model of computation). To the best of our knowledge we are the first to give nontrivial lower bounds for computing the diameter of a graph. These bounds will be valid in a distributed setting.

2 Model and Basic Definitions

Model: As a model of computation we consider a synchronized network of processors represented by an undirected graph $G = (V, E)$. Nodes V correspond to processors and edges E correspond to connections between the processors over which they can communicate. We denote the number of nodes of a graph by n and the number of its edges by m . Each processor has unbounded computational power and initially has no knowledge of the nodes in the graph G other than itself and its immediate neighbors. We consider a round based model where every node can send B bits of information over all its edges in one round. Typically one sets $B = O(\log n)$, which is the number of bits needed to encode an ID of a node of a network (we assume IDs to be in a range polynomial in the network size). In this case of $B = O(\log n)$ the model is just called CONGEST. We are interested in the minimal number of communication-rounds that are needed until some problem is solved. Therefor we assume that internal computation is free. This model is called CONGEST(B) model [20]. To be more formal, we are interested in evaluating a function $g : \{\text{all graphs over } n \text{ nodes}\} \rightarrow S$, where S is e.g. $\{0, 1\}$ or \mathbb{N} and define distributed round complexity as follows:

Definition 2.1. (*distributed round complexity*). Let \mathcal{A}_ε be the set distributed algorithms that use (public)

¹More accurately, researchers distinguish between strictly local and pseudo-local problems. Strictly local problems allow for constant-time distributed algorithms, whereas the distributed time complexity of pseudo-local problems may depend on the size of the network, but is independent of the diameter of the network. Finding a constant approximation of a dominating set in a planar graph is an elaborate example of a strictly local problem, whereas matching and vertex cover are pseudo-local.

randomness (denoted by *pub*) and when used by Alice and Bob, evaluate a function g on the underlying graph G over n nodes (representing the network) with an error probability smaller than ε . Denote by $R_\varepsilon^{dc}(A(G))$ the distributed round complexity (denoted by *dc*) representing the number rounds that an algorithm $A \in \mathcal{A}_\varepsilon$ needs in order to compute $g(G)$. We define

$$R_\varepsilon^{dc}(g) = \min_{A \in \mathcal{A}_\varepsilon} \max_{G=(V,E):|V|=n} R^{cc-pub}(A(G))$$

to be the smallest amount of rounds any algorithm needs to send in order to compute g .

The problems we will consider are computing the diameter and the girth of a graph as well as approximations to them. A set $\{0, \dots, k\}$ is usually denoted by $[k]$.

Definition 2.2. (*distance, diameter*). Let $G = (V, E)$ be a graph and $u, v \in V$ any two nodes in G . The distance $d(u, v)$ between u and v is the length of a shortest path between u and v that consists of edges of G . The diameter $d(G) := \max_{u, v \in V} d(u, v)$ of a graph G is the maximum distance between any two nodes of the graph.

Definition 2.3. (*girth*). The girth of a graph G is the length of the shortest cycle in G . (If G is a forest its girth is infinity.)

Definition 2.4. (*approximation*). Given an optimization problem P over graphs, denote by $OPT_P(G)$ the optimal solution for P on G and by $A(G)$ the solution of an algorithm A for P on G . We say A is ρ -approximative for P if $OPT_P(G) \leq A(G) \leq \rho \cdot OPT_P(G)$ for any graph G .

To obtain our lower bounds we need knowledge on basics of communication complexity that was first introduced by Yao [27]. Here, two computationally unbounded parties Alice and Bob each receive a k -bit string $a \in \{0, 1\}^k$ and $b \in \{0, 1\}^k$ respectively. Alice and Bob can communicate with each other one bit at a time and want to evaluate a function $g : \{0, 1\}^k \times \{0, 1\}^k \rightarrow \{0, 1\}$ on their input. We assume that Alice and Bob have access to public randomness for their computation and we are interested in the number of bits that Alice and Bob need to exchange in order to compute g .

Definition 2.5. (*communication complexity*). Let \mathcal{A}_ε be the set of two-party algorithms that use public randomness (denoted by *pub*) and when used by Alice and Bob, compute g on any input a and b with an error probability smaller than ε . Denote by $R_\varepsilon^{cc-pub}(A(a, b))$ the communication complexity (denoted by *cc*) representing the number of 1-bit messages sent between Alice and Bob while executing an algorithm $A \in \mathcal{A}_\varepsilon$ to compute $g(a, b)$. We define

$$R_\varepsilon^{cc-pub}(g) = \min_{A \in \mathcal{A}_\varepsilon} \max_{a, b \in \{0, 1\}^k} R^{cc-pub}(A(a, b))$$

to be the smallest amount of bits any algorithm needs to send in order to compute g .

A well studied problem in communication complexity is that of set disjointness, where we are given two subsets of $[k]$ and need to decide whether they are disjoint. Here, the strings a and b will indicate membership of elements to each of these sets.

Definition 2.6. (*disjointness problem*). The set disjointness function $\text{disj}_k : \{0, 1\}^k \times \{0, 1\}^k \rightarrow \{0, 1\}$ is defined as follows.

$$\text{disj}_k(a, b) = \begin{cases} 0 & \text{if there is an } i \in [k] \text{ such that } a(i) = b(i) = 1 \\ 1 & \text{otherwise} \end{cases}$$

where $a(i)$ and $b(i)$ are the i -th bit of a and b respectively (indicating whether an element is a member of the corresponding set.)

We will use the following basic theorem that was proven in Example 3.22 in [16].

Theorem 2.7. For any sufficiently small $\varepsilon > 0$ we can bound $R_\varepsilon^{cc-pub}(\text{disj}_k)$ by $\Omega(k)$

3 Our Contribution and Related Work

In this paper we show that even graphs with small diameter require $\Omega(n/B)$ rounds in the congest model to calculate the exact diameter. In section 6 we also show that even an approximation that is closer than a factor of $\frac{3}{2}$ to the diameter is impossible to obtain by using any algorithm that does not use $\Omega(\sqrt{n}/B)$ rounds. This non-approximability result is already interesting by itself but even more due to an ingenious sequential algorithm by Aingworth, Chekuri and Motwani [1]. They provided an $\frac{3}{2}$ -approximation in time $O(m \cdot \sqrt{n \log n} + n^2 \cdot \log n)$ which would imply an $O(\sqrt{n \log n} + \frac{n^2 \cdot \log n}{m})$ -algorithm if it could be fully parallelized. Here we refer to the parallelism that comes from the fact that one can communicate via all m edges in one time slot which can imply stronger speedup than having n processors. An example for such a maximal speedup is computing a BFS-tree starting in some node v on a constant-diameter graph where each node has a unique identifier. This takes sequential time $\Theta(n + m)$ but time $O(1)$ in our distributed model. It might be interesting whether our lower bound can be extended to a lower bound in the sequential model.

Finally we will also show a $\Omega(\sqrt{n}/B)$ lower bound for approximations better than 2 of the girth. Opposed to this the diameter can be 2-approximated in time $O(D)$ by performing a breadth-first search and taking the depth of the resulting tree as an estimate.

The technique we develop to prove our lower bounds is mainly inspired by the connection between communication complexity and distributed computing in [24]. The first paper that introduced a technique to apply lower bounds for communication complexity in a distributed setting is [21] that proved an $\tilde{\Omega}(\sqrt{n}/B + D)$ lower bound² on computing a minimum spanning tree (MST). Later [6] improved their technique by using a modified graph to yield approximation lower bounds for MST. In [24] these bounds were improved further and extended to a long list of problems including non-approximability results. To achieve this they used the graph from [6] but a new technique how to apply lower bounds from communication complexity. In this paper we introduce new graphs based on a new technique and new proofs adapted to these graphs on how to adapt the communication complexity lower bounds. One main difference to the previous results is that our graphs are able to yield $\Omega(n/B)$ lower bounds.

Observe that the bounds we derive can not be obtained by just extending the technique of [24] but require a new approach. To explain why this is the case we summarize the key-ideas of [24]: The first step in [24] is to transfer an $\Omega(n)$ lower bound for the *set disjointness* problem (and the *equality* problem) in communication complexity to an $\tilde{\Omega}(\sqrt{n}/B + D)$ lower bound in distributed computing using a special graph. Then these graphs are modified to yield lower bounds for different kind of problems. Remark that all of the lower bounds proven in [24] are of the type $\tilde{\Omega}(\sqrt{n}/B + D)$ and a main question was whether the technique can be used to prove stronger lower bounds. For many of the problems considered in [24] this is not possible due to almost matching upper bounds. Usually the graphs of [24] are constructed such that any algorithm causes that (roughly) \sqrt{n} bits either need to travel through one short path (taking time (roughly) \sqrt{n}) or in parallel through (roughly) \sqrt{n} long paths of length (roughly) \sqrt{n} (taking time (roughly) \sqrt{n} as well). To prove the lower bounds of [24] these long paths are connected in a special way depending on the problem at hand and questions such as “Is there a minimum spanning tree of a certain weight?” asked for these graphs will heavily rely on these long paths and the above mentioned connections between them. And exactly this fact is the reason why we cannot just use these graphs to prove lower bounds on the diameter: to obtain a meaningful lower bound we always would need to assume that the diameter of our graphs is significantly smaller than \sqrt{n} and it is not clear how we could utilize these long paths of length (roughly) \sqrt{n} that are already longer than the diameter: What if we want to give the algorithm a hard time distinguishing whether the diameter is 8 or 9? A similar reasoning can be done for computing the girth. Also for the girth we are not aware of any nontrivial lower bounds that were proven before. Note that currently we do not see how our technique could be used to yield or improve the results of [24].

A further advantage regards the diameter of the graphs for which our lower bounds are valid. Although previous constructions are very thoughtful they often only work as long as the graph that yields the

²The set $\tilde{\Omega}(f(n))$ includes functions differing from $f(n)$ by up to a polylogarithmic factor, that is e.g. $f(n)/\log^2 n$

algorithm to perform bad has a diameter depending on n . E.g. in [24] the diameter is $\Theta(\log n)$ and the nice follow-up [19] also needs to use $\Theta(\log n)$ diameter-graphs for a range of their parameters. Although previous used graphs can be modified to have only a constant diameter, unfortunately this will be at the cost of getting weaker lower bounds, e.g. [6],[18],[24]. As mentioned in [8] it would be nice to have faster algorithms for graphs of low diameter. We show that there is no hope to compute the diameter fast even on small-diameter graphs: our lower bounds still work if the diameter of the graph is restricted to be less than five.

3.1 Further Related Work Regarding Lower Bounds in Distributed Computing

Although [7] lists lots of lower bounds in distributed computing, only few are known in our model and most of them are already mentioned above and use similar techniques. A further bound that used techniques similar to those in [24] is the unconditional $\Omega(\sqrt{lD} + D)$ lower bound on computing random walks of length l in diameter D graphs by [19]. In [19] the authors were able to provide strong lower bounds that even depend on the diameter itself. One of the further rare bounds in our model is [13] and regards MST verification. They demonstrate why this needs $\Omega(\sqrt{n} + D)$ time.

3.2 Regarding the Diameter and All Pairs Shortest Paths in Sequential Models

One classical approach to compute the diameter is taught in many lectures: perform a breath first search (BFS) from each nodes in the graph - the depth of the deepest such BFS-tree will be the diameter. This takes time $O(n^2 + n \cdot m)$ in most sequential models of computing. In the distributed model considered in this paper, this approach (if not modified) will take time $O(n \cdot D)$ since each BFS-search takes $O(D)$ time.

Due to its importance and close connection to the all pairs shortest path problem (APSP), much effort was spent to obtain fast (sequential) algorithms for various versions of diameter and APSP, e.g. [2, 3, 25, 5, 4]. Although we already mentioned that no lower bounds are known for computing the diameter, at least few are known for the all pairs shortest path problem (APSP). In some sense they are related to our problem as described in the related work section. These are of interest as well since all algorithms to compute the diameter (that we are aware of) solve (at least implicitly) the all pairs shortest paths problem first and compute the diameter from the result. Furthermore lower bounds for the diameter immediately imply lower bounds for the APSP problem. However, it seems that there are only lower bounds for special classes of algorithms. Especially in our model no lower bounds are known for APSP and one has to be careful about how to define the problem in this model – e.g. whether each node should know the n^2 distances between any pair of nodes or only the n distances between itself and any other node.

Regarding known lower bounds for APSP in sequential models, one of the first lower bounds was provided by Kerr in [12] who showed that any oblivious APSP algorithm need to perform $\Omega(n^3)$ comparisons. Later Graham, Yao, and Yao [9] showed that if one wants to use an information-theoretic argument, it will be nontrivial to prove an $\omega(n^2)$ lower bound on the comparison-complexity of APSP, where additions are granted for free. A result by Karger et al. [11] shows that any APSP-algorithm (on directed graphs) that is based on path-comparison takes time $\Omega(n^3)$ on some graph of n^2 edges. In [22] Pettie shows that any algorithm needs $\Omega(m \cdot n + n^2 \cdot \log n)$ time to compute APSP on directed graphs if it uses an hierarchy-based approach as in [10, 23, 26]. By hierarchy-based approach we mean that one first computes some kind of hierarchy, and then solves n shortest paths single source problems using n independent processes. However, observe that the classes of algorithms are very limited since there are several algorithms known (including the one that uses fast matrix multiplication) that can beat any of these lower bounds (at least when m is getting large).

Note that all these lower bounds can easily be broken using fast matrix multiplication (at least when m is not too small). In contrast to this our lower bounds are valid for all algorithms in the model we consider.

4 Relating Distributed Complexity to two-party Communication Complexity

We want to show lower bounds on the distributed runtime of several graph-problems such as “what is the diameter of the underlying graph?”. In this paper we will always assume decision-versions of the problems, e.g. “Is the diameter larger than 4?” which will also imply lower bounds on the original problems. To achieve these lower bounds we will use reductions that transform two-party communication complexity lower bounds (on the number of bits that need to be sent between Alice and Bob) into lower bounds on the round complexity in distributed computing. In this section, we show how to derive a two-party communication version f' from any distributed graph-function f and provide a reduction from f' to f as well as relate their complexities. Later, in sections 5, 6 and 7 we will pick a “base”-function g (e.g. the disjoint set problem) that can be evaluated by two parties and a communication lower bound for this evaluation (e.g. theorem 2.7) to derive a lower bound for f' and thus for f . An overview of the whole reduction-procedure can be found in figure 4.

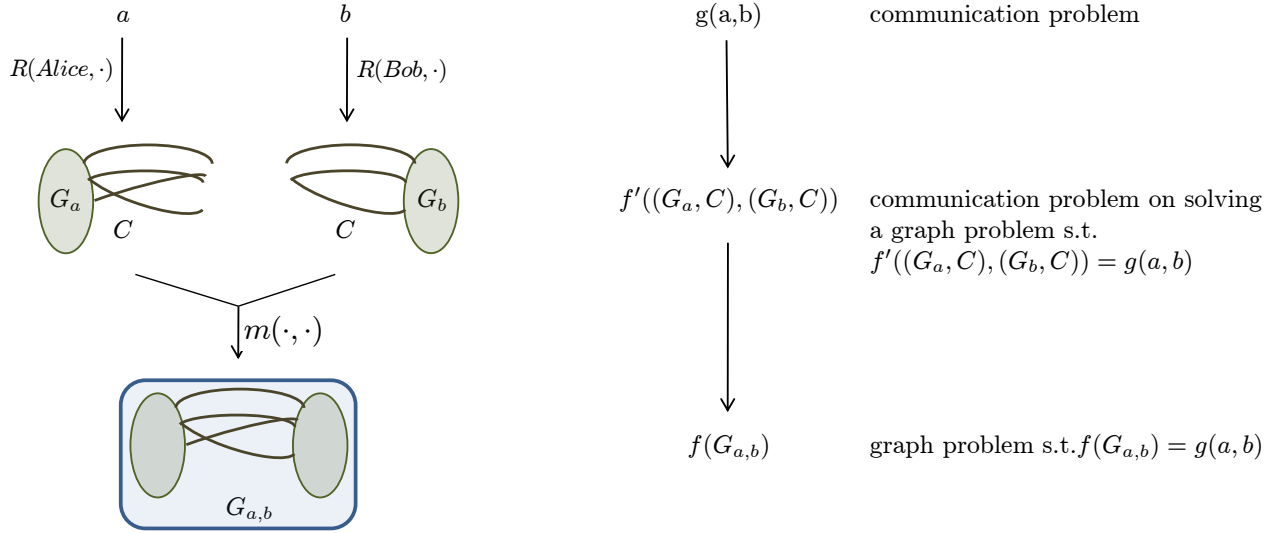


Figure 1: Upper part: This part depends on the problem at hand and will be described in the sections devoted to the problems we study. We reduce the communication problem of computing $g(a,b)$ into a communication problem of computing $f'((G_a, C_k), (G_b, C_k))$ where a part of a graph is given to Alice and part of it is given to Bob. According to the reduction \mathcal{R} , Alice will construct the part G_a (green, left side) from a and Bob G_b (green, right side) from b . Both will add edges C_k (brown) to G_a and G_b respectively. Lower part: This part is independent of f and shared by all of our reductions and described in this section. We reduce the two-party communication problem of computing $f'((G_a, C_k), (G_b, C_k))$ to the graph problem of computing $f(G_{a,b})$. The graph $G_{a,b}$ (blue) is constructed by connecting G_a to G_b using C_k .

First we need to show how to simulate distributed graph-algorithms in two-party communication. This will be simplified by introducing the notation of a cut.

Definition 4.1. (*cut*). Let $G = (V, E)$ be a graph. A cut (G_a, G_b, C_k) is a partition of G into two disjoint subgraphs $G_a = (V_a, E_a)$ and $G_b = (V_b, E_b)$ and a cut-set $C_k \subseteq E$ s.t. $V = V_a \dot{\cup} V_b$ and $E = E_a \dot{\cup} E_b \dot{\cup} C_k$. The cut-set C_k consists of edges whose endpoints are in different subsets of the partition.

Observe that now we can define a two-party communication problem f' according to the graph-problem

f in a canonical way: We define

$$f'((G_a, C_k), (G_b, C_k)) := f(G) \quad \text{for any graph } G \text{ and cut } (G_a, G_b, C_k) \text{ of } G$$

When computing $f'((G_a, C_k), (G_b, C_k))$ Alice will get input (G_a, C_k) and Bob input (G_b, C_k) . Now we formally show how f' can be reduced to f and analyze the complexity-relation in theorem 4.5.

Lemma 4.2. *The function f' can be reduced to f .*

Proof. Let $((G_a, C_k), (G_b, C_k))$ be an input to f' . Given a distributed algorithm A for f , Alice and Bob can use algorithm A to solve $f'((G_a, C_k), (G_b, C_k))$ in direct communication. We hence forth call $M^a(r)$ the set of messages sent by algorithm A over the edges in the cut-set C_k from nodes in V_a to nodes in V_b in the graph G (induced by the cut (G_a, G_b, C_k)) in round r . Conversely we call $M^b(r)$ the messages sent from V_b to V_a in round r . We will show how Alice simulates A on the nodes of $V_a \subseteq V_{a,b}$ of $G_{a,b}$ without knowing the state of the nodes V_b . Bob does the same for G_b . After each simulated round r of the algorithm A , Alice goes through the following steps:

1. Alice collects the messages $M^a(r)$ that nodes in V_a wanted to send over edges in C_k while executing A and sends $M^a(r)$ to Bob using their communication channel.
2. Alice receives $M^b(r)$ from Bob using their communication channel. Alice provides this information to the according simulated nodes in V_a .

Bob does the same with V_b and sends according messages to Alice. Note that using this scheme, Alice and Bob can simulate A on G in direct communication. \square

In the following sections we want to further reduce a “base-”function $g : \{0, 1\}^k \times \{0, 1\}^k \rightarrow \{0, 1\}$ to f' and we define:

Definition 4.3. *A c_k -reduction*

$$\mathcal{R} : \{\text{Alice}, \text{Bob}\} \times \{0, 1\}^k \rightarrow \left\{ \begin{array}{l} (G, C_k) : G \text{ is any graph and } C_k \text{ any subgraph of } G \text{ such that } (G, \cdot, C_k) \\ \text{is a cut of } G \text{ with } |C_k| = c_k \end{array} \right\}$$

is a function that transforms any g -inputs a, b into inputs for f' s.t. $g(a, b) = f'(\mathcal{R}(\text{Alice}, a), \mathcal{R}(\text{Bob}, b))$.

Observe that the size of C_k does not depend on a and b .

Remark 4.4. *Using a reduction as above we obtain $R_\varepsilon^{\text{cc-pub}}(g) = R_\varepsilon^{\text{cc-pub}}(f')$ since the reduction requires $g(a, b) = f'(\mathcal{R}(\text{Alice}, a), \mathcal{R}(\text{Bob}, b))$, this implies that the same number of bits need to be exchanged when computing $g(a, b)$ and $f'(\mathcal{R}(\text{Alice}, a), \mathcal{R}(\text{Bob}, b))$.*

So we will state the relation of the complexities of f' and f depending on c_k .

Theorem 4.5. *Let $B \geq 1$ and f be any function on graphs and f' the function derived from f as described above. And g a base-function that can be reduced to f' using a c_k -reduction. We can bound*

$$\frac{R_\varepsilon^{\text{cc-pub}}(g)}{2 \cdot c_k \cdot B} \leq R_\varepsilon^{\text{dc}}(f)$$

Proof. Given a graph G and cut (G_a, G_b, C_k) , where C_k is of size c_k . We know that $R_\varepsilon^{\text{cc-pub}}(f')$ is a lower bound on the number of bits that any distributed algorithm A must send over edges in C_k . Now these bits can not be encoded in less than $\frac{R_\varepsilon^{\text{cc-pub}}(f')}{B}$ messages in our distributed model and we conclude that $\frac{R_\varepsilon^{\text{cc-pub}}(f')}{B}$ messages need to be sent over cut C_k . In each round any algorithm can send at most $|C_k|$ messages via C_k into each direction and we obtain that $\frac{R_\varepsilon^{\text{cc-pub}}(f')}{2|C_k|B}$ is always a lower bound for $R_\varepsilon^{\text{dc}}(f)$. Observe that here we abstract away how algorithm A works in detail - its round complexity could be actually much higher than suggested by this bound. Due to remark 4.4 the statement follows. \square

We finish this section by defining a map that will be used in each lower-bound section.

Definition 4.6. *Denote by m a map that maps f' -inputs $((G_a, C_k), (G_b, C_k))$ to the graph $G_{a,b}$ that corresponds to the cut, that is $G_{a,b} := (V_{a,b}, E_{a,b})$, s.t. $V_{a,b} := V_a \cup V_b$ and $E_{a,b} := E_a \cup E_b \cup C_k$.*

5 Diameter Lower Bound

Theorem 5.1. *For any $n \geq 10$ and $B \geq 1$ and sufficiently small ε any distributed randomized ε -error algorithm A that computes the exact diameter of a graph requires at least $\Omega\left(\frac{n}{B}\right)$ time for some n -node graph even when the diameter is at most 5.*

Deciding whether a graph G has diameter less than 4 or not will be the decision-version $diam_4$ of the function $diam$, that is

$$diam_4(G) := \begin{cases} 1 & : diam(G) \leq 4 \\ 0 & : \text{else} \end{cases}$$

In order to prove theorem 5.1 we follow the high-level idea explained in section 4 and derive a function $diam'_4$ from $diam_4$ as described in section 4. To prove bounds depending on n , we choose the length k of the input to the base-function g to depend on n and set $k_n := \lfloor \frac{n}{10} \rfloor$. As base-function g we consider the $disj_{k_n}^2$ problem. Now we need to define a reduction \mathcal{R} that given inputs a and b to g maps $(Alice, a)$ and (Bob, b) to inputs $(G_a, C_{k_n}^2)$ and $(G_b, C_{k_n}^2)$ for $diam'_4$. During the reduction \mathcal{R} , Alice defines L and L' , Bob defines R and R' to be the following sets of nodes (as displayed in Figure 2)

$$\begin{aligned} L &= \{l_\nu | \nu \in [2k_n - 1]\} & R &= \{r_\nu | \nu \in [2k_n - 1]\} \\ L' &= \{l'_\nu | \nu \in [2k_n - 1]\} & R' &= \{r'_\nu | \nu \in [2k_n - 1]\} \end{aligned}$$

As a first step (that is independent of the input a) in constructing G_a Alice connects nodes l_ν with nodes

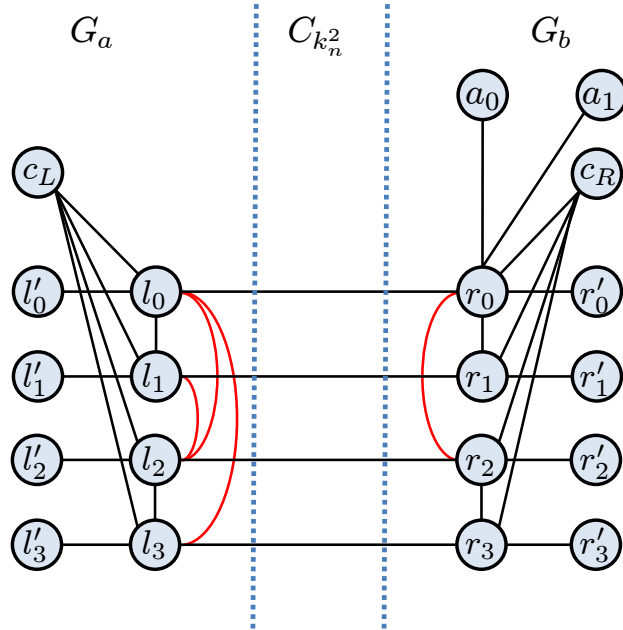


Figure 2: The above graph $G_{a,b}$ is for $n = 20$ (and thus we set $k_n = 2$) and results from inputs $a = (0, 0, 0, 1)$, $b = (0, 1, 1, 1)$ using the reduction \mathcal{R} . Accordingly the red edges represent a and b . To be more detailed, edge (l_0, l_2) represents $a(1) = 0$, edge (l_0, l_3) represents $a(2) = 0$, edge (l_1, l_2) represents $a(3) = 0$ and edge (r_0, r_2) represents $b(1) = 0$. This causes the diameter to be larger than 4. Using theorem 5.3 we conclude that a and b are not disjoint, which is indeed.

l'_ν for all $\nu \in [2k_n - 1]$. In the same way Bob connects r_ν with r'_ν to construct G_b . Furthermore Alice adds a node c_L to G_a that will be connected to all nodes in L and Bob adds a node c_R to G_b that will be connected to all nodes in R . Furthermore Alice adds edges between all nodes l_ν and l_μ where $\nu, \mu < k_n$, such that the subgraph induced by the upper nodes of L is a clique. Similarly Bob adds (clique-) edges

between all nodes l_ν and l_μ where $\nu, \mu \geq k_n$. Furthermore iff $a(i) = 1$ for each $i \in [k_n^2 - 1]$ Alice connects node $l_{i \bmod k_n}$ from the upper half to node $l_{k_n + \lfloor \frac{i}{k_n} \rfloor}$ in the lower half by an edge. Note that this is the only part that depends on the input a and we can represent all values of the $[k_n^2 - 1]$ bits of a by the k_n^2 possible edges between the k_n nodes $\{l_\nu : \nu \in [k_n - 1]\}$ and the k_n nodes $\{l_\nu : \nu \in \{k_n, \dots, 2k_n - 1\}\}$. We call the resulting graph $G_a = (V_a, E_a)$ (see formal definition below) and define G_b in a similar way depending on b . However, in G_b we also add $n - 8k_n + 2$ fill up nodes a_i and edges (a_i, r_0) to G_b . This will ensure that the final graph $m((G_a, C_{k_n^2}), (G_b, C_{k_n^2}))$ has exactly n nodes. More formally we have:

$$\begin{aligned}
V_a &:= L \cup L' \cup \{c_L\} & V_b &:= R \cup R' \cup \{c_R\} \cup \{a_i : i \in [n - 8k_n + 1]\} \\
E_a &:= \bigcup_{\nu=0}^{2k_n-1} \{(l_\nu, l'_\nu), (l_\nu, c_L)\} & E_b &:= \bigcup_{\nu=0}^{2k_n-1} \{(r_\nu, r'_\nu), (r_\nu, c_R)\} \\
&\cup \{(l_\nu, l_\mu) : \nu, \mu \in [k_n - 1]\} & &\cup \{(a_i, r_0) : i \in [n - 8k_n + 1]\} \\
&\cup \{(l_\nu, l_\mu) : \nu, \mu \in \{k_n, \dots, 2k_n - 1\}\} & &\cup \{(l_\nu, l_\mu) : \nu, \mu \in [k_n - 1]\} \\
&\cup \{(l_{i \bmod k_n}, l_{k_n + \lfloor \frac{i}{k_n} \rfloor}) : & &\cup \{(l_\nu, l_\mu) : \nu, \mu \in \{k_n, \dots, 2k_n - 1\}\} \\
&\quad i \in [k_n^2 - 1], a(i) = 0\} & &\cup \{(l_{i \bmod k_n}, l_{k_n + \lfloor \frac{i}{k_n} \rfloor}) : i \in [k_n^2 - 1], b(i) = 0\}
\end{aligned}$$

Finally we define the cut-set $C_{k_n^2} := \{(l_\nu, r_\nu) : \nu \in [2k_n - 1]\}$ to consist of the edges connecting each l_ν to the corresponding r_ν . Thus \mathcal{R} will be a $(2k_n - 1)$ -reduction. Observe that $(G_a, C_{k_n^2})$ can be computed from a without knowing a and $(G_b, C_{k_n^2})$ can be computed without knowing a , thus this can be used to be the reduction \mathcal{R} with the desired properties.

Now we set $G_{a,b} := m((G_a, C_{k_n^2}), (G_b, C_{k_n^2}))$ (see definition 4.6).

Lemma 5.2. *The graph $G_{a,b}$ is a n -node graph with diameter at most 5.*

Proof. The graph $G_{a,b}$ contains the $8k_n$ nodes in L, L', R, R' . Furthermore it contains $\{c_L, c_R\}$ and $n - (8k_n + 2)$ fill up nodes. Thus in total there are n nodes in the graph. Observe that due to the choice of k_n the number $n - (8k_n + 2)$ of fill up nodes is always non-negative: $n - (8k_n + 2) \geq 10k_n - (8k_n + 2) \geq 0$ where the first inequality follows from the choice of k_n and the second inequality is a result of the fact that $n \geq 10$ implies $k_n \geq 1$.

We prove that the diameter is at most 5 by showing that for any nodes u and v in $G_{a,b}$ the distance $d(u, v)$ is at most 5. To do this we distinguishing three cases:

1. u and v are both in G_a : Observe that every node in G_a is connected to c_L via at most 2 hops. This implies that the distance between any two nodes u and v in G_a is $d(u, v) \leq d(u, c_L) + d(c_L, v) \leq 4$.
2. u and v are both in G_b : This case is completely analog to the previous.
3. u is in G_a and v is in G_b : From u it is at most one hop to some node $l_\nu \in L$ and from v it is at most one hop to some node $l_\mu \in R$. Then there is the following u - v -path of length 5: $(u, l_\nu, c_L, l_\mu, r_\mu, v)$. Thus we conclude that $d(u, v) \leq 5$.

□

As mentioned in the high-level description we will now relate the problem of deciding whether a and b are disjoint to the problem of computing the diameter of a graph. We now extend the analysis of the diameter of $G_{a,b}$.

Theorem 5.3. *The diameter of $G_{a,b}$ is 4 if the sets a and b are disjoint, else it is 5.*

Proof. If a and b are not disjoint, then there exists an i such that $a(i) = b(i) = 1$. Let $\nu := i \bmod k_n$ and $\mu = k_n + \lfloor \frac{i}{k_n} \rfloor$. We show that the two nodes l'_ν and r'_μ have distance at least 5 because any l'_ν - r'_μ -path must contain the neighbors l_ν and r_μ of l'_ν and r'_μ . Furthermore the path must contain an edge from the cut-set $C_{k_n^2}$ since these are the only edges that connect G_a to G_b . Thus there are already three edges in

the path. To obtain a path of length 4 we could only add one more edge from either G_a or G_b . When looking at the construction, the only two paths of length 4 that we could hope for are $(l'_\nu, l_\nu, l_\mu, r_\mu, r'_\mu)$ and $(l'_\nu, l_\nu, r_\nu, r_\mu, r'_\mu)$. However, due to $a(i) = b(i) = 1$ and the choice of ν and μ the construction of $G_{a,b}$ does neither include the edge (l_ν, l_μ) nor the edge (r_ν, r_μ) . Thus none of these paths exists and we conclude that $d(l'_\nu, r'_\mu) > 4$. Now theorem 6.2 implies that $d(l'_\nu, r'_\mu) = 5$ if a and b are not disjoint.

Conversely if a and b are disjoint, the diameter of $G_{a,b}$ is at most 4. We prove this by showing that for any nodes u and v in $G_{a,b}$ the distance $d(u, v)$ is at most 4. To do this we distinguish three cases:

1. u and v are both in G_a : same as in proof of theorem 6.2.
2. u and v are both in G_b : same as in proof of theorem 6.2.
3. u is in G_a and v is in G_b : Without loss of generality we can assume $u \in V_a$ and $v \in V_b$. From u it is at most one hop to some node $l_\nu \in L$ and from v it is at most one hop to some node $l_\mu \in R$. Since we assumed that a and b are disjoint there must be at least one of the edges (l_ν, l_μ) or (r_ν, r_μ) . Thus there will be at least one of the paths (l_ν, l_μ, r_μ) or (l_ν, r_ν, r_μ) witnessing that $d(l_\nu, r_\mu) \leq 2$. Thus we conclude that $d(u, v) \leq d(u, l_\nu) + d(l_\nu, r_\mu) + d(r_\mu, v) \leq 4$.

□

Proof. (of theorem 5.1) To solve the $\text{disj}_{k_n^2}$ function using any algorithm for diam_4 we use the reduction from diam'_4 to diam_4 presented in section 4 and the reduction \mathcal{R} from $\text{disj}_{k_n^2}$ to diam'_4 . According to theorem 4.5, we know that

$$\frac{R_\varepsilon^{\text{cc-pub}}(\text{disj}_{k_n^2})}{2 \cdot |C_{k_n^2}| \cdot B} \leq R_\varepsilon^{\text{dc}}(\text{diam}_4)$$

Due to theorem 2.7 we know that $R_\varepsilon^{\text{cc-pub}}(\text{disj}_{k_n^2})$ is at least $\Omega(k_n^2)$. Together with the fact that $|C_{k_n^2}| = 2k_n$ we conclude that $R_\varepsilon^{\text{dc}}(\text{diam}_4) = \Omega(k_n)$. We obtain the stated result since we chose $k_n := \lfloor \frac{n}{10} \rfloor$. □

6 Diameter Approximation Lower Bound

Theorem 6.1. *For any $\delta > 0$, $n \geq 16 \lceil \frac{3}{4\delta} \rceil + 8$ and $B \geq 1$ there exists an ε such that any distributed $(\frac{3}{2} - \delta)$ -approximative ε -error algorithm A that estimates the diameter of a graph requires at least $\Omega\left(\frac{\sqrt{\delta n}}{B}\right)$ time for some n -node graph with diameter at most $14 \lceil \frac{3}{4\delta} \rceil + 4$.*

The high-level idea is to introduce a gap between the diameters. Graphs $G_{a,b}$ constructed from disjoint inputs a and b will have a diameter that is a factor of at least $(\frac{3}{2} - \delta)$ shorter than the diameter of graphs constructed from inputs that were not disjoint.

First we introduce the constant p_s that will later define the length of “short paths” and set it to be $p_s := \lceil \frac{3}{4\delta} \rceil$. During the reduction we will construct graphs $G_{a,b}$ from a and b such that any $(\frac{3}{2} - \delta)$ -approximation algorithm for the diameter will estimate $\text{diam}(G_{a,b})$ to be less than $6p_s$. If a and b are not disjoint the diameter (and thus the estimate) will always be larger than $6p_s$. Since the reduction \mathcal{R} delivers the above promise-problem we can just use the function diam_{6p_s} as the decision-version of $(\frac{3}{2} - \delta)$ -approximating the diameter.

$$\text{diam}_{6p_s}(G) := \begin{cases} 1 & : \text{diam}(G) < 6p_s \\ 0 & : \text{else} \end{cases}$$

In order to prove theorem 6.1 we follow the high-level idea explained in section 4 and derive a function diam'_{6p_s} from diam_{6p_s} as described in section 4. Like in the previous section, to prove lower bounds depending on n , we choose the length k of the input to the base-function g to depend on n and set $k_n := \left\lfloor \sqrt{\frac{n}{16 \lceil \frac{3}{4\delta} \rceil + 8}} \right\rfloor$. This time we consider the $\text{disj}_{k_n^2}$ problem to be the base-function g . Now we need to

define a reduction \mathcal{R} that given inputs a and b to g maps $(Alice, a)$ and (Bob, b) to inputs $(G_a, C_{k_n^2}^a)$ and $(G_b, C_{k_n^2}^b)$ for $diam'_{6p_s}$. During the reduction \mathcal{R} , Alice defines L and L' , Bob defines R and R' to be the following sets of nodes (as displayed in Figure 2) We define L, L', R and R' as in the previous section.

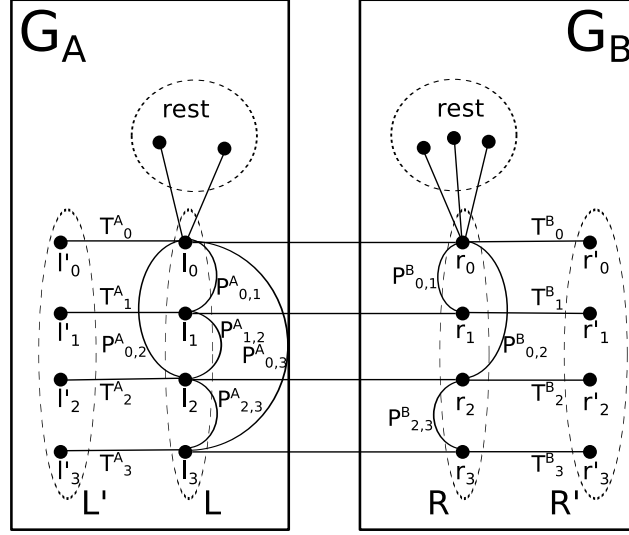


Figure 3: Graph used to calculate $disj_{k_n^2}$ when given a diameter estimator-algorithm. This graph is for the example $k_n = 2, a = (0, 0, 0, 1), b = (0, 1, 1, 1)$. Path $P_{0,2}^A$ represents $a(1) = 0$, path $P_{0,3}^A$ represents $a(2) = 0$, path $P_{1,2}^A$ represents $a(3) = 0$, path $P_{0,2}^B$ represents $b(1) = 0$. Since the sets are not disjoint the diameter is $4p_s$.

Given inputs $a \in \{0, 1\}^{k_n^2}$ and $b \in \{0, 1\}^{k_n^2}$ Alice will construct G_a and Bob G_b . For each $\nu \in [2k - 1]$ Alice adds short paths T_ν^a of length p_s connecting nodes l_ν to l'_ν as depicted in Figure 4. Furthermore for each $(\nu, \mu) \in [2k_n - 1]^2$ Alice adds a long path $P_{\nu,\mu}^a$ of length $p_l := 2p_s$ connecting l_ν to l_μ iff there is no $i \in [k_n^2 - 1]$ with $\nu = i \bmod k_n$ and $\mu = k_n + \lfloor \frac{i}{k_n} \rfloor$ such that $a(i) = 1$. The graph G_b is constructed by Bob in the same way using paths T_ν^b and P_i^b that are added depending on b .

Now we set the cut-set $C_{k_n^2}^a$ that will connect G_a with G_b to be $C_{k_n^2}^a := \cup_{i \in [2k-1]} \{(l_i, r_i)\}$. Thus \mathcal{R} will be a $(2k_n - 1)$ -reduction. We could define $G_{a,b} := m((G_a, C_{k_n^2}^a), (G_b, C_{k_n^2}^b))$ (see definition 4.6), but unfortunately the graph would not necessarily have n nodes yet as each of V_a and V_b is smaller than $n/2$. E.g. for V_a we know:

$$\begin{aligned}
|V_a| &= |L| + |L'| + \#\text{nodes in short paths} + \#\text{nodes in long paths} \\
&= 4k_n + k_n^2 \cdot (p_s - 1) + \sum_{i \in [k_n^2 - 1]: a(i)=1} (p_l - 1) \\
&\leq 4k_n + k_n^2 \cdot (p_l - 1) + k_n^2 \cdot p_s \\
&= k_n^2 \cdot (3p_s + 4) \\
&= \left\lfloor \sqrt{\frac{n}{16 \lceil \frac{3}{4\delta} \rceil + 8}} \right\rfloor^2 \cdot \left(3 \left\lceil \frac{3}{4\delta} \right\rceil + 1 \right) + 4 \\
&\leq n/2
\end{aligned}$$

and can show $|V_b| \leq n/2$ in a similar way. However, we want our lower bound to be valid for all graph-sizes n' and thus need to fill up the graph with nodes until there are n nodes in total. Therefore we add as many nodes $\{f_\nu^a\}$ to G_a such that $|V_a| = n/2$ and as many nodes $\{f_\nu^b\}$ to G_b such that $|V_b| = n/2$.

Finally we can define $G_{a,b} := m((G_a, C_{k_n^2}^a), (G_b, C_{k_n^2}^b))$ using definition 4.6 of m .

Lemma 6.2. *If $a = b = 1^{\lfloor k_n^2 \rfloor}$ the graph is disconnected. Otherwise the graph $G_{a,b}$ over n nodes has diameter at most $16 \lceil p_s \rceil + 4$. Thus the lower bound is valid even for networks of small diameter.*

Proof. In case $a = b = 1^{\lfloor k_n^2 \rfloor}$ there will be no connections between the nodes in the upper part, these are the nodes in $\{l_i : i \in [k-1]\} \cup \{l'_i : i \in [k-1]\} \cup \{r_i : i \in [k-1]\} \cup \{r'_i : i \in [k-1]\} \cup \{a_\nu : \nu \in [n - 8k_n + 1]\}$, and the nodes in the lower part (these are all nodes not listed above). This is due to the construction which will not add any long path with end-points in both sets. However, these long paths are the only way to connect them.

Now we can assume that it is not the case that $a = b = 1^{\lfloor k_n^2 \rfloor}$. Note that the diameter is at most two times the distance from l_0 to the node with largest distance to l_0 . Let u be any node in $G_{a,b}$ we show that $d(u, l_0) \leq 8p_s + 2$ by analyzing two cases:

1. u is in the upper part of $G_{a,b}$ (when looking at figure 6: From u it is at most p_s hops to some node $l \in L$ (or $r \in R$). Due to the construction there will always be a long path connecting l to l_0 (or r to r_0 and then an edge connecting r_0 to l_0). Thus $d(u, l_0) \leq p_s + p_l + 1 \leq 3p_s + 1$.
2. u is in the lower part of $G_{a,b}$ (when looking at figure 6: Since we assume that it is not the case that $a = b = 1^{\lfloor k_n^2 \rfloor}$ there must be some node v in the lower part that is connected to some node w in the upper part by a long path. Now with the same argument as in the first case $d(u, v) \leq p_s + p_l + 1$ and $d(w, l_0) \leq p_s + p_l + 1$. Thus in total we have that

$$d(u, l_0) \leq d(u, v) + d(v, w) + d(w, l_0) \leq 3p_s + 1 + p_l + 3p_s + 1 \leq 8p_s + 2$$

Hence the diameter is at most $16 \lceil p_s \rceil + 4$. □

Lemma 6.3. *The sets a and b are disjoint, if and only if the estimated diameter d' of $G_{a,b}$ is less than $6 \cdot p_s$.*

Proof. If a and b are not disjoint there exists an i such that $a(i) = b(i) = 1$. We will show that the two nodes $l'_{i \bmod k_n}$ and $l'_{k_n + \lfloor \frac{i}{k_n} \rfloor}$ have distance greater than $6 \cdot p_s$. First observe that any path that connects them includes nodes $l_{i \bmod k_n}$ and $l_{k_n + \lfloor \frac{i}{k_n} \rfloor}$. Thus $d(l'_{i \bmod k_n}, l'_{k_n + \lfloor \frac{i}{k_n} \rfloor}) = d(l_{i \bmod k_n}, l_{k_n + \lfloor \frac{i}{k_n} \rfloor}) + 2p_s$. Now the nodes l_j and $r_{j'}$ must have a distance greater than $2p_l$. This is since $a(i) = b(i) = 1$ implies that there is neither a direct path between l_j to $l_{j'}$ nor from r_j to $r_{j'}$ and one need to make a detour visiting another node $v \in L \cup R$. Due to the construction of $G_{a,b}$ this will take at least two long paths. Thus $d(l_{i \bmod k_n}, l_{k_n + \lfloor \frac{i}{k_n} \rfloor}) \geq 2p_l + 2p_s = 6p_s$. Which implies that the diameter d is at least $6 \cdot p_s$. Therefor the estimate d' to the diameter produced by any approximation algorithm (in the sense we defined it) will larger or equal to $6 \cdot p_s$ as well.

Conversely assume that a and b are disjoint and take nodes u and v that define the diameter, that is u and v are chosen such that $d(u, v)$ is maximal with respect to $G_{a,b}$. There is a node $u' \in L \cup R$ such that $d(u, u') \leq p_s$. Similarly there is a node $v' \in L \cup R$ such that $d(v, v') \leq p_s$. Since a and b are disjoint there is always a connection of length $2 + p_l$ between any two nodes in $u', v' \in L \cup R$. Thus $d = d(u, v) \leq 2p_s + 2 + p_l = 4p_s + 2$. Thus we obtain for the estimate d' of any $\frac{3}{2} - \delta$ -approximation algorithm that

$$d' \leq \left(\frac{3}{2} - \delta \right) d = \left(\frac{3}{2} - \delta \right) (4 \cdot p_s + 2) \leq \left(\frac{3}{2} - \frac{3}{4p_s} \right) (4 \cdot p_s + 2),$$

where we use the definition of p_s that depends on δ in the last inequality. This in turn is smaller than $6 \cdot p_s - \frac{3}{2p_s} < 6 \cdot p_s$ and proves the statement. □

Proof. (of theorem 6.1). To solve the $\text{disj}_{k_n^2}$ problem using any $(3/2 - \delta)$ -approximation-algorithm for diam we use the reduction \mathcal{R} from $\text{disj}_{k_n^2}$ to diam'_{6p_s} and observed that \mathcal{R} delivered a promise-problem such that there is a reduction from diam'_{6p_s} to diam via diam_{6p_s} . However, we need to assume that either

a or b contains at least one 0, else $G_{a,b}$ is disconnected to apply lemma 6.2. Fortunately we can ignore this case since Alice and Bob can easily determine using 2 bits of communication if this is the case before simulating A . This will not affect our asymptotic lower bounds. According to theorem 4.5, we know that

$$\frac{R_\varepsilon^{cc-pub}(\text{disj}_{k_n^2})}{2 \cdot |C_{k_n^2}| \cdot B} \leq R_\varepsilon^{dc}(\text{diam}_{4p_s})$$

Due to theorem 2.7 we know that $R_\varepsilon^{cc-pub}(\text{disj}_{k_n^2})$ is at least $\Omega(k_n^2)$. Together with the fact that $|C_{k_n^2}| = 2k_n$ we conclude that for all inputs to g of size k_n we obtain $R_\varepsilon^{dc}(\text{diam}_{4p_s}) = \Omega(k_n)$. We obtain the result since we chose $k_n := \left\lfloor \sqrt{\frac{n}{16 \lceil \frac{3}{4\delta} \rceil + 8}} \right\rfloor$. \square

7 Girth Approximation Lower Bound

Theorem 7.1. *For any $\delta > 0$, $n \geq 16 \lceil \frac{2}{\delta} \rceil + 4$ and $B \geq 1$ there exists an ε such that any distributed $(2 - \delta)$ -approximative ε -error algorithm A that estimates the girth of a graph requires at least $\Omega\left(\frac{\sqrt{\delta n}}{B}\right)$ time for some n -node graph with diameter at most $(8 \lceil \frac{2}{\delta} \rceil + 2)$*

Remark 7.2. *The diameter in the theorem above does not depend on n . Furthermore this theorem can be extended to hold for any larger diameter. However we are interested in small diameters, since then communication distances are short.*

The high-level idea is to introduce a gap between the girths. Graphs $G_{a,b}$ constructed from disjoint inputs a and b will have a girth that is a factor shorter than the diameter of graphs constructed from inputs that were not disjoint.

First we define the constants $p_s := \lceil \frac{2}{\delta} \rceil$ and $p_l := 4 \cdot p_s$ indicating the length of short/long paths that we will use during the reduction \mathcal{R} . During the reduction we will construct graphs $G_{a,b}$ from a and b such that any $(2 - \delta)$ -approximation algorithm for the girth will estimate $\text{girth}(G_{a,b})$ to be less than p_l . If a and b are not disjoint the girth (and thus the estimate) will always be larger than p_l . Since the reduction \mathcal{R} delivers the above promise-problem we can just use the function diam_{p_l} as the decision-version of $(2 - \delta)$ -approximating the diameter.

$$\text{diam}_{p_l}(G) := \begin{cases} 1 & : \text{diam}(G) < p_l p_s \\ 0 & : \text{else} \end{cases}$$

In order to prove theorem 7.1 we follow the high-level idea explained in section 4 and derive a function girth' from girth as described in section 4. To prove lower bounds depending on n , we choose the length k of the input to the base-function g to depend on n and set $k_n := \left\lfloor \sqrt{\frac{n}{16 \lceil \frac{2}{\delta} \rceil + 4}} \right\rfloor$. As base-function g we consider the $\text{disj}_{k_n^2}$ problem. Now we need to define a reduction \mathcal{R} that given inputs a and b to g maps (Alice, a) and (Bob, b) to inputs $(G_a, C_{k_n^2})$ and $(G_b, C_{k_n^2})$ for girth' .

During the reduction \mathcal{R} , Alice defines L and L' , Bob defines R and R' as in the previous section. Given input $a \in \{0, 1\}^{k_n^2}$ and Alice will construct G_a and given input $b \in \{0, 1\}^{k_n^2}$ Bob will construct G_b .

For each $(\nu, \mu) \in [2k_n - 1]^2$ Alice adds a short path $P_{\nu, \mu}^a$ of length p_s connecting l_ν to l_μ iff there is no $i \in [k_n^2 - 1]$ with $\nu = i \bmod k_n$ and $\mu = k_n + \lfloor \frac{i}{k_n} \rfloor$ such that $a(i) = 1$. Furthermore Alice adds paths $P_{i \bmod k_n, k_n + \lfloor \frac{i}{k_n} \rfloor}^a$ of length p_l iff for the corresponding i the input is $a(i) = 1$. An example for this is displayed in figure 4. In a similar way Bob adds paths $P_{\nu, \mu}^b$ to G_b depending b . Thus \mathcal{R} will be a $(2k_n - 1)$ -reduction.

Now we set the cut-set $C_{k_n^2}$ that will connect G_a with G_b to be $C_{k_n^2} := \cup_{i \in [2k_n - 1]} \{(l_i, r_i)\}$. Now we could define $G_{a,b} := m((G_a, C_{k_n^2}), (G_b, C_{k_n^2}))$ (see definition 4.6), but observe that the number of nodes

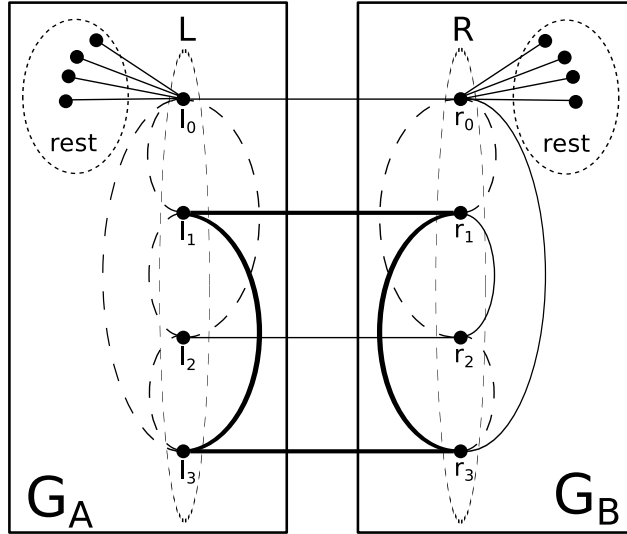


Figure 4: Graph used to calculate $\text{disj}_{k_n^2}$ when given a girth estimator-algorithm. This graph is for the example $k_n = 2, a = (0, 0, 0, 1), b = (0, 1, 1, 1)$. The paths marked as dashed lines are long and the paths that are drawn out are short. To be more detailed, the long $l_0 - l_2$ -path represents $a(1) = 0$, the long $l_0 - l_3$ -path represents $a(2) = 0$, the long $l_1 - l_2$ -path represents $a(3) = 0$ and the short $l_1 - l_3$ -path represents $a(4) = 1$. The long $r_0 - r_2$ -path represents $b(1) = 0$, the short $r_0 - r_3$ -path represents $b(2) = 1$, the short $r_1 - r_2$ -path represents $b(3) = 1$, the short $r_1 - r_3$ -path represents $b(4) = 1$. Since the sets are not disjoint there is a small cycle which is drawn in thick lines.

that we added so far to V_a and V_b is not necessarily n yet:

$$\begin{aligned}
|V_A| &= |L| + \#\text{nodes in long and short paths} \\
&\leq 2k_n + (2k_n)^2 \cdot (p_l - 1) \\
&\leq (2k_n)^2 \cdot p_l \\
&= \left\lceil \sqrt{\frac{n}{16 \lceil \frac{2}{\delta} \rceil + 4}} \right\rceil^2 \cdot 4 \left\lceil \frac{2}{\delta} \right\rceil \\
&\leq n/2
\end{aligned}$$

One can show $|V_b| \leq n/2$ in a similar way. However, we want our lower bound to be valid for all graph-sizes n and thus need to fill up the graph with nodes until it is large enough. Therefore we add as many nodes $\{f_\nu^a\}$ to G_a such that $|V_a| = n/2$ and as many nodes $\{f_\nu^b\}$ to G_b such that $|V_b| = n/2$.

Finally we can define $G_{a,b} := m((G_a, C_{k_n^2}), (G_b, C_{k_n^2}))$ (see definition 4.6).

Lemma 7.3. *Any graph $G_{a,b}$ as constructed above has diameter at most $8 \lceil \frac{2}{\delta} \rceil + 2$. Thus the lower bound is valid even for networks of small diameter.*

Proof. Let u and v be any nodes in $V_{a,b}$. From u and v respectively to the nearest nodes $u', v' \in L \cup R$ it is at most $\frac{p_l+1}{2}$ hops. Since nodes u' and v' are in $L \cup R$ they can reach each other by at most traversing one one path $p_{\nu,\mu}$ (with appropriate ν, μ) of length p_l and one edge that connects L and R . Therefore $G_{a,b}$ has diameter at most $2 \cdot p_l + 2 = 8 \lceil \frac{2}{\delta} \rceil + 2$. \square

Lemma 7.4. *The sets a and b are disjoint, if and only if the estimated girth g' of $G_{a,b}$ is at least p_l .*

Proof. First we prove that if a and b are not disjoint, there will be a cycle of length $2 \cdot p_s + 2$ causing that any estimate g' will be less than p_l . Later we show that if a and b are disjoint, there will be no cycle shorter than p_l .

If a and b are not disjoint there exists an i such that $a(i) = b(i) = 1$. The cycle

$$\left(P_{k_n + \lfloor \frac{i}{k_n} \rfloor, i \bmod k_n}^a, (r_i \bmod k_n, l_i \bmod k_n), P_{i \bmod k_n, k_n + \lfloor \frac{i}{k_n} \rfloor}^b, (l_{k_n + \lfloor \frac{i}{k_n} \rfloor}, r_{k_n + \lfloor \frac{i}{k_n} \rfloor}) \right)$$

consisting of two paths and two edges has size $(2 \cdot p_s + 2)$ due to the definition of the paths. Since we know for the estimate g' that $g' \leq (2 - \delta)g$ where g is the actual girth of graph $G_{a,b}$. From this we conclude that $g' \leq (2 - \delta)g \leq \left(2 - \frac{2}{p_s}\right)g$ by using the definition of δ . Inserting the value of g we obtained above, we get $g' \leq \left(2 - \frac{2}{p_s}\right)(2 \cdot p_s + 2) \leq 4 \cdot p_s - 4 \cdot \frac{1}{p_s} < p_l$

Conversely if a and b are disjoint, there will be no cycle shorter than p_l . We distinguish four cases, the two most important ones are displayed in the figures 5 and 6.

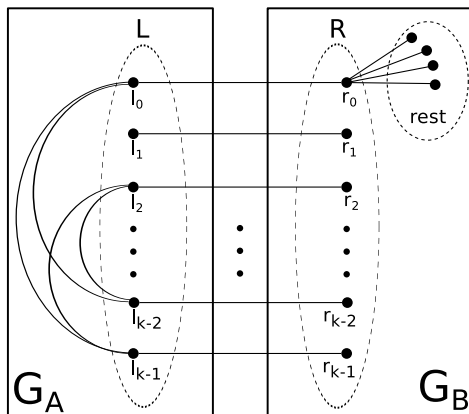


Figure 5: Case 2 of lemma 7.4

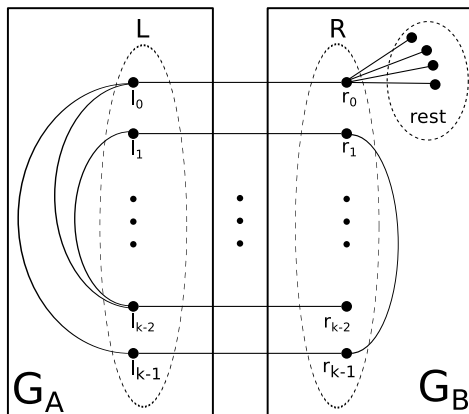


Figure 6: Case 4 of lemma 7.4

1. In case that the smallest cycle contains a long path: This means that the real girth g is at least p_l and thus the estimator g' is also at least p_l . In all following cases we assume that the smallest cycle contain no long path.
2. In case that the the smallest cycle contains no path of length p_l but is completely contained in G_a : Since there can be no long path in the cycle and since short paths go only from the upper to the lower half (by construction), the cycle must contain at least 4 short paths. Hence the girth g and also the estimator g' of $G_{a,b}$ is at least $4 \cdot p_s = p_l$.

3. In case that the smallest cycle contains no path of length p_l but is completely contained in G_b : This case is completely analog to the previous.
4. In case that the smallest cycle contains no path of length p_l but nodes from both G_a and G_b : The cycle must contain 2 edges of type $(r_\nu, l_\nu), (r_\mu, l_\mu)$. If both the ν and μ are smaller than k_n or both are at least k_n , this means that neither the left nor the right nodes can be connected by one short path. Therefore the cycle contains at least 4 short paths. Conversely if ν smaller than k_n and μ at least k_n or the other way around. It is not possible that those edges are directly connected by short paths on both sides, because a and b are disjoint. The connection-path on the left / right side which is not connected by one short path contains at least 3 short paths. Hence the girth and estimator of $G_{a,b}$ is at least $4 \cdot p_s = p_l$.

□

Proof. (of theorem 7.1). To solve the $\text{disj}_{k_n^2}$ problem using any $(2 - \delta)$ -approximation-algorithm for *girth* we use the reduction \mathcal{R} from $\text{disj}_{k_n^2}$ to girth'_{p_l} and observed that \mathcal{R} delivered a promise-problem such that there is a reduction from diam'_{p_l} to diam_{p_l} via diam_{p_l} . However, we need to assume that either a or b contains at least one 0, else $G_{a,b}$ is disconnected to apply lemma 7.4. Fortunately we can ignore this case since Alice and Bob can easily determine using 2 bits of communication if this is the case before simulating A . This will not affect our asymptotic lower bounds. According to theorem 4.5, we know that

$$\frac{R_\varepsilon^{\text{cc-pub}}(\text{disj}_{k_n^2})}{2 \cdot |C_{k_n^2}| \cdot B} \leq R_\varepsilon^{\text{dc}}(\text{girth}_{p_l})$$

Due to theorem 2.7 we know that $R_\varepsilon^{\text{cc-pub}}(\text{disj}_{k_n^2})$ is at least $\Omega(k_n^2)$. Together with the fact that $|C_{k_n^2}| = 2k_n$ we conclude that for all inputs to g of size k_n we obtain $R_\varepsilon^{\text{dc}}(\text{girth}_{p_l}) = \Omega(k_n)$. We obtain the result since we chose $k_n := \left\lfloor \sqrt{\frac{n}{16 \lceil \frac{n}{8} \rceil + 4}} \right\rfloor$. □

References

- [1] D. Aingworth, C. Chekuri, P. Indyk, and R. Motwani. Fast estimation of diameter and shortest paths (without matrix multiplication). *SIAM Journal on Computing*, 28(4):1167–1181, 1999.
- [2] N. Alon, Z. Galil, and O. Margalit. On the exponent of the all pairs shortest path problem. In *Proceedings of the 32nd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 569–575, 1991.
- [3] N. Alon, O. Margalit, Z. Galil, and M. Naor. Witnesses for boolean matrix multiplication and for shortest paths. In *33rd Annual Symposium on Foundations of Computer Science (FOCS) : October 24-27, 1992, Pittsburg, Pennsylvania: proceedings [papers]*, page 417. IEEE Computer Society, 1992.
- [4] G.E. Blelloch, V. Vassilevska, and R. Williams. A new combinatorial approach for sparse graph problems. In *Proceedings of the 35th international colloquium on Automata, Languages and Programming, Part I (ICALP)*, pages 108–120. Springer-Verlag, 2008.
- [5] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *Journal of symbolic computation*, 9(3):251–280, 1990.
- [6] M. Elkin. Unconditional lower bounds on the time-approximation tradeoffs for the distributed minimum spanning tree problem. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing (STOC)*, pages 331–340. ACM, 2004.

- [7] F. Fich and E. Ruppert. Hundreds of impossibility results for distributed computing. *Distributed computing*, 16(2):121–163, 2003.
- [8] J.A. Garay, S. Kutten, and D. Peleg. A sublinear time distributed algorithm for minimum-weight spanning trees. *SIAM Journal on Computing*, 27(1):302–316, 1998.
- [9] R.L. Graham, A.C. Yao, and F.F. Yao. Information bounds are weak in the shortest distance problem. *Journal of the ACM (JACM)*, 27(3):428–444, 1980.
- [10] T. Hagerup. Improved shortest paths on the word ram. In *Proceedings of the 27th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 61–72. Springer-Verlag, 2000.
- [11] D.R. Karger, D. Koller, and S.J. Phillips. Finding the hidden path: Time bounds for all-pairs shortest paths. *SIAM Journal on Computing*, 22:1199, 1993.
- [12] L.R. Kerr. The effect of algebraic structure on the computational complexity of matrix multiplication, cornell university, phd thesis. 1970.
- [13] L. Kor, A. Korman, and D. Peleg. Tight Bounds For Distributed MST Verification. In Thomas Schwentick and Christoph Dürr, editors, *28th International Symposium on Theoretical Aspects of Computer Science (STACS) 2011*, volume 9 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 69–80, Dagstuhl, Germany, 2011. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.
- [14] F. Kuhn, T. Moscibroda, and R. Wattenhofer. What cannot be computed locally! In *Proceedings of the twenty-third annual ACM symposium on Principles of distributed computing (PODC)*, pages 300–309. ACM, 2004.
- [15] F. Kuhn, T. Moscibroda, and R. Wattenhofer. The price of being near-sighted. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm (SODA)*, SODA '06, pages 980–989, New York, NY, USA, 2006. ACM.
- [16] Eyal Kushilevitz and Noam Nisan. *Communication complexity*. Cambridge University Press, 1997.
- [17] N. Linial. Locality in distributed graph algorithms. *SIAM Journal on Computing*, 21:193, 1992.
- [18] Z. Lotker, B. Patt-Shamir, and D. Peleg. Distributed mst for constant diameter graphs. In *Proceedings of the twentieth annual ACM symposium on Principles of distributed computing (PODC)*, pages 63–71. ACM, 2001.
- [19] D. Nanongkai, A. Das Sarma, and G. Pandurangan. A tight unconditional lower bound on distributed random walk computation. In *Proceedings of the 2011 ACM Symposium Principles of Distributed Computing (PODC)*, pages 257–266, 2011.
- [20] D. Peleg. *Distributed computing: a locality-sensitive approach*. Society for Industrial Mathematics, 2000.
- [21] D. Peleg and V. Rubinovich. A near-tight lower bound on the time complexity of distributed minimum-weight spanning tree construction. *SIAM Journal on Computing*, 30(5):1427–1442, 2001.
- [22] S. Pettie. A new approach to all-pairs shortest paths on real-weighted graphs* 1. *Theoretical Computer Science*, 312(1):47–74, 2004.
- [23] S. Pettie and V. Ramachandran. A shortest path algorithm for real-weighted undirected graphs. *SIAM Journal on Computing*, 34(6):1398–1431, 2005.
- [24] A.D. Sarma, S. Holzer, L. Kor, A. Korman, D. Nanongkai, G. Pandurangan, D. Peleg, and R. Wattenhofer. Distributed verification and hardness of distributed approximation. *43rd ACM Symposium on Theory of Computing (STOC)*, 2011.

- [25] R. Seidel. On the all-pairs-shortest-path problem in unweighted undirected graphs. *Journal of Computer and System Sciences*, 51(3):400–403, 1995.
- [26] M. Thorup. Undirected single-source shortest paths with positive integer weights in linear time. *Journal of the ACM (JACM)*, 46(3):362–394, 1999.
- [27] A.C.C. Yao. Some complexity questions related to distributive computing. In *Proceedings of the eleventh annual ACM symposium on Theory of computing (STOC)*, pages 209–213. ACM, 1979.