Networks Cannot Compute Their Diameter in Sublinear Time

Preliminary version, please check for updates.

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

Abstract

We study the problem of computing the diameter of a network in a distributed way. In the model of distributed computation we consider is node can transmit a different (but short) message to each of its neighbors each synchronous round. 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 $\tilde{\Omega}(\sqrt{n}+D)$ rounds. Furthermore we use our new technique to prove an $\tilde{\Omega}(\sqrt{n}+D)$ lower bound on approximating the girth of a graph by a factor $2-\varepsilon$.

1 Introduction

Without doubt, a fundamental property of a network is its diameter (denote by D). 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 (both require time $\Theta(D)$. 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 $\mathcal{O}(D)$ 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 $\mathcal{O}(D)$ time? In this paper, we answer this question negatively, by presenting a lower bound of $\tilde{\Omega}(n)$, 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 an $\tilde{\Omega}(\sqrt{n}+D)$ distributed complexity.

All our bounds hold in the message passing model with limited bandwidth (also known as CONGEST model, [25]), the standard model of distributed computation. In this model 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 $\tilde{\Omega}(n)$ in a system that is m-parallel. That is

[1]More 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 [19], whereas matching and vertex cover are pseudo-local [18].

a speedup of m is possible due to communicating over m edges of the network at the same time. This indicates that the diameter problem does not allow for a high parallel speedup of sequential algorithms.

Usually the fastest known sequential algorithms (e.g. [2, 3, 33]) use the fast matrix multiplication algorithm by Coppersmith and Winograd [5] and run in time $\mathcal{O}(n^{2.376})$. For sparse graphs well thought specialized algorithms such as [4] are 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 are 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. This model is called CONGEST(B) model [26]. Typically one chooses $B = \mathcal{O}(\log n)$, 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 $B = \mathcal{O}(\log n)$ the model is just called CONGEST. We are interested in the minimal number of communication-rounds that are needed until some problem is solved. Therefore we assume that internal computation is free. To be more formal, we are interested in evaluating a function $h : \mathbb{C}_n \to S$, where $\mathbb{C}_n$ is the set of all graphs over n nodes, and S is e.g. $\{0, 1\}$ or $\mathbb{N}$ and define distributed round complexity as follows:

Definition 2.1. *(distributed round complexity). Let $\mathcal{A}_\varepsilon$ be the set of distributed algorithms that use (public) randomness (indicated by "pub") and evaluate a function h on the underlying graph G over n nodes with an error probability smaller than ε. Denote by $R_\varepsilon^{dc-pub}(A(G))$ the distributed round complexity (indicated by "dc") representing the number of rounds that an algorithm $A \in \mathcal{A}_\varepsilon$ needs in order to compute $h(G)$ on G. We define*

$$R_\varepsilon^{dc-pub}(h) = \min_{A \in \mathcal{A}_\varepsilon} \max_{G \in \mathbb{C}_n} R_\varepsilon^{dc-pub}(A(G))$$

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

Throughout the paper we often state results in a less formal way. Example: when h is *diam* (the function that maps a graph to its diameter) and $R_0^{dc-pub}(diam) = \mathcal{O}(n)$, we often just write "the deterministic round complexity of computing the diameter is $\mathcal{O}(n)$".

The problems we consider are computing the diameter and the girth of a graph as well as approximations to them. A set $\{0, \ldots, k\}$ is denoted by writing $[k]_0$.

Definition 2.2. *(distance, diameter, girth). 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. The diameter $D := \max_{u,v \in V} d(u, v)$ of a graph G is the maximum distance between any two nodes of the graph. Sometimes we write $diam(G)$ instead of D. The girth g of a graph G is the length of the shortest cycle in G. (If G is a forest its girth is infinity.)*

Definition 2.3. *(approximation). Given an optimization problem P over graphs, denote by $OPT_P(G) \in \mathbb{Q}$ 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-approximate 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 the basics of communication complexity that was first introduced by Yao in [35]. 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 $h : \{0,1\}^k \times \{0,1\}^k \to \{0,1\}$ 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 h.

Definition 2.4. *(communication complexity). Let $\mathcal{A}_\varepsilon$ be the set of two-party algorithms that use public randomness (denoted by pub) and when used by Alice and Bob, compute h on any input a to Alice and b to Bob with an error probability smaller than ε. Let $A \in \mathcal{A}_\varepsilon$ be an algorithm that computes h. Denote by $R_\varepsilon^{cc-pub}(A(a,b))$ the communication complexity (denoted by cc) representing the number of 1-bit messages exchanged by Alice and Bob while executing algorithm A on a and b. We define*

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

to be the smallest amount of bits any algorithm would need to send in order to compute h.

A well studied problem in communication complexity is that of set disjointness, where we are given two subsets of $[k-1]_0$ and need to decide whether they are disjoint. Here, the strings a and b indicate membership of elements to each of these sets.

Definition 2.5. *(disjointness problem). The set disjointness function* $\mathrm{disj_k} : \{0,1\}^k \times \{0,1\}^k \to \{0,1\}$ *is defined as follows.*

$$\mathrm{disj_k}(a,b) = \begin{cases} 0 & \textit{if there is an } i \in [k-1]_0 \textit{ such that } a(i)=b(i)=1 \\ 1 & \textit{otherwise} \end{cases}$$

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 use the following basic theorem that was proven in Example 3.22 in [20].

Theorem 2.1. *For any sufficiently small $\varepsilon > 0$ we can bound $R_\varepsilon^{cc-pub}(\mathrm{disj_k})$ by $\Omega(k)$.*

3 Our Contribution and Related Work

In this paper we show that networks require $\Omega(n/B)$ rounds in the message passing model with limited bandwidth to calculate the exact diameter. We obtain the lower bound even for graphs with small diameter. This is particularly interesting, since each two nodes are in close communication range to each other. In Section 6 we also show that an approximation to the diameter that is closer than a factor of $\frac{3}{2}$ to the actual diameter is impossible to obtain by using any algorithm that does not use $\Omega(\sqrt{n}/B + D)$ 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 a $\frac{3}{2}$-approximation in time $\mathcal{O}(m \cdot \sqrt{n \log n} + n^2 \cdot \log n)$ which would imply an $\mathcal{O}(\sqrt{n \log n} + \frac{n^2 \cdot \log n}{m})$-algorithm if it could be fully parallelized in our model. 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 $\mathcal{O}(1)$ 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 show an $\Omega(\sqrt{n}/B + D)$ lower bound for approximations better than 2 of the girth. Opposed to this the diameter can be 2-approximated in time $\mathcal{O}(D)$ by performing a breadth-first search from one node 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 as described in [6]. The first paper that introduced a technique to apply lower bounds for communication complexity in a distributed setting is [28] that proved an $\tilde{\Omega}(\sqrt{n}/B + D)$ lower bound[2] on computing a minimum spanning tree (MST). Later [7] improved their technique by using a modified graph to yield approximation lower bounds for MST. In [6] 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 [7] but a new technique on 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 transfer the communication complexity lower bounds. One main difference to the previous results that provided $\tilde{\Omega}(\sqrt{n}/B + D)$ lower bounds is that our graphs are able to yield $\Omega(n/B + D)$ lower bounds.

Observe that the bounds we derive can not be obtained by just extending the technique of [6] but require a new approach. To explain why this is the case we summarize the key-ideas of [6]. The first step in [6] 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 [6] 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 [6] this is not possible due to almost matching upper bounds. Usually the graphs of [6] are constructed such that any algorithm needs to send (roughly) $\sqrt{n}$ bits either 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 [6] 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 heavily rely on these long paths and the above mentioned connections between them. Exactly this fact is the reason why we cannot just use these graphs to prove lower bounds on computing the diameter: to obtain a meaningful lower bound we would always 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. E.g.: What if we want to give the algorithm a hard time distinguishing whether the diameter is 4 or 5? A similar reasoning can be done for computing the girth of a graph. 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 [6] for the problems studied therein.

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 algorithm to perform bad has a diameter depending on n. E.g. in [6] the diameter is $\Theta(\log n)$ and the nice follow-up [24] 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 is at the cost of getting weaker lower bounds, e.g. [7, 23, 6]. As mentioned in [10] 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 by providing worst case graphs. These graphs have even small diameter (at most five).

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

3.1 Further Related Work Concerning Lower Bounds in Distributed Computing. Although [8] lists hundreds of lower bounds in distributed computing, only few are known in our model. Most of them are already mentioned above and use similar techniques. A further bound that used techniques related to those in [6] is the unconditional $\Omega(\sqrt{lD} + D)$ lower bound on computing random walks of length l in diameter D graphs by [24]. The authors of [24] were able to provide strong lower bounds that even depend on the diameter itself. One of the further rare bounds in our model is presented in [17] and concerns MST verification. They demonstrate examples showing that this needs $\Omega(\sqrt{n} + D)$ time.

3.2 Concerning 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 node in the graph - the depth of the deepest such BFS-tree is the diameter. This takes time $\mathcal{O}(n^2 + n \cdot m)$ in most sequential models of computing. In the distributed model considered in this paper, this approach (if not modified) takes time $\mathcal{O}(n \cdot D)$ since each BFS-search takes $\mathcal{O}(D)$ time.

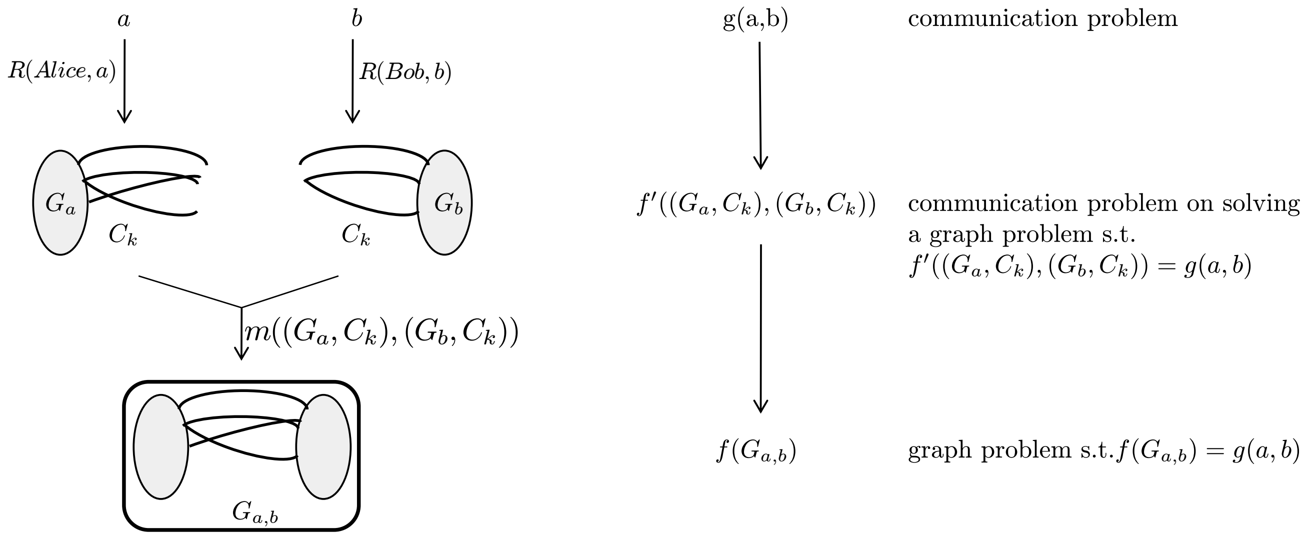
Due to its importance and close connection to the all pairs shortest path problem (APSP), much effort was spent to obtain fast (sequential) algorithms for various versions of computing the diameter and APSP, e.g. in [2, 3, 4, 5, 33]. Although we already mentioned that no lower bounds are known for computing the diameter, at least few are known for APSP. These are of interest as well since all known fast algorithms to compute the exact 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 nontrivial 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. More details on this and lower bounds for distributed computation of APSP can be found in [13].

Regarding known lower bounds for APSP in sequential models, one of the first lower bounds was provided by Kerr in [16] who showed that any oblivious APSP algorithm need to perform $\Omega(n^3)$ comparisons. Later Graham, Yao, and Yao showed in [11] that if one wants to use an information-theoretic argument, it is 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. [15] 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 [29] 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 [12, 30, 34]. 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. Note that all these lower bounds can easily be broken using fast matrix multiplication (at least when m is not too small) demonstrating that these classes of algorithms for which lower bounds are known are rather restrictive. In contrast to this our lower bounds are valid for all algorithms in the model we consider.

3.3 Recent Related Work and Improvements. We summarize progress since the conferece version [9] of this paper appeared. The authors of [14, 27] independently discovered algorithms to solve APSP in time $O(n)$. This implies $O(n)$-algorithms for the diameter that almost match the lower bound provided here. Later, [21] presented a third way to compute APSP in time $O(n)$. Distributed lower bounds and algorithms for various approximations to the diameter and related problems such as center, radius and peripheral vertices were studied in [13, 14, 21, 27]. Among these, most relevant our paper is an asymptotically almost optimal 3/2-approximation to the diameter stated in [13] that is obtained by combining results of [14, 27] with ideas from [31]. The same runtime is achieved in [21] using a different approach. Regarding the girth, a $(2 - 1/g)$-approximation can be computed in time $\mathcal{O}(n^{\frac{2}{3}} + D \cdot \log \frac{D}{g})$ according to [14]. Improved distributed approximations of shortest paths and construction of routing tables were studied in [22, 32]. Our $\tilde{\Omega}(n)$-lower bound for computing the diameter was modified to work for graphs with diameter 2 and 3 in [14].

4 Relating Distributed Round Complexity to Two-Party Communication Complexity

We 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 consider decision-versions of the problems, e.g. "Is the diameter larger than 4?". These versions immediately imply lower bounds on the original problems. To achieve these lower bounds we use reductions that transform two-party communication complexity lower bounds (on the number of bits that need to be exchanged by 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 pick a "base"-function h (e.g. the disjointness function) that can be evaluated by two parties and a communication lower bound for this evaluation (e.g. Theorem 2.1) to derive a lower bound for f' and thus for f. An overview of the whole reduction-procedure can be found in Figure 4.



a
b
$R(Alice, a)$
$R(Bob, b)$
G_a
C_k
C_k
G_b
$m((G_a, C_k), (G_b, C_k))$
$G_{a,b}$
g(a,b)
communication problem
$f'((G_a, C_k), (G_b, C_k))$
communication problem on solving a graph problem s.t. $f'((G_a, C_k), (G_b, C_k)) = g(a, b)$
$f(G_{a,b})$
graph problem s.t. $f(G_{a,b}) = g(a, b)$

Figure 1: Upper part: This part depends on the problem at hand and is described in the sections devoted to the problems we study. In general we reduce the communication problem of computing $h(a, b)$ to 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 a part of it is given to Bob. According to the reduction $\mathcal{R}$**ed**, Alice constructs part G_a (light gray, left side) from a and Bob G_b (light gray, right side) from b. Both adds the same set of edges C_k to G_a and G_b respectively. Lower part: This part is independent of f and shared by all of our reductions and is 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 $G_{a,b}$ 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. To keep things simple we introduce 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 $E = E_a \dot\cup E_b \dot\cup C_k$, where $\dot\cup$ denotes the disjoint union of two sets. The cut-set C_k consists of $c_k := |C_k|$ edges whose endpoints are in different subsets of the node-partition.*

Observe that now we can define a two-party communication problem f' according to the graph-problem f in a canonical way. That is we define

$$f'((G_a, C_k), (G_b, C_k)) := f(G)$$

for any graph G and cut (G_a, G_b, C_k) of G. When computing $f'((G_a, C_k), (G_b, C_k))$ Alice gets input (G_a, C_k) and Bob gets input (G_b, C_k). Now we formally show how f' can be reduced to f and analyze the complexity-relation in Theorem 4.1.

Lemma 4.1. *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 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. In 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 during round r of the execution of algorithm 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.

At the same time 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.

In the following sections we want to further reduce a "base-"function (such as $\mathrm{disj_k}$) of type $h : \{0,1\}^k \times \{0,1\}^k \to \{0,1\}$ to f' and define:

Definition 4.2. *A c_k-reduction*

$$\mathcal{R}\mathbf{ed} : \{Alice, Bob\} \times \{0,1\}^k \to \{(H, C_k) : G \text{ is any graph and } H \text{ is any subgraph of } G \text{ such that } (H, G \setminus H, C_k) \text{ is a cut of } G \text{ with } |C_k| = c_k\}$$

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

Observe that the size of C_k does not depend on a nor b.

Remark 4.1. *Using a reduction as above we obtain $R_\varepsilon^{cc-pub}(g) = R_\varepsilon^{cc-pub}(f')$ since the reduction requires $h(a, b) = f'(\mathcal{R}\mathbf{ed}(Alice, a), \mathcal{R}\mathbf{ed}(Bob, b))$ which in turn implies that the same number of bits need to be exchanged when computing $h(a, b)$ and $f'(\mathcal{R}\mathbf{ed}(Alice, a), \mathcal{R}\mathbf{ed}(Bob, b))$.*

We state the relation of the complexities of f' and f depending on c_k since it turns out that exactly these c_k edges can be used to simulate the single communication channel of Alice and Bob.

Theorem 4.1. *Let $B \geq 1$ and f be any function on graphs and f' the function derived from f as described above. Let h be a base-function that can be reduced to f' using a c_k-reduction. We can bound*

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

Proof. Given a graph G and cut (G_a, G_b, C_k), where C_k is of size c_k. We know that $R_\varepsilon^{cc-pub}(f')$ is a lower bound on the number of bits that any distributed algorithm A must send over edges in C_k. This is because the information a and b are stored in G_a and G_b and these parts of the graph can use exactly the edges in C_k to communicate with each other. Now these bits can not be encoded in less than $\frac{R_\varepsilon^{cc-pub}(f')}{B}$ messages in our distributed model and we conclude that $\frac{R_\varepsilon^{cc-pub}(f')}{B}$ 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 $\frac{R_\varepsilon^{cc-pub}(f')}{2|C_k|B}$ is always a lower bound for $R_\varepsilon^{dc-pub}(f)$. 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 4.1 the statement follows.

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

Definition 4.3. *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.*

Remark 5.1. *The diameter of the graph used to prove 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.*

Deciding whether a graph G has diameter less than 5 or not is the decision-version $diam_5$ of the function $diam$, that is

$$diam_5(G) := \begin{cases} 1 & : diam(G) < 5 \\ 0 & : \text{else} \end{cases}$$

In order to prove Theorem 5.1 we follow the high-level idea explained in Section 4, where f is $diam_5$, and derive a function $diam_5'$ from $diam_5$ as described in Section 4. To prove bounds depending on n, we choose the length k of the input to the base-function h to be a function $k(n)^2$ depending on n and set $k(n) := \left\lfloor \frac{n}{10} \right\rfloor$. As base-function h we consider the $disj_{k(n)^2}$ problem. Now we need to define a reduction $\mathcal{R}\mathbf{ed}$ that given inputs a and b to h maps $(Alice, a)$ and (Bob, b) to inputs $(G_a, C_{k(n)^2})$ and $(G_b, C_{k(n)^2})$ for $diam_5'$. During the reduction $\mathcal{R}\mathbf{ed}$, Alice defines the following sets of nodes L and L', Bob defines R and R' (as displayed in Figure 2):

$$L = \{l_\nu | \nu \in [2k(n)-1]_0\} \quad R = \{r_\nu | \nu \in [2k(n)-1]_0\}$$
$$L' = \{l'_\nu | \nu \in [2k(n)-1]_0\} \quad R' = \{r'_\nu | \nu \in [2k(n)-1]_0\}$$

As a first step (that is independent of the input a) in constructing G_a, Alice connects nodes l_ν with nodes l'_ν for all $\nu \in [2k(n)-1]_0$. In the same way Bob connects r_ν with r'_ν to construct G_b. Furthermore

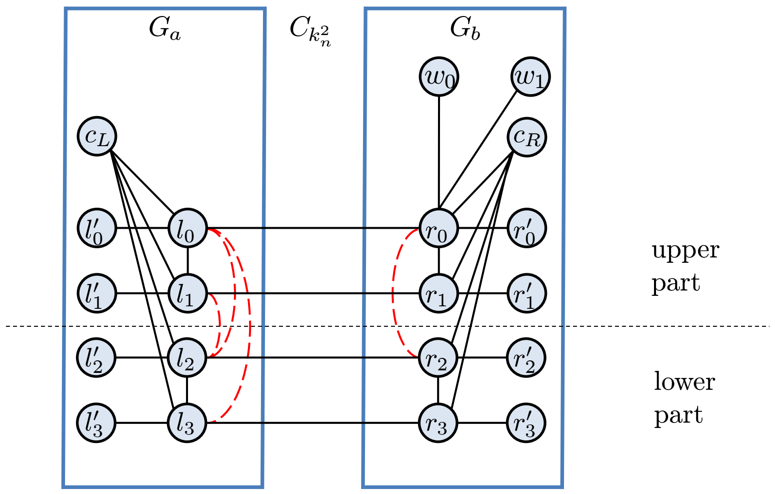


Figure 2: The above graph $G_{a,b}$ is for $n = 20$. Therefore we set $k(n) = 2$ and add two fill-up nodes. and results from inputs $a = (0, 0, 0, 1), b = (0, 1, 1, 1)$ using the reduction $\mathcal{R}\mathbf{ed}$. Accordingly the dashed red edges represent a and b. To be more detailed, edge (l_0, l_2) represents $a(0) = 0$, edge (l_0, l_3) represents $a(1) = 0$, edge (l_1, l_2) represents $a(2) = 0$ and edge (r_0, r_2) represents $b(0) = 0$. This causes the diameter to be larger than 4 witnessed by $d(l'_1, r'_3) = 5$. Using Theorem 5.2 we conclude that a and b are not disjoint, which is indeed true.

Alice adds a node c_L to G_a that is connected to all nodes in L and Bob adds a node c_R to G_b that is connected to all nodes in R. Then 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. She is doing the same with the lower nodes of L by adding edges between all nodes l_ν and l_μ where $\nu, \mu \geq k(n)$. Similarly Bob adds (clique-) edges between all nodes r_ν and r_μ where $\nu, \mu < k(n)$ and $\nu, \mu \geq k(n)$ respectively. Furthermore, for each $i \in [k(n)^2 - 1]_0$, if $a(i) = 0$, Alice connects node $l_{i \bmod k(n)}$ from the upper half to node $l_{k(n)+\left\lfloor \frac{i}{k(n)} \right\rfloor}$ in the lower half by an edge. An example of this can be found in Figure 2 with detailed explanations in the caption. Note that this is the only part that depends on the input a and we can represent all values of the $[k(n)^2-1]_0$ bits of a by the $k(n)^2$ possible edges between the $k(n)$ nodes $\{l_\nu : \nu \in [k(n)-1]_0\}$ and the $k(n)$ nodes $\{l_\nu : \nu \in \{k(n), \ldots, 2k(n) - 1\}\}$. 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. So far G_a and G_b contain the $8k(n)$ nodes in L, L', R, R' as well as c_L and c_R. Therefore we add $n - 8k(n) - 2$ fill-up nodes $\{w_0, \ldots, w_{n-8k(n)-3}\}$ and edges (w_i, r_0) to G_b. This ensures that the final graph $m((G_a, C_{k(n)^2}), (G_b, C_{k(n)^2}))$ has exactly n nodes such that the lower bound holds for all n. More formally we have:

$$\begin{aligned}
V_a :=\ & L \cup L' \cup \{c_L\} \\
E_a :=\ & \bigcup_{\nu=0}^{2k(n)-1} \{(l_\nu, l'_\nu), (l_\nu, c_L)\} \\
& \cup \{(l_\nu, l_\mu) : \nu \neq \mu \wedge \nu, \mu \in [k(n)-1]_0\} \\
& \cup \{(l_\nu, l_\mu) : \nu \neq \mu \wedge \nu, \mu \in \{k(n), \ldots, 2k(n)-1\}\} \\
& \cup \{(l_{i \bmod k(n)}, l_{k(n)+\left\lfloor \frac{i}{k(n)} \right\rfloor}) : i \in [k(n)^2-1]_0, a(i) = 0\}
\end{aligned}$$

$$\begin{aligned}
V_b :=\ & R \cup R' \cup \{c_R\} \cup \{w_i : i \in [n - 8k(n) + 1]_0\} \\
E_b :=\ & \bigcup_{\nu=0}^{2k(n)-1} \{(r_\nu, r'_\nu), (r_\nu, c_R)\} \\
& \cup \{(w_i, r_0) : i \in [n - 8k(n) + 1]_0\} \\
& \cup \{(r_\nu, r_\mu) : \nu \neq \mu \wedge \nu, \mu \in [k(n)-1]_0\} \\
& \cup \{(r_\nu, r_\mu) : \nu \neq \mu \wedge \nu, \mu \in \{k(n), \ldots, 2k(n)-1\}\} \\
& \cup \{(r_{i \bmod k(n)}, r_{k(n)+\left\lfloor \frac{i}{k(n)} \right\rfloor}) : i \in [k(n)^2-1]_0, b(i) = 0\}
\end{aligned}$$

Finally we define the cut-set $C_{k(n)^2} := \{(l_\nu, r_\nu) : \nu \in [2k(n)-1]_0\}$ to consist of the $2k(n)$ edges connecting each l_ν to the corresponding r_ν and threfore G_a to G_b. Thus $\mathcal{R}$**ed** is a $2k(n)$-reduction. Observe that $(G_a, C_{k(n)^2})$ can be computed from a without knowing b and $(G_b, C_{k(n)^2})$ can be computed from b without knowing a, thus the reduction $\mathcal{R}$**ed** has the desired properties. Now we set $G_{a,b} := m((G_a, C_{k(n)^2}), (G_b, C_{k(n)^2}))$ (see Definition 4.3).

Lemma 5.1. *The graph $G_{a,b}$ is an n-node graph with diameter at most* 5.

Proof. The graph $G_{a,b}$ contains the $8k(n)$ nodes in L, L', R and 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 distinguish three cases:

1. Nodes u and v are both in G_a: To observe this we note 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. Nodes u and v are both in G_b: This case is completely analog to the previous, thus $d(u, v) \leq 4$.

3. Node u is in G_a and node v is in G_b (or the other way round): 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 $r_\mu \in R$. 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 relate the problem of deciding whether a and b are disjoint to the problem of computing the diameter of a graph. To achieve this we extend the analysis of the diameter of $G_{a,b}$.

Theorem 5.2. *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 $\nu := i \mod k(n)$ and $\mu = k(n) + \left\lfloor \frac{i}{k(n)} \right\rfloor$. We show that the two nodes l'_ν and r'_μ have distance at least 5. To observe this we first note that 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 any path. To obtain a path of length 4 we can 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 can 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 $d((l'_\nu, r'_\mu) > 4$. Combined with Lemma 5.1 this 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. Nodes u and v are both in G_a: Same as in proof of Lemma 5.1, thus $d(u, v) \leq 4$.

2. Nodes u and v are both in G_b: Same as in proof of Lemma 5.1, thus $d(u, v) \leq 4$.

3. Node u is in G_a and node v is in G_b (or the other way round): 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 $r_\mu \in R$. 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 is 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 $\mathrm{disj}_{\mathrm{k(n)}^2}$ function using any algorithm for $diam_5$ we use the reduction from $diam'_5$ to $diam_5$ presented in Section 4 and the reduction $\mathcal{R}\mathbf{ed}$ from $\mathrm{disj}_{\mathrm{k(n)}^2}$ to $diam'_5$ presented above. We can apply Theorem 4.1 and know that

$$\frac{R_\varepsilon^{cc-pub}(\mathrm{disj}_{\mathrm{k(n)}^2})}{2|C_{k(n)^2}| \cdot B} \leq R_\varepsilon^{dc-pub}(diam_5)$$

Due to Theorem 2.1 we know that $R_\varepsilon^{cc-pub}(\mathrm{disj}_{\mathrm{k(n)}^2})$ is at least $\Omega(k(n)^2)$. Together with the fact that $|C_{k(n)^2}| = 2k(n)$ we conclude that $R_\varepsilon^{dc-pub}(diam_5) = \Omega(k(n)/B)$. 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 $1 > \delta > 0$, $n \geq 12 \left\lceil \frac{3}{4\delta} \right\rceil + 8$ and $B \geq 1$ and sufficiently small ε, any distributed ε-error algorithm A that $(\frac{3}{2} - \delta)$-approximates 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 $16 \left\lceil \frac{3}{4\delta} \right\rceil + 4$.*

REMARK 6.1. *The diameter of the graph used to prove 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 diameters within a family of graphs. Graphs $G_{a,b}$ constructed from disjoint inputs a and b 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 in the construction of the graphs defines the length of "short paths" that we add to the graphs. We set this length to be $p_s := \left\lceil \frac{3}{4\delta} \right\rceil$. During the reduction we construct graphs $G_{a,b}$ from a and b such that any $(\frac{3}{2} - \delta)$-approximation algorithm for the diameter estimates $diam(G_{a,b})$ to be less than $6p_s$ if a and b are disjoint. If a and b are not disjoint the diameter (and thus the estimate) is always larger than $6p_s$. Since the reduction $\mathcal{R}\mathbf{ed}$ delivers the above promise-problem[3] we can just use the function $diam_{6p_s}$ as the decision-version of $(\frac{3}{2} - \delta)$-approximating the diameter.

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

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 h to be a function $k(n)^2$ depending on n and set $k(n) := \left\lfloor \sqrt{\frac{n}{12\lceil\frac{3}{4\delta}\rceil+8}} \right\rfloor$. This time we consider the $\mathrm{disj_{k(n)^2}}$ problem to be the base-function h. Now we need to define a reduction $\mathcal{R}\mathbf{ed}$ that given inputs a and b to h, maps $(Alice, a)$ and (Bob, b) to inputs $(G_a, C_{k(n)^2})$ and $(G_b, C_{k(n)^2})$ for $diam'_{6p_s}$. We define L, L', R and R'

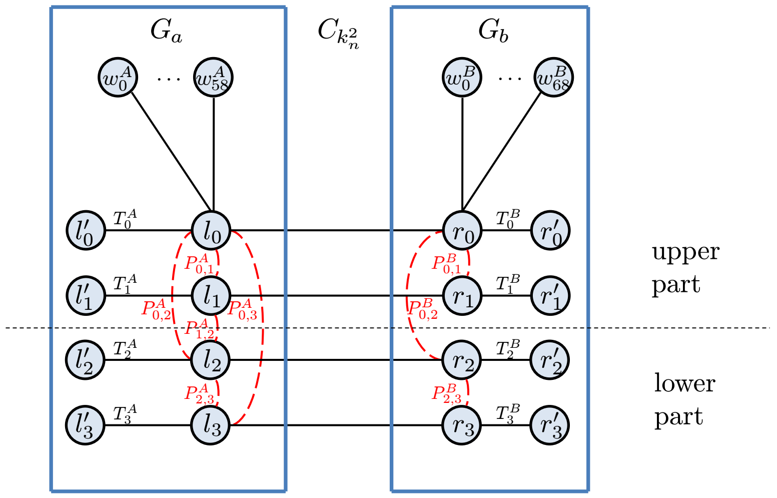


Figure 3: Graph used to calculate $\mathrm{disj_{k(n)^2}}$ when given a diameter estimator-algorithm. The graph in the example is for $n = 200$ and $\delta = 1/4$, thus $p_s = 3$ and $p_l = 6$ and $k(n) = 2$. Let the input strings be $a = (0, 0, 0, 1)$ and $b = (0, 1, 1, 1)$. The long Path $P^A_{0,2}$ represents $a(0) = 0$, the long path $P^A_{0,3}$ represents $a(1) = 0$, the long path $P^A_{1,2}$ represents $a(2) = 0$, the long path $P^B_{0,2}$ represents $b(0) = 0$. Note that only important nodes are explicitly displayed. In G_a we include $100 - \bar{n}_a = 59$ fill-up nodes, in G_b we include $100 - \bar{n}_b = 69$ fill-up nodes. Since the sets are not disjoint the diameter is larger than $6p_s$ as witnessed by $d(l'_1, r'_3) = 6p_s + 1$.

as in the previous section. Given inputs $a \in \{0,1\}^{k(n)^2}$ and $b \in \{0,1\}^{k(n)^2}$ Alice constructs G_a and Bob constructs G_b. For each $\nu \in [2k-1]_0$ Alice adds a short path T^A_ν of length p_s connecting nodes l_ν to l'_ν as depicted in Figure 3. Now Alice adds long paths $P^A_{\nu,\mu}$ of length $p_l := 2p_s$ for $\nu \neq \mu \in [k-1]_0$ and for $\nu \neq \mu \in \{k, 2k-1\}$. Furthermore for each $i \in [k(n)^2 - 1]_0$ with $a(i) = 0$ Alice computes $\nu := i \bmod k(n)$ and $\mu := k(n) + \left\lfloor \frac{i}{k(n)} \right\rfloor$ and adds a long path $P^A_{\nu,\mu}$ connecting l_ν to l_μ. The graph G_b is constructed by Bob in the same way using paths T^B_ν and $P^B_{\nu,\mu}$ that are added depending on b.

Now we set the cut-set $C_{k(n)^2}$ that connects G_a with G_b to be $C_{k(n)^2} := \cup_{i \in [2k(n)-1]_0}\{(l_i, r_i)\}$. Thus $\mathcal{R}\mathbf{ed}$ is a $2k(n)$-reduction. We could define $G_{a,b} := m((G_a, C_{k(n)^2}), (G_b, C_{k(n)^2}))$ (see Definition 4.3), but

[3]In a promise problem the input is promised to belong to a subset of all possible inputs.

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. let $\bar{n}_a$ denote the number nodes that were added to V_a so far. Then we know:

$$\begin{aligned}\bar{n}_a &= |L| + |L'| + \#\text{nodes in paths of } G_a \\ &= 2k(n) + 2k(n) + \underbrace{2k(n)\cdot(p_s-1)}_{\text{nodes in } T_i^A} + \underbrace{\frac{1}{2}(k(n)-1)k(n)(p_l-1)}_{\substack{\text{all-to-all connections between} \\ \text{nodes } l_0,\dots,l_{k(n)-1}}} \\ &\quad + \underbrace{\frac{1}{2}(k(n)-1)k(n)(p_l-1)}_{\substack{\text{all-to-all connections between} \\ \text{nodes } l_{k(n)},\dots,l_{2k(n)-1}}} + \underbrace{\sum_{i\in[k(n)^2-1]_0:a(i)=0}(p_l-1)}_{\text{paths depending on } a}\end{aligned}$$

and can bound this further by

$$\begin{aligned}&\leq 4k(n) + 2k(n)\cdot p_s + 2k(n)^2\cdot(p_l-1) \\ &\leq k(n)^2\cdot(6p_s+4) \\ &= \left\lfloor\sqrt{\frac{n}{12\left\lceil\frac{3}{4\delta}\right\rceil+8}}\right\rfloor^2\cdot\left(6\left\lceil\frac{3}{4\delta}\right\rceil+4\right) \leq n/2\end{aligned}$$

and can show $\bar{n}_b \leq n/2$ 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 fill-up nodes $w_0^A,\dots,w_{n/2-\bar{n}_a-1}^A$ to G_a (each connected by an edge to l_0) such that $|V_a| = n/2$ and as many fill-up nodes $w_0^B,\dots,w_{n/2-\bar{n}_b-1}^B$ to G_b (each connected by an edge to r_0) such that $|V_b| = n/2$. Note that in case n is odd, we can just add one more fill-up node e.g. to G_b.

Finally we set $G_{a,b} := m((G_a, C_{k(n)^2}), (G_b, C_{k(n)^2}))$ using Definition 4.3 of m.

LEMMA 6.1. *If $a = b = 1^{k(n)^2}$ the graph is disconnected. Otherwise the graph $G_{a,b}$ over n nodes has diameter at most $16p_s + 4$. Thus the lower bound on approximating the diameter is valid when communication distances are short.*

Note that it is likely that with a more detailed analysis the diameter can be bounded to be even smaller.

Proof. In case $a = b = 1^{k(n)^2}$ there are no connections between the nodes in the upper part, these are the nodes in $\{l_i, l_i', r_i, r_i', w_\nu^A, w_\mu^B : i \in [k-1]_0, \nu \in \{0,\dots,n/2-\bar{n}_a-1\}, \mu \in \{0,\dots,n/2-\bar{n}_b-1\}\}$, and the nodes in the lower part (these are all nodes not listed above). This is due to the construction which does not add any long path with end-points in both parts. However, these long paths are the only way to connect these parts.

For the second statement we can assume that it is not the case that $a = b = 1^{k(n)^2}$. 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. Node u is in the upper part of $G_{a,b}$: From u it is at most p_s hops to some node $l \in L$ (or $r \in R$). Due to the construction there is always 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. Node u is in the lower part of $G_{a,b}$: Since we assume that it is not the case that $a = b = 1^{k(n)^2}$ 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 we obtain $d(w, l_0) \le p_s + p_l + 1$. Applying this argument in the lower part we obtain $d(u, v) \le p_s + p_l + 1$. Thus, in total we have that $d(u, l_0) \le d(u, v) + d(v, w) + d(w, l_0) \le 2p_s + 3p_l + 2$ which in turn is less or equal to $8p_s + 2$.

Hence the diameter is at most $16p_s + 4$. note that this bound can be optimized.

LEMMA 6.2. *The sets a and b are disjoint, if and only if the estimate d' of the diameter of $G_{a,b}$ is less than $6p_s$.*

Proof. If a and b are not disjoint there exists an i such that $a(i) = b(i) = 1$. Let $\nu := i \mod k(n)$ and $\mu := k(n) + \left\lfloor \frac{i}{k(n)} \right\rfloor$. We show that the two nodes l'_ν and r'_μ have distance greater than $6p_s$. First observe that any path that connects them includes nodes l_ν and r_μ. Thus $d(l'_\nu, r'_\mu) = d(l_\nu, r_\mu) + 2p_s$. Now we argue that nodes l_ν and r_μ must have a distance greater than $2p_l = 4p_s$. This is since $a(i) = b(i) = 1$ implies that there is neither a direct path between l_ν to l_μ nor from r_ν to r_μ and one need to make a detour visiting another node $v \in L \cup R$. Due to the construction of $G_{a,b}$ this takes at least two long paths and one edge. Thus $d(l'_\nu, l'_\mu) \ge 2p_l + 2p_s + 1 = 6p_s + 1$. Which implies that the diameter d is at least $6p_s$. Therefore the estimate d' to the diameter produced by any approximation algorithm is larger or equal to $6p_s$ 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 $d(u, u') \le p_s$. Similarly there is a node $v' \in L \cup R$ such that $d(v, v') \le p_s$. Since a and b are disjoint the construction always yields a connection of length $1 + p_l$ between any two nodes in $u', v' \in L \cup R$. Thus $d = d(u, v) \le 2p_s + 2 + p_l = 4p_s + 2$. We obtain for the estimate d' for the diameter d of any $\frac{3}{2} - \delta$-approximation algorithm that

$$\begin{aligned} d' &\le \left(\frac{3}{2} - \delta\right) d \\ &= \left(\frac{3}{2} - \delta\right)(4p_s + 2) \\ &\le \left(\frac{3}{2} - \frac{3}{4p_s}\right)(4p_s + 2), \end{aligned}$$

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

Proof. (of Theorem 6.1). To solve the $\mathrm{disj}_{\mathrm{k(n)^2}}$ problem using any $(3/2 - \delta)$-approximation-algorithm for *diam* we use the reduction $\mathcal{R}\mathbf{ed}$ from $\mathrm{disj}_{\mathrm{k(n)^2}}$ to $diam'_{6p_s}$ and observed that $\mathcal{R}\mathbf{ed}$ 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 in order to apply lemma 6.1. Fortunately we can ignore this case since Alice and Bob can easily determine whether $a = b = 1^{k(n)^2}$ s.t. $D = \infty$ using 2 bits of communication. In this case G_a and G_b are not connected to each other. This does not affect our asymptotic lower bounds. According to Theorem 4.1, we know that

$$\frac{R_\varepsilon^{cc-pub}(\mathrm{disj}_{\mathrm{k(n)^2}})}{2|C_{k(n)^2}| \cdot B} \le R_\varepsilon^{dc-pub}(diam_{6p_s})$$

Due to Theorem 2.1 we know that $R_\varepsilon^{cc-pub}(\mathrm{disj}_{\mathrm{k(n)}^2})$ 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 h of size $k(n)$ we obtain $R_\varepsilon^{dc-pub}(diam_{6p_s}) = \Omega(k(n)/B)$. We obtain the result since we chose $k(n) := \left\lfloor \sqrt{\frac{n}{12\lceil\frac{3}{4\delta}\rceil+8}} \right\rfloor = \Theta(\sqrt{\delta n})$.

7 Girth Approximation Lower Bound

Theorem 7.1. *For any $\delta > 0$, $n \geq 32\left\lceil\frac{2}{\delta}\right\rceil - 4$ and $B \geq 1$ and sufficiently small ε, any distributed $(2-\delta)$-approximate 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 $\left(8\left\lceil\frac{2}{\delta}\right\rceil + 2\right)$*

Remark 7.1. *The diameter and girth of the graph used to prove the theorem above do 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 within a family of graphs. Graphs $G_{a,b}$ constructed from disjoint inputs a and b have a girth that is a factor shorter than the girth of graphs constructed from inputs that were not disjoint. Thus a good enough approximation-algorithm can distinguish them.

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

$$girth_{p_l}(G) := \begin{cases} 1 & : girth(G) < p_l \\ 0 & : \text{else} \end{cases}$$

In order to prove Theorem 7.1 we follow the high-level idea explained in Section 4 and derive a function $girth'_{p_l}$ from $girth_{p_l}$ as described in Section 4. To prove lower bounds depending on n, we choose the length k of the input to the base-function h to be a function $k(n)^2$ depending on n and set $k(n) := \left\lfloor \sqrt{\frac{n}{32\lceil\frac{2}{\delta}\rceil-4}} \right\rfloor$. As base-function h we consider the $\mathrm{disj}_{\mathrm{k(n)}^2}$ problem. Now we need to define a reduction $\mathcal{R}$**ed** that given inputs a and b to h, maps $(Alice, a)$ and (Bob, b) to inputs $(G_a, C_{k(n)^2})$ and $(G_b, C_{k(n)^2})$ for $girth'_{p_l}$. During the reduction $\mathcal{R}$**ed**, Alice defines the set of nodes L and Bob defines R as in the previous section. Given input $a \in \{0,1\}^{k(n)^2}$, Alice constructs G_a and given input $b \in \{0,1\}^{k(n)^2}$ Bob constructs G_b.

For each $i \in [k(n)^2 - 1]_0$, if the input is $a(i) = 1$, Alice adds a short path $P^A_{\nu,\mu}$ connecting l_ν to l_μ of length p_s (with $\nu = i \mod k(n)$ and $\mu = k(n) + \left\lfloor\frac{i}{k(n)}\right\rfloor$). If $a(i) = 0$, Alice adds a long path $P^A_{\nu,\mu}$ of length p_l. Alice also adds a long path $P^A_{\nu,\mu}$ of length p_l between all pairs l_ν and l_μ that were not already connected above. An example for this is displayed in Figure 4. In a similar way Bob adds paths $P^B_{\nu,\mu}$ to G_b depending on b. Thus $\mathcal{R}$**ed** is a $2k(n)$-reduction.

We set the cut-set $C_{k(n)^2}$ that connects G_a with G_b to be $C_{k(n)^2} := \cup_{i\in[2k-1]_0}\{(l_i, r_i)\}$. Now we could define $G_{a,b} := m((G_a, C_{k(n)^2}), (G_b, C_{k(n)^2}))$ (see Definition 4.3), but observe that the number of nodes that we added so far to V_a and V_b is not necessarily n yet. Denote by $\bar{n}_a$ the number of nodes that have

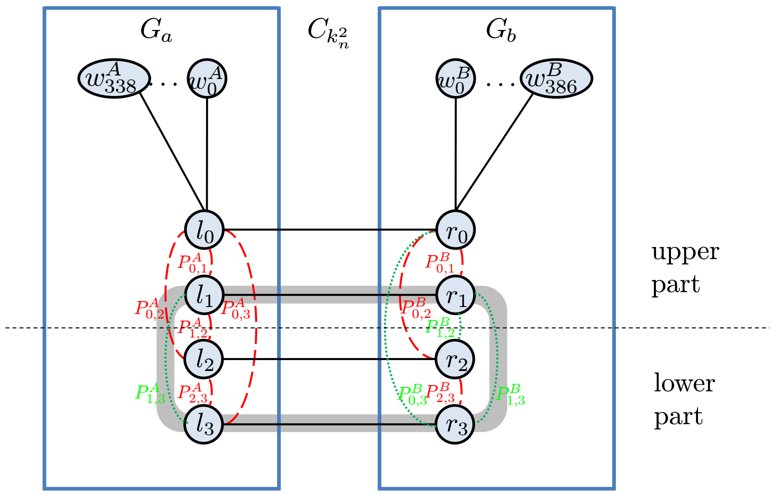


Figure 4: Graph used to calculate $\mathrm{disj}_{\mathrm{k(n)^2}}$ when given a girth estimator-algorithm. The graph in the example is for $n = 1010$ (only important nodes are explicitly displayed) and $\delta = 1/4$, thus $k(n) = 2$, $p_s = 8$ and $p_l = 32$. We use $a = (0, 0, 0, 1)$ and $b = (0, 1, 1, 1)$. The green dotted lines represent short paths; the red dashed lines represent long paths. To be more detailed, the long $l_0 - l_2$-path $P^A_{0,2}$ represents $a(0) = 0$, the long $l_0 - l_3$-path $P^A_{0,3}$ represents $a(1) = 0$, the long $l_1 - l_2$-path $P^A_{1,2}$ represents $a(2) = 0$ and the short $l_1 - l_3$-path $P^A_{1,3}$ represents $a(3) = 1$. The long $r_0 - r_2$-path $P^B_{0,2}$ represents $b(0) = 0$, the short $r_0 - r_3$-path $P^B_{0,3}$ represents $b(1) = 1$, the short $r_1 - r_2$-path $P^B_{1,2}$ represents $b(2) = 1$, the short $r_1 - r_3$-path $P^B_{1,3}$ represents $b(3) = 1$. In this example, G_a contains 5 long paths and 1 short path. Since G_a should contain 505 nodes, $505 - \bar{n}_a = 339$ fill-up nodes are in G_a. On the other side, G_b contains 3 long paths and 3 short path. Since G_b should contain 505 nodes as well, $505 - 4 - \bar{n}_b = 387$ fill-up nodes are in G_b. Since the sets are not disjoint, there is a cycle of length $2p_s + 2$ involving edges (l_1, r_1) and (l_3, r_3), as well as paths $P^A_{1,3}$ and $P^B_{1,3}$. This cycle is indicated by a light-gray background. Note, that the constants in the proof could be improved.

been added to V_a so far. We have that

$$\begin{aligned}\bar{n}_a &= |L| + \text{nodes in paths of } G_a \\ &= 2k(n) \quad + \quad \underbrace{(p_s - 1) \cdot \sum_{i \in [k(n)^2 - 1]_0} a(i)}_{\text{nodes in short paths}} \quad + \quad \underbrace{(p_l - 1) \cdot \left(\frac{1}{2}(2k(n))(2k(n) - 1) - \sum_{i \in [k(n)^2 - 1]_0} a(i) \right)}_{\text{nodes in long paths}},\end{aligned}$$

which can be bounded by $2k(n) + (p_l - 1) \cdot k(n)(2k(n) - 1)$. Since $p_l \geq 4$, we can further bound this to be

$$\begin{aligned}&\leq 2k(n)^2 \cdot (p_l - 1) \\ &= 2 \cdot \left\lfloor \sqrt{\frac{n}{32 \left\lceil \frac{2}{\delta} \right\rceil - 4}} \right\rfloor^2 \cdot \left(4 \left\lceil \frac{2}{\delta} \right\rceil - 1 \right) \\ &\leq n/2\end{aligned}$$

One can show $\bar{n}_b \leq n/2$ 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 fill-up nodes $w_0^A, \ldots, w_{\bar{n}_a - n/2-1}^A$ to G_a (each connected by an edge to l_0) such that $|V_a| = n/2$ and fill-up nodes $w_0^B, \ldots, w_{n/2-\bar{n}_b-1}^B$ to G_b (each connected by an edge to r_0) such that $|V_b| = n/2$. Note that in case n is odd, we can just add one more fill-up node e.g. to G_b.

Finally we set $G_{a,b} := m((G_a, C_{k(n)^2}), (G_b, C_{k(n)^2}))$ (see Definition 4.3).

Lemma 7.1. *Any graph $G_{a,b}$ as constructed above has diameter at most $8\left\lceil\frac{2}{\delta}\right\rceil + 2$. Thus the lower bound on approximating the girth is valid even when communication distances are short.*

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 (e.g. if u is in the middle of a long path in G_a). Since nodes u' and v' are in $L \cup R$ they are connected by a path $P_{\nu,\mu}^A$ or $P_{\nu,\mu}^B$ (with appropriate ν, μ) of length p_l and one edge that connects L and R. Therefore $G_{a,b}$ has diameter at most $2p_l + 2 = 8\left\lceil\frac{2}{\delta}\right\rceil + 2$.

Lemma 7.2. *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 is a cycle of length $2p_s + 2$ causing that any $(2-\delta)$-estimate g' is less than p_l. Later we show that if a and b are disjoint, there is 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

$$\left(P_{\nu,\mu}^A, (r_\nu, l_\nu)\right), P_{\nu,\mu}^B, (l_\mu, r_\mu))$$

consisting of two short paths and two edges has length $(2p_s + 2)$ 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 p_s. Inserting the value of g we obtained above, we get $g' \leq \left(2 - \frac{2}{p_s}\right)(2p_s + 2) = 4p_s - \frac{4}{p_s} < p_l$

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

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.

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), and two nodes ν, μ can only be connected by one path, e.g. $P_{\nu,\mu}^A$, the cycle must contain at least 4 short paths, see Figure 5. Hence the girth g of $G_{a,b}$ and therefore its estimate g' is at least $4p_s = p_l$.

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 case.

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_ν, l_ν), (r_μ, l_μ). If both 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 are in the upper/lower part and can be connected by one short path each. Therefore the cycle contains at least 4 short paths. Conversely if ν is smaller than $k(n)$ and μ is at least $k(n)$ or the other way around, it is

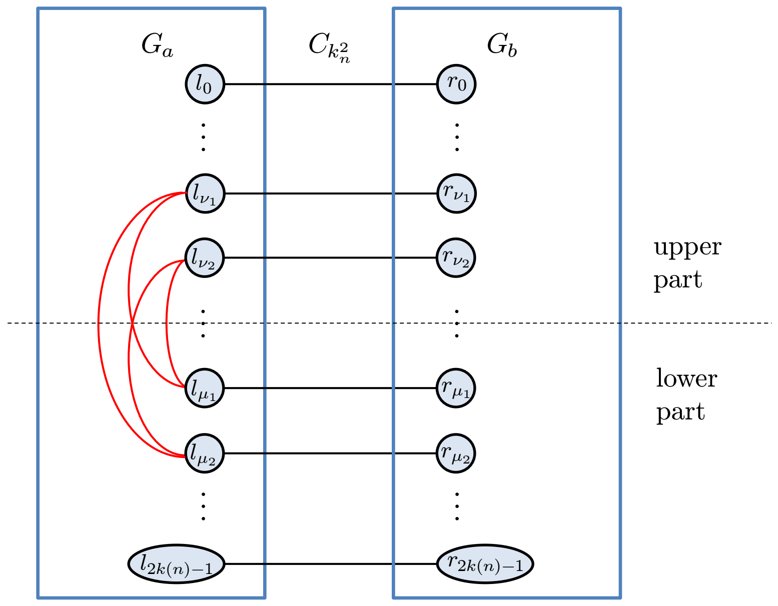


Figure 5: Case 2 of Lemma 7.2: A smallest cycle within G_a that contains no path of length p_l must contain 4 paths of length p_s.

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, see Figure 6. Hence the girth and estimator of $G_{a,b}$ is at least $4p_s + 2 > p_l$.

Proof. (of Theorem 7.1). To solve the $\mathrm{disj}_{\mathrm{k(n)^2}}$ problem using any $(2-\delta)$-approximation algorithm for *girth* we use the reduction $\mathcal{R}$**ed** from $\mathrm{disj}_{\mathrm{k(n)^2}}$ to $girth_{p_l}$ and observed that $\mathcal{R}$**ed** delivers a promise-problem such that there is a reduction from $girth_{p_l}$ to *girth* via $girth'_{p_l}$. Now we apply Lemma 7.2 and use Theorem 4.1 to derive that

$$\frac{R_\varepsilon^{cc-pub}(\mathrm{disj}_{\mathrm{k(n)^2}})}{2|C_{k(n)^2}| \cdot B} \leq R_\varepsilon^{dc-pub}(girth_{p_l})$$

Due to Theorem 2.1 we know that $R_\varepsilon^{cc-pub}(\mathrm{disj}_{\mathrm{k(n)^2}})$ 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 h of size $k(n)$ it is $R_\varepsilon^{dc-pub}(girth_{p_l}) \in \Omega(k(n)/B)$. We obtain the stated result since we chose $k(n) := \left\lfloor \sqrt{\frac{n}{32\lceil\frac{2}{\delta}\rceil - 4}} \right\rfloor$.

Acknowledgements: We thank the anonymous reviewers for helpful comments and suggestions.

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 (SICOMP)*, 28(4):1167–1181, 1999.

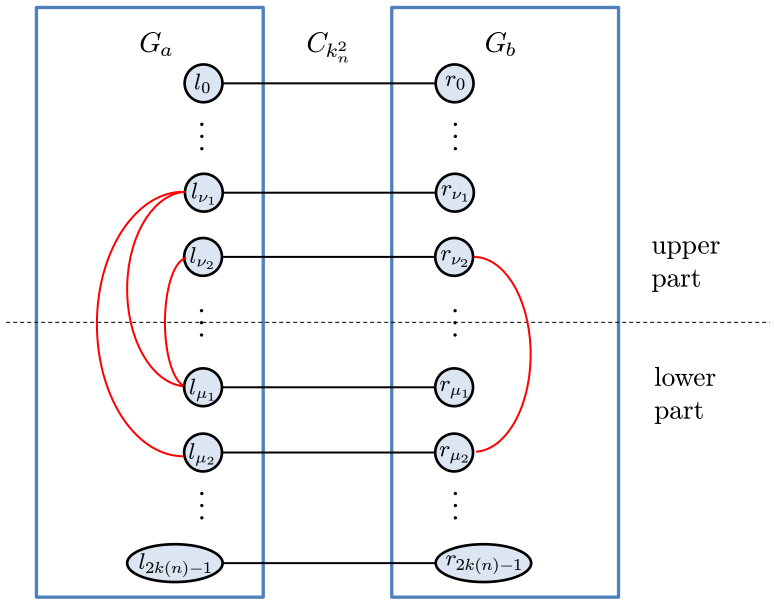


Figure 6: Second part of Case 4 of Lemma 7.2: A smallest cycle that contains nodes from both, G_a and G_b, but no path of length p_l, must contain 4 paths of length p_s. In this example it also contains edges (r_{ν_2}, l_{ν_2}), (r_{μ_2}, l_{μ_2})

[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. Galilt, and M. Naor. Witnesses for boolean matrix multiplication and for shortest paths. In *Proceedings of the 33rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 417–426, 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, 2008.

[5] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *Journal of symbolic computation (JSC)*, 9(3):251–280, 1990.

[6] A. Das Sarma, S. Holzer, L. Kor, A. Korman, D. Nanongkai, G. Pandurangan, D. Peleg, and R. Wattenhofer. Distributed verification and hardness of distributed approximation. *Proceedings of the 43rd annual ACM Symposium on Theory of Computing (STOC)*, 2011.

[7] M. Elkin. Unconditional lower bounds on the time-approximation tradeoffs for the distributed minimum spanning tree problem. In *Proceedings of the 36th annual ACM symposium on Theory of computing (STOC)*, pages 331–340, 2004.

[8] F. Fich and E. Ruppert. Hundreds of impossibility results for distributed computing. *Distributed computing*, 16(2):121–163, 2003.

[9] S. Frischknecht, S. Holzer, and R. Wattenhofer. Networks Cannot Compute Their Diameter in Sublinear Time. In *Proceedings of the 23rd annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1150–1162.

[10] J.A. Garay, S. Kutten, and D. Peleg. A sublinear time distributed algorithm for minimum-weight spanning trees. *SIAM Journal on Computing (SICOMP)*, 27(1):302–316, 1998.

[11] R.L. Graham, A.C.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.

[12] 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, 2000.

[13] S. Holzer, D. Peleg, L. Roditty, E. Tal, and R. Wattenhofer. Optimal distributed all pairs shortest paths and applications. *http://www.dcg.ethz.ch/ stholzer/APSP-full.pdf (preliminary full version of two merged papers to be submitted to a journal).*

[14] S. Holzer and R. Wattenhofer. Optimal distributed all pairs shortest paths and applications. In *Proceedings of the 31st annual ACM symposium on Principles of distributed computing (PODC)*, pages 355–364. ACM, 2012.

[15] D.R. Karger, D. Koller, and S.J. Phillips. Finding the hidden path: Time bounds for all-pairs shortest paths. *SIAM Journal on Computing (SICOMP)*, 22:1199, 1993.

[16] L.R. Kerr. The effect of algebraic structure on the computational complexity of matrix multiplication, cornell university, phd thesis. 1970.

[17] L. Kor, A. Korman, and D. Peleg. Tight Bounds For Distributed MST Verification. In Thomas Schwentick and Christoph Dürr, editors, *Proceedings of the 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.

[18] F. Kuhn, T. Moscibroda, and R. Wattenhofer. What cannot be computed locally! In *Proceedings of the 23rd Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC)*, pages 300–309, 2004.

[19] Fabian Kuhn and Roger Wattenhofer. Constant-time distributed dominating set approximation. *Distributed Computing*, 17(4):303–310, 2005.

[20] E. Kushilevitz and N. Nisan. *Communication complexity.* Cambridge University Press, 1997.

[21] C. Lenzen and D. Peleg. Efficient distributed source detection with limited bandwidth. page pages not known yet, 2013.

[22] Christoph Lenzen and Boaz Patt-Shamir. Fast routing table construction using small messages. *arXiv preprint arXiv:1210.5774*, 2012.

[23] Z. Lotker, B. Patt-Shamir, and D. Peleg. Distributed mst for constant diameter graphs. In *Proceedings of the 20th Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC)*, pages 63–71, 2001.

[24] D. Nanongkai, A. Das Sarma, and G. Pandurangan. A tight unconditional lower bound on distributed random walk computation. In *Proceedings of the 30th Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC)*, pages 257–266, 2011.

[25] D. Peleg. *Distributed Computing: A Locality-Sensitive Approach.* SIAM, 2000.

[26] D. Peleg. *Distributed computing: a locality-sensitive approach.* 2000.

[27] D. Peleg, L. Roditty, and E. Tal. Distributed algorithms for network diameter and girth. In *Proceedings of the 39th International Colloquium on Automata, Languages and Programming (ICALP), to appear*, 2012.

[28] 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 (SICOMP)*, 30(5):1427–1442, 2001.

[29] S. Pettie. A new approach to all-pairs shortest paths on real-weighted graphs* 1. *Theoretical Computer Science (TCS)*, 312(1):47–74, 2004.

[30] S. Pettie and V. Ramachandran. A shortest path algorithm for real-weighted undirected graphs. *SIAM Journal on Computing (SICOMP)*, 34(6):1398–1431, 2005.

[31] L. Roditty and V.V. Williams. Fast approximation algorithms for the diameter and radius of sparse graphs. *Proceedings of the 45th annual ACM symposium on Theory of computing (STOC), to appear*, 2013.

[32] A.D. Sarma, M. Dinitz, and G. Pandurangan. Efficient computation of distance sketches in distributed networks. *24th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA), to appear*, 2012.

[33] R. Seidel. On the all-pairs-shortest-path problem in unweighted undirected graphs. *Journal of Computer and System Sciences (JCSS)*, 51(3):400–403, 1995.

[34] M. Thorup. Undirected single-source shortest paths with positive integer weights in linear time. *Journal of the ACM (JACM)*, 46(3):362–394, 1999.

[35] A.C.C. Yao. Some complexity questions related to distributive computing. In *Proceedings of the 11th annual ACM symposium on Theory of computing (STOC)*, pages 209–213, 1979.