Approximation of Distances and Shortest Paths in the Broadcast Congest Clique*

Stephan Holzer[1] and Nathan Pinsker[2]

1 **Massachusetts Institute of Technology (MIT), Cambridge, USA**
holzer@mit.edu
1 **Massachusetts Institute of Technology (MIT), Cambridge, USA**
npinsker@mit.edu

Abstract

We study the broadcast version of the CONGEST-CLIQUE model of distributed computing. This model operates in synchronized rounds; in each round, any node in a network of size n can send the same message (i.e. broadcast a message) of limited size to every other node in the network. Nanongkai presented in [STOC'14 [25]] a randomized $(2 + o(1))$-approximation algorithm to compute all pairs shortest paths (APSP) in time[1] $\tilde{\mathcal{O}}(\sqrt{n})$ on weighted graphs. We complement this result by proving that any randomized $(2 - o(1))$-approximation of APSP and $(2 - o(1))$-approximation of the diameter of a graph takes $\tilde{\Omega}(n)$ time in the worst case. This demonstrates that getting a negligible improvement in the approximation factor requires significantly more time. Furthermore this bound implies that already computing a $(2 - o(1))$-approximation of all pairs shortest paths is among the hardest graph-problems in the broadcast-version of the CONGEST-CLIQUE model, as any graph-problem where each node receives a linear amount of input can be solved trivially in linear time in this model. This contrasts a recent $(1 + o(1))$-approximation for APSP that runs in time $\mathcal{O}(n^{0.15715})$ and an exact algorithm for APSP that runs in time $\tilde{\mathcal{O}}(n^{1/3})$ in the unicast version of the CONGEST-CLIQUE model, a more powerful variant of the broadcast version.

This lower bound in the broadcast CONGEST-CLIQUE model is derived by first establishing a new lower bound for $(2 - o(1))$-approximating the diameter in weighted graphs in the CONGEST model, which is of independent interest. This lower bound is then transferred to the CONGEST-CLIQUE model.

On the positive side we provide a deterministic version of Nanongkai's $(2+o(1))$-approximation algorithm for APSP [25]. To do so we present a fast deterministic construction of small hitting sets. We also show how to replace another randomized part within Nanongkai's algorithm with a deterministic source-detection algorithm designed for the CONGEST model in [21].

1998 ACM Subject Classification C.2.1 Distributed networks

Keywords and phrases distributed computing, distributed algorithms, approximation algorithms

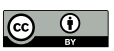
Digital Object Identifier 10.4230/LIPIcs.OPODIS.2015.6

1 Introduction

In a distributed message passing model a network is classically represented as a graph. In this graph any node can send (pass) one message to its neighbors in every round. There are two major research directions concerning message passing models.

* Work supported by the following grants: AFOSR Contract Number FA9550-13-1-0042, NSF Award 0939370-CCF, NSF Award CCF-1217506, NSF Award number CCF-AF-0937274.

[1] We use the convention that $\tilde{\Omega}(f(n))$ is essentially $\Omega(f(n)/\text{polylog}f(n))$ and $\tilde{\mathcal{O}}(f(n))$ is essentially $\mathcal{O}(f(n)\text{polylog}f(n))$.



The first research direction deals with determining the locality and congestion of problems. Using the LOCAL model [29], where message-size is unbounded, one tries to characterize the locality of problems, which is the ability of a node to make decisions regarding a problem purely based on information on its local neighborhood in a graph. Using the CONGEST model [29], one tries to characterize the delays caused by congestion. We assume for this model that message sizes are bounded to $\mathcal{O}(\log n)$ bits per round, and weights are encodable in $\mathcal{O}(\log n)$ bits. Congestion arises due to bottlenecks in the network that do not provide enough bandwidth during the computation. Both, the LOCAL model and the CONGEST model are classic models that have received a great deal of attention in the past decades. Recently it was pointed out in [28] that the CONGEST model does not avoid interference from locality issues, while the LOCAL model avoids interference from congestion. To be more precise, congestion is completely avoided in the LOCAL model due to unlimited bandwidth. On the other hand the complexity of algorithms in the CONGEST model may still depend on the local structure of a graph (e.g. lower bounds transfer from the LOCAL model). To truly separate the study of congestion from locality, one needs to consider networks that avoid locality issues. These are e.g. networks in which each node is directly connected to any other node in the network (represented by a clique), which is a network in which any graph problem can be solved within one round in case unlimited bandwidth is available. Such a model was introduced earlier by Lotker et al. [23] with the intention to study overlay networks that have this property and was coined the CONGEST-CLIQUE model. Examples of parallel systems design that recently provided additional motivation to this original motivation to study the CONGEST-CLIQUE as an overlay network [23] are included in Section 2. Note that in the broadcast setting, a simple algorithm can solve the vast majority of problems in linear time: each node v can simply broadcast the IDs of all of v's neighbors and weights of incident edges in time $\mathcal{O}(n)$. Then each node in the network has full information on the graph and can perform any computation (including e.g. NP-complete problems) internally, which does not contribute to the runtime.

The second research direction focuses on determining the power of broadcast compared to (multi-)unicast. Broadcast denotes the setting in which a node can only send the same message to all its neighbors at the same time, while in a (multi-)unicast setting each node can send different messages to different neighbors at the same time.

Results of this paper push both research directions. To be more precise, we present a linear lower bound and new improved bounds for a broadcast model (the BCC model, see definition below) that purely studies congestion.

▶ **Definition 1.** When applied to the CONGEST-CLIQUE model, we denote by UCC model the (multiple-)unicast version of the CONGEST-CLIQUE model, and by BCC model the broadcast version of the CONGEST-CLIQUE model [9, 23].

1.1 Contribution

In this context, this paper extends the work of [9, 15, 25]. Drucker, Kuhn and Oshman [9] started to study the difference in computational power between the UCC and the BCC models. Like [9] we present a linear lower bound in the BCC model. The lower bounds of [9] were the first deterministic (and conditional randomized) linear lower bounds in this model. Their lower bounds consider subgraph detection. Ours are the first unconditional randomized linear lower bounds, while we consider $(2 - o(1))$-approximations of APSP and diameter. Three main conclusions from this result are:

Table 1 Summary of new and previous results for problems we study on positively weighted graphs in the BCC model. Recent results of [4] in the UCC model are summarized in Section 2.

approx. factor	APSP	Diameter	SSSP
1	$\mathcal{O}(n)^{\#}$	$\mathcal{O}(n)^{\#}$	$\widetilde{\mathcal{O}}(\sqrt{n})^{*}$
$2-o(1)$	$\widetilde{\mathcal{O}}(n)^{\#}, \widetilde{\Omega}(n)^{\dagger}$	$\widetilde{\mathcal{O}}(n)^{\#}, \widetilde{\Omega}(n)^{\dagger}$	—
2	—	$\widetilde{\mathcal{O}}(\sqrt{n})^{*}$	—
$2+o(1)$	$\widetilde{\mathcal{O}}(\sqrt{n})^{\ddagger}$	$\widetilde{\mathcal{O}}(\sqrt{n})^{\ddagger}$	—

#) Trivial bound: collect the whole topology in a single node, perform computation internally.
∗) Nanongkai's SSSP algorithm [25]. See Remark 14 for the diameter approximation.
†) Our randomized lower bound, see Theorem 12.
‡) Our deterministic version of the randomized algorithm of [25], see Theorem 19.

- There is a (at least quadratic) difference in the complexity between computing a $(2+o(1))$-approximation [25] and a $(2-o(1))$-approximation of APSP in this model.
- Computing a $(2-o(1))$-approximation of APSP is among the hardest graph-problems in the BCC model, as any graph-problem (with $\mathcal{O}(\log n)$-encodable weights) can be solved in linear time.
- There is a clear separation between the UCC and BCC model with respect to APSP computation. Our lower bounds contrast the results of [4], who showed that e.g. even exact APSP can be solved in the UCC model within $\tilde{\mathcal{O}}(n^{1/3})$ time and $(1+o(1))$-approximated in time $\mathcal{O}(n^{0.15715})$.

Note that this lower bound strengthens the $\tilde{\Omega}(\sqrt{n})$ lower bound for exact computation of APSP in the BCC model by [4] in terms of runtime and extends it to approximations.

Technical Overview: To obtain our lower bounds, we first use techniques of Frischknecht et al. [10] to derive an $\tilde{\Omega}(n)$-round lower bound to $(2-o(1))$-approximate the diameter of weighted graphs in the CONGEST model. This implies an $\tilde{\Omega}(n)$-round lower bound to $(2-o(1))$-approximate APSP. To prove our lower bounds, we modify a construction for unweighted graphs that was claimed in [14] and can also be found in [32] to the weighted setting. We then use this construction to transfer lower bounds for set disjointness from two-party communication complexity (first studied by Kushilevitz [20]). Next we transfer this lower bound from the CONGEST model to the BCC model.

Apart from these lower bounds, we derive positive results on computing APSP by extending the line of work of [15, 25]. We start by replacing the randomized parts of a recent result by Nanongkai [25], who presented an algorithm for a $(2+o(1))$-approximation of the all-pairs shortest paths problem in the BCC model in $\tilde{\mathcal{O}}(\sqrt{n})$ rounds, with deterministic ones. We then show that the resulting algorithm can be transferred to the UCC model with an improvement in runtime.

1.2 Structure of the Paper

We review related work in Section 2 and define the computation models and terminology that we work with in Section 3. Our lower bounds are presented in Section 4, where present a review of two-party communication complexity, state the lower bounds for the CONGEST model and transfer them to the BCC model. A key-ingredient for our upper bounds is a

deterministic hitting set construction, which we present in Section 5. Finally, in Section 6, we present our deterministic version of Nanongkai's all-pairs shortest paths approximation algorithm in the BCC model. We conclude by briefly mentioning some open problems and directions for future work in Section 7.

2 Related Work

Algorithms in the BCC and UCC models: The first to study the CONGEST-CLIQUE model were Lotker et al. [23], where they presented an $\mathcal{O}(\log\log n)$-round algorithm for constructing a minimum spanning tree in the UCC model. This was improved by Pemmaraju and Sardeshmukh to $\mathcal{O}(\log\log\log n)$ in [30]. Lenzen obtained in [22] an $\mathcal{O}(1)$-round algorithm in the UCC model for simultaneously routing n messages per vertex to their assigned destination nodes, as well as an $\mathcal{O}(1)$ algorithm for sorting $\mathcal{O}(n^2)$ numbers, given that each vertex begins the algorithm knowing $\mathcal{O}(n)$ numbers. Independently Patt-Shamir and Teplitsky [28] showed a similar, but slightly weaker result on sorting in the UCC model. Later Hegeman et al. [13] provided constant and near-constant (expected) time algorithms for problems such as computing a 3-ruling set, a constant-approximation to metric facility location, and (under some assumptions) a constant-factor approximations to the minimum spanning tree in the UCC model. Holzer [14] provided a deterministic $\mathcal{O}(\sqrt{n})$-algorithm for exact unweighted SSSP (equivalent to computing a breadth first search tree) in the BCC model. Independently Nanongkai [25] provided randomized (w.h.p.) algorithms in the BCC model that take $\tilde{\mathcal{O}}(n^{1/2})$ rounds to compute (exact) SSSP, and $\tilde{\mathcal{O}}(n^{1/2})$ rounds to $(2+o(1))$-approximate APSP on weighted graphs. Much of our work for deterministic APSP builds off [25], primarily on his idea of "shortcut edges", which do not change the weighted shortest path length between any two nodes but decrease the diameter of the graph. This is combined with a deterministic h-hop multi-source shortest paths scheduling technique implied by the source-detection algorithm of Lenzen and Peleg [21], which works in the broadcast version of the CONGEST model. Note that other versions that could have been used, such as the one presented in [7, 14], only work in the (multi-)unicast version. Recently Censor-Hillel, Kaski, et al. [4] transferred fast matrix multiplication algorithms into the UCC model using results from [22] and derived a runtime of $\mathcal{O}(n^{1/3})$ in semirings and $\mathcal{O}(n^{0.15715})$ in rings. Using this they obtain an $\mathcal{O}(n^{0.15715})$ algorithm for triangle detection and undirected unweighted APSP. Both papers also solve APSP on directed weighted graphs in time $\tilde{\mathcal{O}}(n^{1/3})$. In addition [4] presents an $(1+o(1))$-approximation for exact directed weighted APSP in time $\mathcal{O}(n^{0.15715})$, while [4] derives results for fast diameter and girth computation as well as for 4-cycle detection.

Lower bounds in the BCC and UCC models: Drucker et al. [9] were the first to provide lower bounds in the BCC model. They derived these bounds by transferring lower bounds for set disjointness in the 3-party NOF model to the congested clique. In addition [9] showed that "a slightly super-constant lower bound on the number of rounds required to compute some explicit function in the unicast CONGEST-CLIQUE model (when message size is 1) would imply a new lower bound on ACC (constant depth circuits), and an $\Omega(\log\log n)$ lower bound for the unicast CONGEST-CLIQUE model would imply new a lower bound for threshold circuits (the class TC). While explicit lower bounds in the UCC model remain open and might have a major impact to other fields of (Theoretical) Computer Science as mentioned above, they argue nonconstructively that most problems have a linear lower bound in the UCC model with a counting argument. Independent and simultaneously to us, the

authors of [4] presented an $\tilde{\Omega}(\sqrt{n})$ lower bound for APSP in the BCC model, which they derive from matrix multiplication lower bounds that they state.

Lower bounds in the CONGEST model (all – including ours – in the unicast version): Frischknecht et al. [10] (which is based on [5]) showed an $\tilde{\Omega}(n)$ lower bound for exact computation of the diameter of an unweighted graph. In this paper we draw on these ideas to obtain lower bounds for (2-o(1))-approximating the diameter of weighted graphs in the BCC model. Note that also Nanongkai [25] presents an $\tilde{\Omega}(n)$-time lower bound for any $poly(n)$-approximation algorithm for APSP on weighted graphs in the CONGEST model and shows that any $\alpha(n)$-approximation of APSP on unweighted graphs requires $\tilde{\Omega}(n/\alpha(n))$ time. However, his proof relies on an information-theoretic argument and uses a star-shaped graph such that it cannot be extended to the BCC model, as in this model every node could simply broadcast its distance from the center to all other nodes. Assuming a girth conjecture, Izumi and Wattenhofer show in [17] that constructing distance oracles with stretch $2t$ in unweighted (weighted) graphs takes $\Omega(n^{1/(t+1)})$ rounds ($\Omega(n^{\frac{1}{2}+\frac{5}{t}})$ rounds). When $o(n^{\epsilon})$ label size is required, assuming the girth-conjecture can be dropped. In contrast to this, our lower bound does not assume relabeling. Our construction and the construction of [17] build on top of [10] and appeared at the same time: [17] and the technical report [16].

Connections to systems and other models: Finally we want to provide examples of parallel systems that might benefit from theoretical results in the CONGEST-CLIQUE model. These include systems that provide all-to-all communication between $10,000$ nodes at full bandwidth [27]. In addition [12] showed a close connection between the UCC model and popular parallel systems such as MapReduce [6] and analyzed which kind of algorithms for the UCC model can be simulated directly in MapReduce. Pregel [24] is a system that simulates algorithms designed for message-passing models such as the CONGEST model (the input graph is split among several machines). Klauk et al. [19] study large-scale graph processing systems such as Pregel [24] in a theoretic way (k-machine model), which also includes the CONGEST-CLIQUE model. Finally, the authors of [9] pointed out that the BCC model is used in streaming [1], cryptology [11] and mechanism design [8]. They also establish connections between the UCC and ACC as well as TC0 circuits.

3 Model and Definitions

We first introduce the CONGEST model and then derive the CONGEST-CLIQUE model, which is at the center of this paper.

The CONGEST Model: Our network is represented by an undirected graph $G = (V, E)$, where nodes V model processors or computers and edges E model links between the processors. Edges can have associated *weights* $w : E \to \{a/p \mid a \in \{1, \dots, p^2\} \subset \mathbb{N}\}$ for some $p \in poly(n)$. This ensures that each weight is a positive multiple of $1/p$ and can be encoded in $\mathcal{O}(\log n)$ bits. Two nodes can communicate directly with each other if and only if they are connected by some edge from set E. We also assume that the nodes have unique IDs in the range of $\{1, \dots, poly(n)\}$ and infinite computational power.[2] At the beginning, each node knows only the IDs of its neighbors and the weights of its incident edges.

[2] This assumption is made by the model because it is used to study communication complexity. Note that we do not make use of this, as our algorithms perform efficient computations.

We consider a model where nodes can send messages to their neighbors over synchronous rounds of communication. During a round, each node u can send a message of B bits through each edge connecting u to some other vertex v. We assume $B = \mathcal{O}(\log n)$ during our algorithms, which is the standard choice [29] and state our lower bounds depending on arbitrary B. The message will arrive at node v at the end of the round. We analyze the performance of an algorithm in this model by measuring the worst-case number of communication rounds required for the algorithm to complete.

Let $\mathcal{A}$ be the set of distributed deterministic algorithms that evaluate a function g on an underlying graph $G \in \mathbb{G}_n$ over n nodes, where $\mathbb{G}_n$ is the set of connected graphs over these nodes. We define the *distrubuted round complexity* of an integer-valued function g as follows:

▶ **Definition 2** (Distributed Round Complexity)**.** The distributed round complexity $R^{dc}(g)$ is defined to be $\min_{A\in\mathcal{A}} \max_{G\in\mathbb{G}_n} R^{dc}(A(G))$. In other words, $R^{dc}(A(G))$ represents the number of rounds that an algorithm $A \in \mathcal{A}$ needs in order to compute $g(G)$.

We denote by $R_{\varepsilon}^{dc-pub}(g)$ the (public coin[3]) randomized round complexity of g when the algorithms have access to public coin randomness and compute the desired output with an error probability smaller than ε.

The CONGEST-CLIQUE Model: In this model every vertex in a network G can directly communicate with every other vertex in G. Note that although the communication graph is a clique, we are interested in solving a problem on a subgraph G of the clique. Working under the broadcast and (multi-)unicast versions of the CONGEST model while making this assumption gives us the BCC model and the UCC model, respectively.

Problems and Definitions: For any nodes u and $v \in V$, a *(u,v)-path* P is a path $(u = x_0, x_1, \ldots, x_l = v)$ where $(x_i, x_{i+1}) \in E$ for all i. We define the weight of a path P to be $w(P) := \sum_{i=0}^{l-1} w(x_i, x_{i+1})$. Let $P_G(u, v)$ denote the set of all (u,v)-paths in G. We define $d_w(u, v) = \min_{P\in P_G(u,v)} w(P)$; in other words, $d_w(u, v)$ is the weight of the shortest (weighted) path from u to v in G. The (weighted) *diameter* D_w of (G, w) is defined as $\max_{u,v\in V} d_w(u, v)$. For unweighted graphs G (i.e. $w(e) = 1$ for all $e \in E$), we omit w from our notations. In particular, $d(u, v)$ is the (hop-)distance between u and v in G, and D is the diameter of the unweighted network G.

▶ **Definition 3** (Single Source Shortest Paths and All-Pairs Shortest Paths)**.** In the (weighted) single source shortest paths problem (SSSP), we are given a weighted network (G, w) and a source node s. We want each node v to know the distance $d_w(s, v)$ between itself and s. In the (weighted) all pairs shortest paths problem (APSP), each node $v \in V$ needs to know $d_w(u, v)$ for all $u \in V$.

For any α, we say an algorithm A is an *α-approximation* algorithm for SSSP if each node v obtains a value $\widetilde{d}_w(s, v)$ from A, such that $d_w(s, v) \leq \widetilde{d}_w(s, v) \leq \alpha \cdot d_w(s, v)$. Similarly, we say A is an α-approximation algorithm for APSP if each node v obtains values $\widetilde{d}(u, v)$ such that $d_w(u, v) \leq \widetilde{d}_w(u, v) \leq \alpha d_w(u, v)$ for all u. Note that this is one-sided error; an algorithm A is not an α-approximation algorithm if it ever outputs a value $\widetilde{d}_w(s, v) < d_w(s, v)$.

[3] This is mainly of interest for our lower bounds. Our algorithms also work with private randomness.

4 Lower Bounds for Weighted and Unweighted Diameter Computation and Approximation

Frischknecht et al. proved in [10] that any algorithm that computes the exact diameter of an unweighted graph requires at least $\Omega(\frac{n}{B})$ rounds of communication in the unicast CONGEST model. Note that they consider an arbitrary message-size B; the CONGEST model typically considers only $B = \mathcal{O}(\log n)$. We consider arbitrary B here as well. Their lower bound is achieved by constructing a reduction from the two-party communication problem of *set disjointness* to the problem of calculating the diameter of a particular unweighted graph G. We extend their construction that considers exact computation of the diameter of an unweighted graphs to the case of $(2 - 1/poly(n))$-approximating the diameter in a (positively) weighted graph. This is done by assigning weights to the edges in their (unweighted) construction in a convenient way and deriving the approximation-factor. We start by reviewing basic tools from two-party communication complexity and then present the modification of the construction of [10] for the CONGEST model in Section 4.3. Subsequently we transfer this bound to the BCC model.

4.1 A Review of Basic Two-Party Communication Complexity

It is necessary to review the basics of two-party communication complexity in order to present our results in a self-contained way. In the remaining part of this subsection we restate the presentation given in [15] only for completeness and convenience of the reader.

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 4** (Communication complexity)**.** Let $\mathcal{A}_\delta$ be the set of two-party algorithms that use public randomness (denoted by pub), which when used by Alice and Bob, compute h on any input a (to Alice) and b (to Bob) with an error probability smaller than δ. Let $A \in \mathcal{A}_\delta$ be an algorithm that computes h. Denote by $R_\delta^{cc-pub}(A(a,b))$ 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

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

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 $\{0,\dots,k-1\}$ and need to decide whether they are disjoint. Here, the strings a and b indicate membership of elements to each of these sets.

▶ **Definition 5** (Disjointness problem)**.** The set disjointness function $\mathrm{disj_k} : \{0,1\}^k \times \{0,1\}^k \to \{0,1\}$ is defined as follows.

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

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 use the following basic theorem that was proven in Example 3.22 in [20] and in [2, 3, 18, 31].

▶ **Theorem 6.** *For any sufficiently small $\delta > 0$ we can bound $R_\delta^{cc-pub}(\mathrm{disj_k})$ by $\Omega(k)$.*

4.2 Lower Bounds for Weighted Diameter Computation in the Unicast CONGEST Model

▶ **Theorem 7.** *For any $n \geq 10$ and $B \geq 1$ and sufficiently small ε any distributed randomized ε-error algorithm A that computes a $(2 - 1/poly(n))$-approximation of the diameter of a positively weighted graph requires at least $\Omega(\frac{n}{B})$ time for some n-node graph.*

We follow the strategy of [10] and reduce the function $disj_{k(n)^2}$ to finding the diameter of a graph G. Note that the graph in [10] is unweighted, while ours is weighted. We set a parameter $k(n)$ to be $k(n) = \lfloor \frac{n}{10} \rfloor$ and construct a graph $G_{a,b}$. We do so by defining a graph $G_a = (V_a, E_a)$ that depends on inputs a and a graph $G_b = (V_b, E_b)$ that depends on b. Based on these graphs G_a and G_b, we derive the graph $G_{a,b}$ containing both G_a and G_b. We start by constructing sets of nodes $L = \{l_v | v \in \{1, \ldots, 2k(n) - 1\}\}$ and $R = \{r_v | v \in \{1, \ldots, 2k(n) - 1\}\}$. Let $L_1 = \{l_v | v \in \{1, \ldots, k(n) - 1\}\}$ and $L_2 = \{l_v | v \in \{k(n), \ldots, 2k(n) - 1\}\}$, and define $R_1 = \{r_v | v \in \{1, \ldots, k(n) - 1\}\}$ and $R_2 = \{r_v | v \in \{k(n), \ldots, 2k(n) - 1\}\}$. We add a node c_L to V_a and a node c_R to V_b, then add edges from c_L to all nodes in L and from c_R to all nodes in R. We also add edges between each pair of nodes in L_1, R_1, L_2, and R_2, and from l_i to r_i for $i \in \{1, \ldots, 2k(n) - 1\}$. Finally, we add an edge from c_L to c_R. Note that these sets of (right/left) nodes only depend on the lengths of the inputs. In the proof we define edges E_a that connect nodes in V_a depending on a. We also define edges E_b that connect nodes in V_b depending on b.

As in [10], we can represent the $k(n)^2 - 1$ bits of input a by the $k(n)^2$ possible edges between the $k(n)$ nodes L_1 and $k(n)$ nodes L_2. More specifically, we choose the mapping from integers in $\{1, \ldots, k(n)^2 - 1\}$ to pairs of integers in $\{1, \ldots, k(n) - 1\} \times \{k(n), \ldots, 2k(n) - 1\}$, such that i is mapped to $(l_{u_i}, l_{v_i}) = \left(i \mod k(n), k(n) + \left\lfloor \frac{i}{k(n)} \right\rfloor\right)$. We add edge (l_{u_i}, l_{v_i}) to G_a if and only if $a(i) = 0$, and likewise represent the bits of b by adding edge (r_{u_i}, r_{v_i}) to G_b if and only if $b(i) = 0$.

We call the graph defined by these edges $G_a = (V_a, E_a)$, and construct a similar graph G_b for input b. We define the cut-set $C_{k(n)^2} = \{(l_v, r_v) : v \in \{0, \ldots, 2k(n) - 1\}\}$ to be the $2k(n)$ edges connecting each l_v to the corresponding r_v. We will refer to the sets of vertices $L_1 \cup R_1 = \{l_v | v \in \{1, \ldots, k(n) - 1\}\} \cup \{r_v | v \in \{1, \ldots, k(n) - 1\}\}$ as **UP** (upper part of the graph) and $L_2 \cup R_2 = \{l_v | v \in \{k(n), \ldots, 2k(n) - 1\}\} \cup \{r_v | v \in \{k(n), \ldots, 2k(n) - 1\}\}$ as **LP** (lower part of the graph).

Figure 1 visualize this and we note that the former set is in the upper portion of the graph, and the latter is in the lower portion. Finally, we set $G_{a,b} = G_a \cup G_b \cup C_k$.

Now we assign weights to the edges in this construction. We set the weight of every edge in G_a and in G_b to be 1, and the weight of each edge in $C_{k(n)^2}$ to be $1/p$, the smallest possible weight (see definition of the weights in Section 3).

▶ **Lemma 8.** *The weighted diameter of $G_{a,b}$ is at most $2 + 1/p$.*

Proof. We show case by case that for any nodes u and v in $G_{a,b}$ the distance $d_w(u, v)$ is at most $2 + 1/p$. The cases are as follows:

1. **Nodes u and v are both in G_a:** Every node in G_a other than C_L is connected to C_L by an edge of length 1, and thus each node in G_a can reach any other node in G_a using at most two edges of length 1. Thus, $d_w(u, v) \leq d_w(u, c_L) + d_w(c_L, v) \leq 2$.

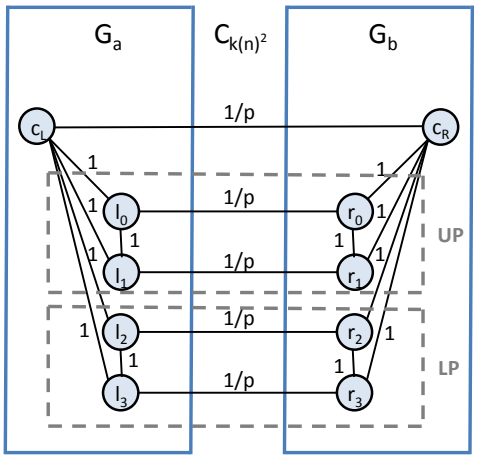


Figure 1 Base graph of (weighted) diameter $2 + 1/p$.

2. **Nodes u and v are both in G_b:** This case is identical to the previous case, so $d_w(u, v) \leq 2$.
3. **Node u is in G_a and node v is in G_b (or vice verse):** From u it is at most one hop to C_L of length 1, and from v it is at most one hop to C_R of length 1. Since the edge between c_L and c_R has weight $1/p$, we conclude that $d_w(u, v) \leq d_w(u, c_L) + d_w(c_L, c_R) + d_w(c_R, v) = 2 + 1/p$.

◀

Following the ideas of [10], we reduce the problem of deciding disjointness between sets a and b to computing the diameter of a graph.

▶ **Lemma 9.** *The diameter of $G_{a,b}$ is $1 + 2/p$ if the sets a and b are disjoint, else it is $2 + 1/p$.*

Proof. **If inputs a and b are not disjoint,** then there exists an $i \in \{1, \ldots, k(n)^2\}$ such that $a(i) = b(i) = 1$. Let us fix such an i for now and 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 of at least $2 + 1/p$. The path must contain an edge of length $1/p$ from the cut-set $C_{k(n)^2}$, since these are the only edges that connect G_a to G_b. To obtain a path of length $1 + 1/p$ we are only allowed to add one more edge from either G_a or G_b. When looking at the construction, the only two paths of length $1 + 1/p$ that we could hope for are (l_ν, l_μ, r_μ) and (l_ν, r_ν, r_μ). However, due to $a(i) = b(i) = 1$ and the implied choice of ν and μ, we know that the construction of $G_{a,b}$ does not include edge (l_ν, l_μ) nor edge (r_ν, r_μ). Thus none of these paths exists and we conclude that $d_w(l_\nu, r_\mu) \geq 2 + 1/p$. **Conversely if a and b are disjoint,** the diameter of $G_{a,b}$ is at most $1 + 2/p$. We prove this by showing that for any nodes u and v in $G_{a,b}$ the distance $d_w(u, v)$ is at most $1 + 2/p$. To do this we distinguish three cases:

1. **Node u is in G_a and node v is in G_b (or vice versa):** When considering the nodes c_L and c_R, we notice that from each of these nodes every other node in the graph can be reached within 2 hops, one of which has weight $1/p$. Now we can assume without loss of generality that $u = l_\nu \in G_a$ and $v = r_\mu \in G_b$ for some $\mu, \nu \in \{1, \ldots, 2k(n) - 1\}$. Since we assumed that a and b are disjoint there must be either at least one of the edges (l_ν, l_μ) or (r_ν, r_μ) in case that one of the nodes is in **UP** and the other node is in **LP**. Thus there is at least one of the paths (l_ν, l_μ, r_μ) or (l_ν, r_ν, r_μ) with $d_w(l_\nu, r_\mu) \leq 1 + 1/p$. In the remaining case u, v are both in **UP** or both in **LP**, and we make use of the clique-edges (among

nodes in the G_a (or G_b) part of **UP** (or **LP**), there are 4 cliques in total) and conclude that u and v are connected by path (l_ν, r_ν, r_μ) of length $d_w(l_\nu, r_\nu) + d_w(r_\nu, r_\mu) = 1 + 1/p$.

2. **Nodes u and v are both in G_a:** Let $u = a_i$ and $v = a_j$. If an edge between u and v does not directly exist, then we know an edge between b_i and b_j must exist. Thus we can get from u to v by using the edges (u, b_i), (b_i, b_j) and (b_j, v), for a path of total length $1 + 2/p$.
3. **Nodes u and v are both in G_b:** we use identical logic to case 2, where both u and v are in G_a; the distance between these nodes is at most $1 + 2/p$.

Finally note, that these two cases combined with the upper bound from Lemma 8 imply that $d_w(l_\nu, r_\mu) = 2 + 1/p$ if and only if a and b are not disjoint. ◀

▶ **Lemma 10.** *Computing a $(2 - o(1))$-approximation of the diameter in positively weighted graphs requires the exchange of $\Omega(n^2)$ bits of information.*

Proof. This follows immediately from Theorem 6; computing $disj_{k(n)^2}$ requires the exchange of $\Omega(k(n)^2) = \Omega(n^2)$ bits of information through the edges in $C_{k(n)}$ in order to decide if a and b are disjoint. ◀

Proof of Theorem 7. We use the graph $G_{a,b}$ constructed above to show that any algorithm A that computes a $(2 - 1/p')$-approximation of the diameter requires $\Theta(\frac{n}{B})$ time, for a p' that we define later in terms of p.

First note that in case the diameter is $(1 + 2/p)$ any A must output a value of at most $(1 + 2/p)(2 - 3/p) = 2 + 1/p - 6/p^2$. As this value is strictly smaller than the other possible diameter of $G_{a,b}$, which is $(2 + 1/p)$, any $(2 - 3/p)$-approximation algorithm can decide whether the $D_w(G_{a,b})$ is $(1 + 1/p)$ or $(2 + 1/p)$. We set $p = 3 \cdot p'$ to get our desired $(2 - 1/p')$-approximation algorithm. Based on this one can decide if inputs a and b, that were used to construct the graph $G_{a,b}$, are disjoint.

We know due to Theorem 6 that any algorithm must exchange $\Omega(k(n)^2)$ bits of information through the edges in $C_{k(n)}$ in order to decide if a and b are disjoint. As the bandwidth of $C_{k(n)}$ is $\mathcal{O}(|C_{k(n)}| \cdot B) = \mathcal{O}(k(n) \cdot B)$, we conclude that $\Omega(k(n)/B)$ rounds are necessary to do so. Due to the choice of $k(n)$ we conclude that $\Omega(\frac{n}{B})$ rounds are necessary to $(2 - 1/p)$-approximate the diameter of a graph. ◀

4.3 Lower Bounds for Weighted Diameter Computation in the BCC Model

Given a two-party communication problem f' with inputs a, b, we define the *f'-derived graph* $G_{a,b}$ to be a graph constructed from f' as described previously, with G_a encoding the input a to one party and G_b encoding the input b to two parties.

▶ **Theorem 11.** *Given a two-party communication problem f' with inputs a, b, if $R_\epsilon^{cc-pub}(f')$ is a lower bound on the number of bits that must be communicated in f', then solving the problem on the f'-derived graph $G_{a,b}$ with a randomized algorithm A must take at least $\frac{R_\epsilon^{cc-pub}(f')}{nB}$ rounds in the* BCC *model.*

Before presenting the proof, we want to stress that the output our algorithm A depends only on the edges of $G_{a,b}$. Other edges of the clique not mentioned in the construction of $G_{a,b}$ are still present in the CONGEST-CLIQUE model (not in the CONGEST model studied in Theorem 7) but can be used only for communication. These (additional) communication edges are not part of the lower bound construction and do not affect the diameter of the graph $G_{a,b}$.

Proof. In each round, any algorithm can send at most $|G_a| \cdot B$ bits of information from G_a to G_b, as each vertex in G_a must broadcast the same B bits to all other vertices in G_b in the BCC model. Similarly, any algorithm can send at most $|G_b| \cdot B$ bits from G_b to G_a. There are no further nodes outside of $G_{a,b}$ that could increase the bandwidth. Thus, any algorithm can exchange at most $(|G_a| + |G_b|) \cdot B = nB$ bits between G_a and G_b in each round. Therefore $\frac{R_\epsilon^{cc-pub}(f')}{nB}$ is a lower bound on the number of rounds that algorithm A must take. ◀

▶ **Theorem 12.** *Computing a $(2-o(1))$-approximation of the diameter in positively weighted graphs in the* BCC *model takes $\Omega(n/B)$ rounds.*

Proof. Computing a $(2-o(1))$-approximation of the diameter in positively weighted graphs is shown to require the exchange of $\Omega(k(n)^2)$ bits of information at the end of the proof of Theorem 10 above. Due to the choice of $k(n)$, these are $\Omega(n^2)$ bits. The statement then follows directly from an application of Theorem 11. ◀

▶ **Theorem 13.** *Computing the diameter exactly in unweighted graphs takes $\Omega(n/B)$ in the* BCC *model.*

Proof. Computing the exact diameter of unweighted version of the graph $G_{a,b}$ is shown to require $\Omega(n^2/B)$ bits of information to be exchanged in [10]. Thus, the result follows by Theorem 11 using similar arguments as in the proof of Theorem 12. ◀

▶ Remark 14. Note that a 2-approximation of the diameter of positively weighted graphs is achievable by computing SSSP starting in an arbitrary node, and returning twice the length of the largest distance computed. To compute (exact) SSSP we can use the SSSP-algorithm presented in [25], that runs in $\tilde{\mathcal{O}}(\sqrt{n})$ time.

5 Deterministic Hitting Set Computation in the BCC Model

▶ **Definition 15.** Given a node $u \in V$, the set $S^k(u)$ of a node $u \in G$ contains the k nodes closest to u in a weighted graph G, with ties broken by node ID. In other words, $S^k(u) \subset V$ has the following properties:

1. $|S^k(u)| = k$, and
2. for all $s \in S^k(u)$ and $t \notin S^k(u)$, either (i) $d_w(u,s) < d_w(u,t)$, or (ii) $d_w(u,s) = d_w(u,t)$ and the ID of s is smaller than the ID of t.

▶ **Definition 16.** A k-hitting set S of a graph $G = (V, E)$ is a set of nodes such that, for every node $v \in V$, there is at least one node of S in $S^k(v)$.

Our algorithm (see Appendix A.3 of the full version [16] for pseudocode) takes as input a graph G and an integer k, and returns a k-hitting set $S \subseteq V$. The algorithm works as follows: each node starts by broadcasting its k incident edges of smallest weight to all other nodes. If the node does have less than k neighbors, it just broadcasts the weight of all its incident edges. This enables every node u to locally compute a set $S^k(u)$, consisting of the k closest nodes to u in G ([25], Observation 3.12). By closest we refer to the weighted distance of nodes to u; note that $S^k(u)$ always has k nodes for any $k \leq n$, as the graph is connected. We initialize $S := \emptyset$; S is updated over time until it is our desired k-hitting set. Let at any time R be composed of the sets $S^k(v)$ such that $S^k(v) \cap S = \emptyset$ (initially R contains all $S^k(v)$). We repeatedly find the vertex v_{max} that is contained in the largest number of elements in R (breaking ties by minimum node ID). We then add this v_{max} to S and update

R accordingly. In Lemma 17 we show that this method of greedily constructing a hitting set achieves a $\mathcal{O}(\log n)$-approximation of the smallest possible hitting set.

▶ **Lemma 17.** *Given a graph G, if the smallest possible hitting set uses N vertices, then S contains at most $\mathcal{O}(N \log n)$ vertices.*

Proof. This proof can be found in Appendix A.3 of the full version of this paper [16]. ◀

▶ **Lemma 18.** *Procedure* HittingSet *described in Appendix A.3 of the full version of this paper* [16] *computes a k-hitting set of size $\tilde{\mathcal{O}}(n/k)$ in $\mathcal{O}(k)$ rounds.*[4]

Proof. See Appendix A.3 of [16]. ◀

6 Deterministic $(2 + o(1))$-Approximation of APSP in Time $\tilde{\mathcal{O}}(n^{1/2})$ in the BCC Model

Nanongkai provides a randomized distributed algorithm ([25], Algorithm 5.2) to $(2 + o(1))$-approximate APSP in the BCC model that runs in $\tilde{\mathcal{O}}(n^{1/2})$ time. At a high level, this algorithm works by

1. choosing a random *$\sqrt{n}$-hitting set* $R \subseteq V$ of size $\tilde{\mathcal{O}}(\sqrt{n})$ such that for all nodes in V, there is some node in R within $\sqrt{n}$ hops,
2. $(1 + o(1))$-approximate (using random delays to avoid congestion) shortest paths from each node in the hitting set R to every node in V,
3. using these shortest paths to approximate shortest paths between all pairs of nodes.

We have previously presented a method to deterministically compute a $\sqrt{n}$-hitting set $R \subseteq V$ in Section 5. Nanongkai uses a randomized procedure to compute shortest paths from this hitting set to all other nodes, which we will replace by a deterministic one in this paper. This results in a deterministic $\tilde{\mathcal{O}}(n^{1/2})$ round algorithm:

▶ **Theorem 19.** *The deterministic Algorithm 2 (stated fully in* [16]*) returns a $(2 + o(1))$-approximation of APSP in time $\tilde{\mathcal{O}}(n^{1/2})$.*

The remainder of this section is devoted to explaining and analyzing this algorithm in order to prove this theorem (see pseudocode in Appendix A.2 of the full version [16]). While doing so, we also review the majority of Algorithm 5.2 of [26]. We do this to be able to point out our modifications exactly and to argue that each step can indeed be done in the BCC model, while the original implementation of Algorithm 5.2 of [26] is just stated for the CONGEST-CLIQUE model (without distinguishing between BCC and UCC models). As shown in Theorem 5.3 of [26], Algorithm 5.2 of [26] computes a $(2 + o(1))$-approximation of APSP on weighted graphs. Note that we only change the implementation of the algorithm to be deterministic, meaning we can immediately derive the same approximation ratio (with probability 1 instead of w.h.p.).

Given a graph G, we start by computing a *k-shortcut graph* G^k of G for $k = \sqrt{n}$, defined below.

▶ **Definition 20** (k-shortcut graph)**.** The shortcut graph $G^k = (V, E^k)$ is obtained by adding an edge (u, v) of weight $d_w(u, v)$ to E^k for every $u \in V$ and $v \in S^k(u)$.

[4] By using the $\mathcal{O}$-notation we implicitly assume that $k \leq n^{1-\text{polylog}n}$, which will always be the case in this paper.

To construct this graph, each node begins by broadcasting the k lightest edges adjacent to it. If there are less than k edges adjacent to a node, that node just broadcasts all of them and their weights. Based on this information each node $u \in V$ can compute $S^k(u)$, since running e.g. k rounds of Dijkstra's algorithm will only need the k-lightest edges incident to each node (as argued in [26]). During the next $O(k)$ time steps, each node u simultaneously broadcasts its $S^k(u)$ and creates a simulated shortcut edge from every node $u \in G$ to every node $v \in S^k(u)$. New edge weights $w'(u,v) := \min\{w(u,v), \min_{z \in S^k(u)} d_w(u,z) + d_w(z,v)\}$ are assigned to this graph. Then, node u locally computes a k-hitting set R of G, as described in Section 5.

To further describe our algorithm we need the following definitions.

▶ **Definition 21** (h-hop SSSP ([25], Definition 3.1))**.** Consider a network (G, w) and a given integer h. For any nodes u and v, let $P^h(u,v)$ be the set of all (u,v)-paths containing at most h edges. Define the h-hop distance between u and v as

$$d_w^h(u,v) = \begin{cases} min_{P \in P^h(u,v)} w(P) & : \ P^h(u,v) \neq \emptyset \\ \infty & : \ otherwise. \end{cases}$$

Let h-hop SSSP be the problem where, for a given weighted network (G, w), source node s (node s knows that it is the source), and integer h (known to every node), we want every node u to know $dist_{G,w}^h(s,u)$.

▶ **Definition 22** (MSSP, h-hop MSSP [26])**.** Given a set $S \subseteq V$, the multi-source shortest paths problem (MSSP) is to compute the SSSP tree from each node in S. This problem is also referred to as the S-shortest paths problem (S-SP). In the h-hop MSSP problem (a.k.a. h-hop S-SP [7, 14]) one is interested in the h-hop versions of SSSP w.r.t source nodes S.

Nanongkai states an MSSP algorithm that works in the CONGEST model, and computes $(1 + o(1))$-approximate distances on weighted graphs. The main idea of this algorithm is based on the following theorem.

▶ **Theorem 23** ([25], Theorem 3.3)**.** *Consider any n-node weighted graph (G, w) and integer h. Let $\epsilon = 1/\log n$, and let W be the maximum-weight edge in G. For any i and edge (x,y), let $D'_i = 2^i$ and $w'_i(x,y) = \left\lceil \frac{2hw(x,y)}{\epsilon D'_i} \right\rceil$. For any nodes u and v, if we let*

$$\widetilde{d}_w^h(u,v) = min \left\{ \frac{\epsilon D'_i}{2h} \times d_{w'_i}(u,v) \mid i : d_{w'_i}(u,v) \leq (1 + 2/\epsilon)h \right\},$$

then $d_w^h(u,v) \leq \widetilde{d}_w^h(u,v) \leq (1+\epsilon) \cdot d_w^h(u,v)$.

This theorem states that we can compute an $(1+\varepsilon)$-approximation of h-hop-bounded SSSP when we run $\mathcal{O}(\log n)$ many h-hop-bounded SSSP computations rooted in node u, each with modified weights $w'_1, \ldots . w'_{\log n}$. To obtain an $(1+\varepsilon)$-approximation for h-hop-bounded MSSP for sources S, Nanongkai performs $\mathcal{O}(\log n)$ many h-hop-bounded MSSP computations rooted in S, each with modified weights $w'_1, \ldots . w'_{\log n}$. In each execution of a h-hop MSSP, Nanongkai starts all h-hop SSSP computations in all nodes of S simultaneously and delays each step of any h-hop SSSP algorithm by a random amount. This is shown to guarantee that with high probability the $|S|$ copies of h-hop SSSP do not conflict with each other.

We can adapt Nanongkai's h-hop MSSP algorithm to a deterministic setting using the source detection algorithm of [21].

▶ **Definition 24** ((S, H, K)-source detection [21])**.** Given an unweighted graph G and $H, K \in \mathbb{N}_0$, the (S, H, K)-source detection problem is to output for each node $u \in V$ the set $L_u(H, K)$ of all (up to) K closest sources in S to u, which are at most H hops away.

▶ **Lemma 25** (Theorem 4.4, [21])**.** *The (S, H, K)-source detection problem can be solved in the* CONGEST *model in* $\min(H, D) + \min(K, |S|)$ *rounds.*

In Algorithm 1 of [21] (which corresponds to Lemma 25), each node always broadcasts the same message within each time step to all neighbors. Therefore it runs in the broadcast version of the CONGEST model. The algorithm is stated for unweighted graphs; we adapt it to weighted graphs by replacing every edge e of weight $w(e)$ by a path of $w(e)$ edges, each of weight 1. The simulation of these new nodes and edges is handled by the two nodes adjacent to e, and is equivalent to delaying any transmission through e by $w(e)$ rounds as it is done in [25]. This transforms a weighted graph into an unweighted one.

We now use the above deterministic procedure instead of Nanongkai's randomized one to approximate weighted h-hop MSSP on the hitting set by choosing $S := R$. In each execution of the unweighted h-hop MSSP on R, during iteration i, we set the weight $w'_i(x, y)$ to be $\left\lceil \frac{2hw'(x,y)}{\epsilon 2^i} \right\rceil$, then we execute Lenzen and Peleg's (S, H, K)-source detection algorithm (Lemma 25) on graph G^k using weight w'_i with $R := S$ and $H := h$. Furthermore, we set $K := |R|$ to guarantee that all sources within h hops are detected. We use the fact that, in our model, nodes at any distance in the graph G can directly communicate with each other, so $|D| = 1$ and $min(H, D) = h$.

After all $\mathcal{O}(\log n)$ executions of the source-detection algorithm have completed, each node $u \in V$ knows its distance to every node in R under every set of weights w_i. By Theorem 23, this allows us to compute a $(1 + o(1))$-approximation of $d^h_w(s, u)$ on G^k when choosing $\varepsilon = 1/\log n$, which according to [26] is equal to $d_w(s, u)$ for each $s \in R$ and $u \in V$. This is proven in [26] via the choice of h and k, which we do not change. Finally we broadcast these weights and compute like in [26] the value $d''(u, v)$, which is shown in [26] to be a $(2 + o(1))$-approximation.

Proof of Theorem 19. *Runtime*: Broadcasting the k lowest-weight edges, one by one in each round, takes k rounds in the BCC model. Computing $S^k(u)$ and w' takes no additional communication. By Lemma 17 we can compute the k-hitting set R is computed in time $\mathcal{O}(k)$ in the BCC model. Computing weights w'_i in takes $\mathcal{O}(\log W)$ rounds. Each execution of $(R, h, |R|)$-source detection takes $h + |R|$ time steps on the (simulated) undirected graph, and there are $\mathcal{O}(\log W)$ iterations, see Lemma 25. Since $h = \mathcal{O}(n^{1/2})$ and $|R| = \tilde{\mathcal{O}}(n/k) = \tilde{\mathcal{O}}(\sqrt{n})$ (see Lemma 17) and $\log W = \mathcal{O}(\log n)$, as $W \in$ poly n. The remaining parts of the algorithm only perform broadcasts, which take $|R| = \tilde{\mathcal{O}}(\sqrt{n})$ rounds. Therefore the total runtime is $\tilde{\mathcal{O}}(\sqrt{n})$.

Approximation ratio: The $(2 + o(1))$-approximation ratio for our algorithm is immediately derived from [25], as we do not change Nanongkai's algorithm besides modifying it to execute deterministically. ◀

7 Open Problems

It is natural to ask whether our method of proving lower bounds for the diameter in the BCC model can be extended to other problems. Of particular interest are those discussed in [10], since these problems use similar graph constructions for proving lower bounds. It would

also be of interest to further reduce the runtime of approximating APSP in the BCC and UCC model, maybe also at the cost of larger approximation factors.

References

1. N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. J. Comput. and Syst. Sciences, 58(1):137-147, 1999. *Journal of Computer and System Sciences*, 58(1):137–147, 1999.
2. László Babai, Peter Frankl, and Janos Simon. Complexity classes in communication complexity theory (preliminary version). In *Proceedings of the 27th annual IEEE Symposium on Foundations of Computer Science, FOCS 1986, Toronto, Ontario, Canada, 27-29 October 1986*, pages 337–347, 1986.
3. Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, and D. Sivakumar. An information statistics approach to data stream and communication complexity. *Journal of Computer and System Science*, 68(4):702–732, 2004. doi:10.1016/j.jcss.2003.11.006.
4. Keren Censor-Hillel, Petteri Kaski, Janne Korhonen, Christoph Lenzen, Ami Paz, and Jukka Suomela. Algebraic methods in the congested clique. *arXiv preprint 1503.04963*, 2015.
5. Atish Das Sarma, Stephan Holzer, Liah Kor, Amos Korman, Danupon Nanongkai, Gopal Pandurangan, David Peleg, and Roger Wattenhofer. Distributed verification and hardness of distributed approximation. *SIAM Journal on Computing*, 41(5):1235–1265, 2012.
6. Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM (CACM)*, 51(1):107–113, 2008.
7. Benjamin Dissler. Efficient multi-aggregation with applications to centrality computation. Semester thesis, ETH Zürich, Department of Information Technology and Electrical Engineering, Zürich, Switzerland, 2013.
8. Shahar Dobzinski, Noam Nisan, and Sigal Oren. Economic efficiency requires interaction. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 – June 03, 2014*, pages 233–242, 2014.
9. Andrew Drucker, Fabian Kuhn, and Rotem Oshman. On the power of the congested clique model. In *Proc. of the 33rd annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing, PODC 2014, Paris, France, July 15-18, 2014*, pages 367–376, 2014.
10. S. Frischknecht, S. Holzer, and R. Wattenhofer. Networks cannot compute their diameter in sublinear time. In Yuval Rabani, editor, *Proc. of the 23rd Annual ACM-SIAM Symp. on Discrete Algorithms, SODA 2012*, pages 1150–1162, 2012.
11. O. Goldreich and A. Warning. Secure multi-party computation, 1998. Unpublished manuscript.
12. James W. Hegeman and Sriram V. Pemmaraju. Lessons from the congested clique applied to mapreduce. In Magnús M. Halldórsson, editor, *Structural Information and Communication Complexity – 21st International Colloquium, SIROCCO 2014, Takayama, Japan, July 23-25, 2014. Proceedings*, volume 8576 of *Lecture Notes in Computer Science*, pages 149–164. Springer, 2014.
13. James W. Hegeman, Sriram V. Pemmaraju, and Vivek Sardeshmukh. Near-constant-time distributed algorithms on a congested clique. In *DISC*, pages 514–530, 2014. doi:10.1007/978-3-662-45174-8_35.
14. S. Holzer and R. Wattenhofer. Optimal distributed all pairs shortest paths and applications. In Darek Kowalski and Alessandro Panconesi, editors, *Proceedings of the 31st annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing, PODC 2012, Funchal, Madeira, Portugal, July 16-18, 2012*, pages 355–364, 2012.
15. Stephan Holzer. *Distance Computation, Information Dissemination, and Wireless Capacity in Networks, Diss. ETH No. 21444*. Phd thesis, ETH Zurich, Zurich, Switzerland, 2013.

16 Stephan Holzer and Nathan Pinsker. Approximation of distances and shortest paths in the broadcast congest clique. *CoRR*, abs/1412.3445, 2014. URL: `http://arxiv.org/abs/1412.3445`.

17 Taisuke Izumi and Roger Wattenhofer. Time lower bounds for distributed distance oracles. In *Principles of Distributed Systems*, pages 60–75. Springer, 2014.

18 Bala Kalyanasundaram and Georg Schnitger. The Probabilistic Communication Complexity of Set Intersection. *SIAM Journal of Discrete Mathematics*, 5(4):545–557, 1992. `doi:10.1137/0405044`.

19 Hartmut Klauck, Danupon Nanongkai, Gopal Pandurangan, and Peter Robinson. The distributed complexity of large-scale graph processing. *arXiv preprint arXiv:1311.6209 (to appear at SODA'15)*, 2013.

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

21 C. Lenzen and D. Peleg. Efficient distributed source detection with limited bandwidth. In Panagiota Fatourou and Gadi Taubenfeld, editors, *Proceedings of the 32nd annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing, PODC 2013, Montreal, Quebec, Canada, July 22-24, 2013*, pages 375–382, 2013.

22 Christoph Lenzen. Optimal deterministic routing and sorting on the congested clique. In Panagiota Fatourou and Gadi Taubenfeld, editors, *Proceedings of the 32nd annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing, PODC 2013, Montreal, Quebec, Canada, July 22-24, 2013*, pages 42–50, 2013.

23 Zvi Lotker, Elan Pavlov, Boaz Patt-Shamir, and David Peleg. MST construction in O(log log n) communication rounds. In *Proceedings of the 15th annual ACM Symposium on Parallel Algorithms and Architectures, SPAA 2003, San Diego, California, USA, June 7-9,2003*, pages 94–100, 2003.

24 Grzegorz Malewicz, Matthew H Austern, Aart JC Bik, James C Dehnert, Ilan Horn, Naty Leiser, and Grzegorz Czajkowski. Pregel: a system for large-scale graph processing. In Ahmed K. Elmagarmid and Divyakant Agrawal, editors, *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2010, Indianapolis, Indiana, USA, June 6-10, 2010*, pages 135–146. ACM, 2010.

25 Danupon Nanongkai. Distributed approximation algorithms for weighted shortest paths. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, STOC'14, pages 565–573, 2014.

26 Danupon Nanongkai. Distributed approximation algorithms for weighted shortest paths. *CoRR*, abs/1403.5171, 2014. URL: `http://arxiv.org/abs/1403.5171`.

27 Edmund B Nightingale, Jeremy Elson, Jinliang Fan, Owen S Hofmann, Jon Howell, and Yutaka Suzue. Flat datacenter storage. In *OSDI*, pages 1–15, 2012.

28 Boaz Patt-Shamir and Marat Teplitsky. The round complexity of distributed sorting: Extended abstract. In *Proceedings of the 30th Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, PODC'11, pages 249–256, New York, NY, USA, 2011. ACM.

29 David Peleg. *Distributed computing: a locality-sensitive approach*. Society for Industrial and Applied Mathematics, Philadelphia, Pennsylvania, USA, 2000.

30 Sriram V. Pemmaraju and Vivek B. Sardeshmukh. Algebrisation in distributed graph algorithms: Fast matrix multiplication in the congested clique. *arXiv preprint 1412.2109*, 2014.

31 Alexander A. Razborov. On the Distributional Complexity of Disjointness. *Theoretical Computer Science*, 106(2):385–390, 1992.

32 Roger Wattenhofer. Principles of Distributed Computing, Lecture 11, ETH Zurich, Zurich, Switzerland, http://dcg.ethz.ch/lectures/podc_allstars/lecture/chapter11.pdf, 2011.