A Faster Distributed Single-Source Shortest Paths Algorithm

Sebastian Forster
Department of Computer Sciences
University of Salzburg
Salzburg, Austria

Danupon Nanongkai
EECS
KTH Royal Institute of Technology
Stockholm, Sweden

Abstract—We devise new algorithms for the single-source shortest paths (SSSP) problem with non-negative edge weights in the CONGEST model of distributed computing. While close-to-optimal solutions, in terms of the number of rounds spent by the algorithm, have recently been developed for computing SSSP approximately, the fastest known exact algorithms are still far away from matching the lower bound of $\tilde{\Omega}(\sqrt{n} + D)$ rounds by Peleg and Rubinovich [SIAM Journal on Computing 2000], where n is the number of nodes in the network and D is its diameter. The state of the art is Elkin's randomized algorithm [STOC 2017] that performs $\tilde{O}(n^{2/3}D^{1/3} + n^{5/6})$ rounds. We significantly improve upon this upper bound with our two new randomized algorithms for polynomially bounded integer edge weights, the first performing $\tilde{O}(\sqrt{nD})$ rounds and the second performing $\tilde{O}(\sqrt{n}D^{1/4} + n^{3/5} + D)$ rounds. Our bounds also compare favorably to the independent result by Ghaffari and Li [STOC 2018]. As side results, we obtain a $(1+\epsilon)$-approximation $\tilde{O}((\sqrt{n}D^{1/4} + D)/\epsilon)$-round algorithm for *directed* SSSP and a new work/depth trade-off for exact SSSP on directed graphs in the PRAM model.

Index Terms—distributed algorithms; shortest paths;

I. Introduction

In this paper, we consider the fundamental problem of computing single-source shortest paths (SSSP) in the CONGEST model [1] of distributed computing. The CONGEST model is one of the major message-passing models studied in the literature and is characterized by synchronized communication in a network via non-faulty bounded-bandwidth links. Such a communication network is modeled by a graph in which the nodes correspond to the processors in the network and the edges correspond to the communication links between the processors.

In unweighted graphs, the SSSP problem can be solved in $O(D)$ communication rounds by performing breadth first search, where D is the diameter of the graph. In weighted graphs, where we assume that the edge weights do *not* represent the communication speed, a straightforward distributed variant of the Bellman-Ford algorithm [2], [3], [4] computes SSSP in $O(n)$ rounds, where n is the number of nodes in the network. Moreover, Peleg and Rubinovich [5] showed a

lower bound of $\tilde{\Omega}(\sqrt{n} + D)$ for this problem[1], where D is the *unweighted* diameter of the graph, when viewed as the underlying communication network. (D is often called *hop diameter*.) The past few years have witnessed many improved upper bounds when *approximate solutions* are allowed (e.g. [6], [7], [8], [9], [10], [11], [12]), culminating in the close-to-optimal [13], [5], [14], [15], [16] approximation scheme of Becker et al. [10]. For *exact* algorithms, the $O(n)$ bound from the Bellman-Ford algorithm was the state of the art for a long time until in a recent breakthrough, Elkin [17] proved an upper bound that is sublinear in n for undirected graphs with non-negative edge weights; he obtained a randomized algorithm that performs $O((n \log n)^{5/6})$ rounds for $D = O(\sqrt{n \log n})$, and $O(D^{1/3}(n \log n)^{2/3})$ rounds for larger D.

A. Our Results

Our main result is a significant improvement upon Elkin's upper bound for polynomially bounded integer edge weights[2] and $D = \tilde{o}(n)$. We devise two randomized (Las Vegas) algorithms, the first performing $\tilde{O}(\sqrt{nD})$ rounds and the second performing $\tilde{O}(\sqrt{n}D^{1/4} + n^{3/5} + D)$ rounds; both bounds hold with high probability and in expectation. Our first bound matches the $\tilde{\Omega}(\sqrt{n} + D)$ lower bound of Peleg and Rubinovich [5] up to polylogarithmic factors when D is polylogarithmic in n.

Our algorithms in fact work in a more restricted model called *Broadcast CONGEST*, where a node must send the same message to every neighbor in each round. The bound holds when edge weights are in $\{0, 1, \ldots, \text{poly}(n)\}$ as typically assumed in the literature. Note that our bounds hold even when edges are *directed*. The directed case refers to when edges have directions, but the communication is always bi-directional. A question about this case was raised in [7]. The previous bound by Elkin does not work for this case. Note that in general our bounds have to be multiplied by the number of bits needed to represent edge weights (which is $O(\log n)$ under a typical assumption), while Elkin's bound does not.

a) Independent Work by Ghaffari and Li [18]: Recently in STOC 2018, and independently from our results, Ghaffari

---

[1] In this paper, we use $\tilde{O}(\cdot)$-, $\tilde{\Omega}(\cdot)$-, $\tilde{\Theta}(\cdot)$-, and $\tilde{o}(\cdot)$-notation to suppress factors that are polylogarithmic in n.

[2] In general, our running times scale with the number of bits needed to represent edge weights.

and Li presented a distributed algorithm for exact SSSP with time complexity $\tilde{O}(n^{3/4}D^{1/4})$ and another algorithm with complexity $\tilde{O}(n^{3/4+o(1)} + \min\{n^{3/4}D^{1/6}, n^{6/7}\} + D)$. Our bounds compare favorably with theirs in the entire range of parameters. Like our bounds, their bounds hold when edges are directed. Their bounds also depend on the number of bits needed to represent edge weights.

b) Additional Results: Using techniques developed in this paper, we obtain two additional results, which might be interesting for putting our main result and our techniques into context. The first is a $(1 + \epsilon)$-approximation $\tilde{O}((\sqrt{n}D^{1/4} + D)/\epsilon)$-round algorithm for *directed* SSSP. This algorithm is Monte Carlo, meaning that its approximation guarantee holds with high probability, and works in the Broadcast CONGEST model. Previously such a bound was known only for the special case of *single-source reachability* [19], where we want to know whether there is a directed path from the source node to each node. None of previous approximation algorithms for SSSP (e.g. [6], [7], [8], [9], [10]) can handle this case.

Our second result is a new work/depth trade-off for exact SSSP on directed graphs in the PRAM model. We provide algorithm with $\tilde{O}(mt + mn/t + n^3/t^3)$ work and $\tilde{O}(t)$ depth (for any fixed $1 \leq t \leq n$). The parallel SSSP problem has received considerable attention in the literature [20], [21], [22], [23], [24], [25], [26], [27], [28], but we are aware of only two relevant results for exact SSSP in directed graphs. First, the algorithm of Spencer [20] has $\tilde{O}(n^3/t^2 + m)$ work and $\tilde{O}(t)$ depth (for any $1 \leq t \leq n$). Second, the algorithm of Klein and Subramanian has $\tilde{O}(m\sqrt{n})$ work and $\tilde{O}(\sqrt{n})$ depth, but does not give any trade-offs. Thus, our algorithm gives a better trade-off than the state of the art when $t = \tilde{o}(\sqrt{n})$ and in this case it has $\tilde{O}(mn/t + n^3/t^3)$ work and $\tilde{O}(t)$ depth. In this range, the trade-off is (up to polylogarithmic factors) the same as the one of Shi and Spencer [24] for undirected graphs.[3]

B. Further Related Work

Closely related are *All-Pairs Shortest Paths (APSP)* and *k-Source Shortest Paths* (*k*-SSP) problems. The current best upper bound for APSP is $\tilde{O}(n^{5/4})$ [29]. The situation is more complicated for *k*-SSP. We refer to [30], [31], [32], [33], [17], [29], [18], [34] and references therein for details.

The exact shortest paths problem has also received attention in the Congested Clique model, which is the special case of the CONGEST model where the network is a clique. Nanongkai obtained a SSSP algorithm that performs $\tilde{O}(\sqrt{n})$ rounds in this model and Censor-Hillel et al. [35] obtained an algorithm that performs $\tilde{O}(n^{1/3})$ rounds, solving the more general APSP problem.

There are many other graph problems studied in the CONGEST model, such as minimum spanning tree, minimum cut, and maximum flow; see, e.g., [36], [37], [38], [39], [40], [41], [42], [43], [44], [45], [46], [47].

---

[3]The algorithm of Shi and Spencer uses a hop set construction that is tailored to undirected graphs; the same hop set construction has later also been used in the distributed setting [7], [17].

C. Overview of Techniques

The starting point of our algorithms is the classic *scaling framework* which was heavily used in the sequential setting since the 80s (e.g. [48], [49], [50]). In the context of distributed shortest paths, this framework was used first in the algorithm of Huang et al. [29] for APSP. This algorithm follows Gabow's *bitwise* scaling technique [48], where edge weights are considered one bit at a time so that distances between nodes can be bounded from above. This approach was later taken by Ghaffari and Li [18] for computing SSSP. Our starting point is rather different from Huang et al. [29] and Ghaffari and Li [18] in that we will follow *recursive* scaling, Gabow's second scaling technique [48], and its extension by Klein and Subramanian [21], who applied it for SSSP in the PRAM model. At a high level, this approach uses a very general reduction to extend certain *approximate* SSSP algorithms to *exact* SSSP algorithms. Intuitively, this allows us to borrow some existing tools developed for approximation algorithms.

The challenge here lies in two special aspects of this reduction. First, it is an inherent necessity of the reduction that the approximation algorithm works on directed graphs and thus our exact SSSP algorithm also works on directed graphs – assuming that the underlying communication network is undirected. Because of this, it is not a surprise that we achieve a new approximation algorithm that can handle the directed case as an additional result.

Second, and more importantly, the reduction will not work for any arbitrary approximate SSSP algorithm, as it requires the distance estimates returned by the algorithm to satisfy an additional condition about 'dominating' the edge weights of the input graph. Klein and Subramanian can ensure this property by (1) computing a directed hop set to effectively reduce the approximate shortest path diameter of the graph, (2) adding the edges of the hop set to the input graph, and (3) obtaining the distance estimate from performing a "small" number iterations of the Bellman-Ford algorithm, exploiting the reduced approximate shortest path diameter. This approach cannot directly be transferred to the CONGEST model because the additional edges of the hop set can only be "simulated" by the nodes in the network, and the standard way of doing so will not yield a small enough guarantee on the number of rounds spent in the final Bellman-Ford step. We circumvent this problem by using a different algorithm design instead, where we (1) compute a skeleton graph consisting of a few "important" nodes, including the source node, and edges between the skeleton nodes with weights equal to approximate pairwise distances, (2) solve recursively on the skeleton graph, and (3) run a "small" number of iterations of the Bellman-Ford algorithm incorporating the distances from the source computed in step (2) to obtain distance estimates with the domination property. This may sound a bit circular at first because we again face the problem of having to compute exact SSSP on the skeleton graph. However, computation on the smaller skeleton graph is now simulated by performing global broadcasts in the whole network. We can thus benefit from the fact that we

can use slightly different algorithm design techniques, such as "adding" edges to the graph without immediately increasing the number of rounds, just like in the PRAM model.

To this end, we note that although our algorithm follows the same general framework as that by Klein and Subramanian [21], it is not simply a mere "translation" of their PRAM algorithm to the CONGEST model. It in fact leads to a new work/depth trade-off in the PRAM model that could not be achieved with the algorithm of Klein and Subramanian.

D. Organization

The rest of this paper is organized as follows: In Section II we introduce all definitions necessary for our main results and review existing tools from the literature. In Section III we schematically develop the central "auxiliary" algorithm that we need to obtain our results. We prove its correctness, but do not yet analyze its model-specific complexity. In Sections IV and V we obtain our two main results by providing two slightly different implementations of the auxiliary algorithm together with their complexity analyses. Finally, in Section VI we sketch how our techniques imply additional results for approximate SSSP in the Broadcast CONGEST model and exact SSSP in the PRAM model.

II. PRELIMINARIES

A. CONGEST Model and Problem Formulation

The *CONGEST model* [1] is a synchronous message-passing model with non-faulty bounded-bandwidth links. More formally, consider a communication network of processors modeled by an undirected graph $N = (V, L)$ where each node of V models a processor and each pair of nodes $\{u, v\} \in L \subseteq \binom{V}{2}$ indicates a bidirectional communication link between the processors corresponding to u and v, respectively. In the remainder of this paper, we identify nodes and processors. In the CONGEST model, the initial knowledge about N is distributed among the nodes: every node has a unique identifier of size $O(\log n)$ (where $n = |V|$) initially known only to itself and its neighbors, i.e., the nodes to which it has direct communication links. The nodes communicate in synchronous rounds: At the beginning of each round, every node may send to each of its neighbors a message of size $B = \Theta(\log n)$ and subsequently receive the messages sent by its neighbors. Before the next round begins, each node may perform internal computation based on all messages it has received so far and its local knowledge of the network. The complexity of an algorithm is usually measured in the number of rounds and the total number of messages sent until the algorithm terminates, whereas internal computation is considered free. Typically, the asymptotic complexity is expressed in terms of $n := |V|$ and the diameter D of the unweighted communication network N, i.e., the maximum distance between any pair of nodes in N. The restriction of the bandwidth B to $\Theta(\log n)$ distinguishes the CONGEST model from the *LOCAL model* [51]. The *Broadcast CONGEST model* is a specialization of the CONGEST model with the additional constraint that, whenever a node sends a message in some round, it has to send (*broadcast*) *the same* message to all of its neighbors. In general, we call a round in which, for every node, the message sent to the neighbors is the same, a *broadcast round*.

In the single-source shortest paths (SSSP) problem, we are given a weighted directed graph $G = (V, E, w)$ and a distinguished source node $s \in V$ and the task is to compute the distance $d_G(s, v)$ from s to v in G for every node $v \in V$. In the CONGEST model, we require that the nodes of $G = (V, E, w)$ are equal to the nodes of the underlying communication network $N = (V, L)$ and that for every directed edge $(u, v) \in E$ there is an undirected communication link $\{u, v\} \in L$. The input is distributed among the nodes of the network as follows: initially, every node knows (a) whether it is the source s or not and (b) its set of incoming and outgoing edges together with their weight. The SSSP problem is solved as soon as every node knows its distance from s in G.

In this paper, we consider the restriction of the SSSP problem to *non-negative integer edge weights*, where we let W denote the maximum edge weight. We work under the typical assumption that each edge weight can be encoded in a constant number of messages, i.e. $B = \Omega(\log W)$, which implies that W is polynomial in n. Nevertheless, we do work out the dependence on W in our algorithms for the interested reader. Observe, that in the CONGEST model an implicit shortest path tree rooted at s can be constructed in additional $O(\log n)$ rounds: first every node sends its distance from s to all of its neighbors and then every node v determines one incoming neighbor u such that $d_G(s, v) = d_G(s, u) + w(u, v)$ which will serve as the parent in the shortest path tree.

B. Notation and Terminology

Consider a weighted directed graph $G = (V, E, w)$ with source node $s \in V$, where V of size $n := |V|$ is the set of nodes, $E \subseteq V^2$ of size $m := |E|$ is the set of edges, and $w : E \to \{0, 1, \ldots, W\}$ assigns a non-negative integer weight to each edge. For every path π in G, denote the weight of π in G by $w(\pi) := \sum_{e \in \pi} w(e)$. For every pair of nodes $u, v \in V$ denote the distance from u to v in G, i.e., the weight of the shortest path from u to v in G, by $d_G(u, v)$.

In defining multiplicative distance approximation, one can consider both overestimation and underestimation of the true distance. For technical reasons, we will refer to both types of distance estimates in our algorithms and thus use the following convention: If $\alpha \geq 1$, then $\tilde{d}(u, v)$ is an α-approximation of $d_G(u, v)$ for a pair of nodes $u, v, \in V$ if

$$d_G(u, v) \leq \tilde{d}(u, v) \leq \alpha \cdot d_G(u, v)$$

and if $\alpha < 1$, then $\hat{d}(u, v)$ is an α-approximation of $d_G(u, v)$ if

$$\alpha \cdot d_G(u, v) \leq \hat{d}(u, v) \leq d_G(u, v) \,.$$

Since the edge weights, and thus all pairwise distances, are integer, we assume without loss of generality that the distance estimates $\tilde{d}(u, v)$ and $\hat{d}(u, v)$ are integer (otherwise $\lfloor \tilde{d}(u, v) \rfloor$ and $\lceil \hat{d}(u, v) \rceil$, respectively, will serve as the desired α-approximations).

For every pair of nodes $u, v \in V$ and every integer $h \geq 1$, we define the shortest h-hop path from u to v to be the path of minimum weight among all paths from u to v with at most h edges. The h-hop distance from u to v in G, denoted by $d_G^h(u, v)$ is the weight of the shortest h-hop path from u to v.

C. Toolkit

In the following, we review known tools that we use to design our algorithm.

1) Reduction from Approximate SSSP to Exact SSSP: We will apply the following reduction by Klein and Subramanian [21] for extending certain approximate SSSP algorithms to exact SSSP algorithms. The reduction is based on Gabow's recursive scaling algorithm [48]. Initially this reduction was formulated for the PRAM model, but it can be adapted to the CONGEST model in a straightforward way.

Theorem 1 (Implicit in [21]). *Assume there is an auxiliary algorithm $\mathcal{A}$ in the CONGEST model that, given a directed graph $G = (V, E, w_G)$ with positive integer edge weights and a source node s, computes a distance estimate $\hat{d}(s, \cdot)$ such that*

$$\tfrac{1}{2} \cdot d_G(s, v) \leq \hat{d}(s, v) \leq d_G(s, v) \tag{1}$$

for every node $v \in V$ and

$$\hat{d}(s, v) \leq \hat{d}(s, u) + w_G(u, v) \tag{2}$$

for every edge $(u, v) \in E$. Then there is an exact SSSP algorithm $\mathcal{B}$ in the CONGEST model that, given a directed graph $G = (V, E, w_G)$ with non-negative integer weights in the range $w_G(e) \in \{0, 1, \ldots, W\}$ for every edge $e \in E$ and a source node s, makes $O(\log nW)$ calls to algorithm $\mathcal{A}$ (on graphs $G' = (V, E, w_{G'})$ with positive integer weights in the range $w_{G'}(e) \in \{1, \ldots, O(n^2)\}$ for every edge $e \in E$) and has an additional overhead of $\tilde{O}(D \log W)$ broadcast rounds. If $\mathcal{A}$ is a randomized Monte Carlo algorithm that is correct with high probability, then $\mathcal{B}$ is a Las Vegas algorithm whose bounds on the number of $O(\log nW)$ calls to $\mathcal{A}$ and the additional overhead of $\tilde{O}(D \log W)$ broadcast rounds, respectively, hold with high probability and in expectation.

We give a proof of Theorem 1 for completeness in Appendix A.

Our task of designing an exact SSSP algorithm thus reduces to designing an auxiliary approximate SSSP algorithm satisfying conditions (1) and (2). Note that (2) is a special form of the *triangle inequality* expressing that the distance estimates *dominate* the edge weights of G. The reduction will fail if this additional constraint is not met, and thus we cannot use arbitrary distance estimates, which poses a challenge in designing the auxiliary algorithm. As Klein and Subramanian [21] observed, one can for example ensure (2) by obtaining $\hat{d}(s, \cdot)$ as the exact distances from s in a suitable supergraph $G' = (V, E', w_{G'})$ of $G = (V, E, w_G)$ where $E \subseteq E'$ and $w_{G'}(e) \leq w_G(e)$ for every edge $e \in E$.

2) Computing Bounded Hop Distances: We will use several primitives for computing (approximate) bounded-hop distances. The well-known "cornerstone" for this task is the Bellman-Ford algorithm.

Lemma 2 (Folklore). *In the Broadcast CONGEST model, one can, given a weighted directed graph G with source node s and an integer hop parameter $h \geq 1$, compute, for every node v, the value $d_G^h(s, v)$ and make it known to v in $O(h)$ rounds such that each node in total broadcasts at most $O(h)$ messages to its neighbors by using a synchronized version of the Bellman-Ford algorithm*

While the dependence on the number of rounds in the Bellman-Ford algorithm is optimal, there is some room for improvement in the number of messages. Using a well-known weight-rounding technique [21], [52], [53], [54], [55], [56], [7], the number of messages can be reduced at the cost of introducing some approximation error. Algorithmically, this technique amounts to performing a single-source shortest path computation up to bounded distance in an integer-weighted graph. For this task, one can use a shortest path algorithm similar to breadth-first search that charges each node (at most) once for sending distance information to its neighbors.

Lemma 3 (Implicit in [7]). *In the Broadcast CONGEST model, there is an algorithm that, given a directed graph G with positive integer edge weights, a fixed source node s, and an integer hop parameter $h \geq 1$, computes, for every node v, a distance estimate $\tilde{d}(s, v)$ ultimately known to v such that*

$$d_G(s, v) \leq \tilde{d}(s, v) \leq (1 + \epsilon) d_G^h(s, v)$$

in $\tilde{O}(h \log W / \epsilon + D)$ rounds where each node in total broadcasts at most $\tilde{O}(\log W)$ messages to its neighbors.

This approximation algorithm can be leveraged to compute approximate h-hop distances from a set of k sources. Note however that if we run k instances of the algorithm of Lemma 3, then in the worst-case it might happen that a single node might have to send different messages to its neighbors in all k instances of the algorithm simultaneously. The naive approach of circumventing this type of congestion will blow up the number of rounds by a factor of k. Nanongkai [7] showed that the congestion can be avoided using a random delay technique.

Lemma 4 ([7]). *In the Broadcast CONGEST model, there is a randomized algorithm that, given a directed graph G with positive integer edge weights, a fixed set of source nodes S of size $k = |S|$, and an integer hop parameter $h \geq 1$, computes, for every source node $s \in S$ and every node v, a distance estimate $\tilde{d}(s, v)$ ultimately known to v such that, with high probability,*

$$d_G(s, v) \leq \tilde{d}(s, v) \leq (1 + \epsilon) d_G^h(s, v)$$

in $\tilde{O}(h \log W / \epsilon + k + D)$ rounds.

Note that, in combination with the aforementioned rounding technique, one can alternatively use the deterministic source

detection algorithm of Lenzen and Peleg [33], [57] for this task.

3) Randomized Hitting Set Construction: Similar to previous randomized shortest path algorithms, we will use the fact that, for a specified parameter h, one can by a simple randomized process obtain a set C of size $\tilde{O}(n/h)$ such that every shortest path consisting of h nodes contains a node of C with high probability, i.e., C is a *hitting* set for the system of sets defined by the shortest paths with h nodes. This technique was introduced to the design of graph algorithms by Ullman and Yannakakis [58]. A general lemma can be formulated as follows.

Lemma 5. *Let $c \geq 1$, let U be a set of size s, and let $\mathcal{S} = \{S_1, S_2, \ldots, S_k\}$ be a collection of sets over the universe U of size at least q. Let T be a subset of U that was obtained by choosing each element of U independently with probability $p = \min(x/q, 1)$ where $x = c \ln(ks) + 1$. Then, with high probability (whp), i.e., probability at least $1 - 1/s^c$, the following two properties hold:*

1) *For every $1 \leq i \leq k$, the set S_i contains an element of T, i.e., $S_i \cap T \neq \emptyset$.*
2) *$|T| \leq 3xs/q = O(cs \log(ks)/q)$.*

To apply the lemma for hitting shortest paths consisting of h nodes in a graph $G = (V, E, w)$, set $U = V$, $q = h$, and for every pair of nodes u and v such that there is shortest path from u to v with exactly h nodes define a corresponding set S_i containing all the nodes on one of these shortest paths, resulting in $k \leq n^2$.

III. Schematic Auxiliary Algorithm

In the following, we present an *auxiliary algorithm* that computes distance estimates satisfying the preconditions (1) and (2) of Theorem 1 and can thus be extended to an exact SSSP algorithm using the recursive-scaling approach. We formulate the auxiliary algorithm in a schematic manner and defer model-specific implementation details and the complexity analysis to later sections. The auxiliary algorithm is parameterized by an integer parameter h whose relevance will only become clear in the complexity analysis. Formally, the guarantees obtained in this section can be summarized as follows.

Lemma 6 (Main Lemma). *For any directed input graph $G = (V, E, w_G)$ with fixed source node s and any integer $1 \leq h \leq n$, the auxiliary algorithm below consisting of **Steps 1–6** computes, for every node $v \in V$, a distance estimate $\hat{d}(s, v)$ satisfying conditions (1) and (2) of Theorem 1.*

The auxiliary algorithm proceeds as follows:

Step 1. Construct a set $C \subseteq V$ of *skeleton nodes* containing (a) the source node s and (b) additionally, for every pair of nodes u and v such that the shortest path from u to v in G consists of exactly $\lceil h/2 \rceil$ nodes, at least one node on one of these shortest paths.

Step 2. For each skeleton node $x \in C$, compute 2-approximate h-hop distances from x, i.e., distance estimates $\tilde{d}(x, \cdot)$ such that

$$d_G(x, v) \leq \tilde{d}(x, v) \leq 2 \cdot d_G^h(x, v) \qquad (3)$$

for every node $v \in V$.

Step 3. Construct the *skeleton graph* $H = (C, C^2, w_H)$ with edge weight $w_H(x, y) = \tilde{d}(x, y)$ for every $(x, y) \in C^2$.

Step 4. Compute distances from s on the skeleton graph H, i.e., $d_H(s, x)$ for every skeleton node $x \in C$.

Step 5. Construct the *augmented graph* $G' = (V, E \cup \{s\} \times C, w_{G'})$ with the weight function given by

$$w_{G'}(u, v) = \begin{cases} d_H(u, v) & \text{if } (u, v) \in (\{s\} \times C) \setminus E \\ 2 \cdot w_G(u, v) & \text{if } (u, v) \in E \setminus (\{s\} \times C) \\ \min(d_H(u, v), 2 \cdot w_G(u, v)) \\ \qquad \text{if } (u, v) \in E \cap (\{s\} \times C). \end{cases}$$

Step 6. Compute the h-hop distances from s in the augmented graph G', i.e., $d_{G'}^h(s, v)$ for every node $v \in V$ and return $\hat{d}(s, v) := \frac{1}{2} \cdot d_{G'}^h(s, v)$ for every node $v \in V$.

The correctness proof has two main parts. We first argue (see Lemma 7) that the distances in G' are a 2-approximation of the distances in G. Then we show (see Lemma 8) that the distance estimates computed by our algorithm, namely $\hat{d}(s, v) := \frac{1}{2} \cdot d_{G'}^h(s, v)$ for every node $v \in V$, are proportional to the distances from s in G' (independent of any hop bound), i.e., $d_{G'}^h(s, v) = d_{G'}(s, v)$. By combining these two facts (see Lemma 9) it follows that the distance estimates returned by the auxiliary algorithm satisfy conditions (1) and (2) of Theorem 1.

Lemma 7. *For every node $v \in V$, $d_G(s, v) \leq d_{G'}(s, v) \leq 2 \cdot d_G(s, v)$.*

Proof. First, observe that by the definition of G' we clearly have $d_{G'}(s, v) \leq 2 \cdot d_G(s, v)$ for every node $v \in V$ as every path of G is also contained in G' and the weight of such a path in G' is at most twice its weight in G by the definition of the edge weights in G'.

We now show that $d_{G'}(s, v) \geq d_G(s, v)$ for every node $v \in V$. Observe that it is sufficient to show that $w_{G'}(u, v) \geq d_G(u, v)$ for every edge (u, v) of G' as then every path from s to v in G' has weight at least $d_G(s, v)$, which in particular also applies to the shortest path from s to v in G'. Note that by the definition of the edge weights in G' we now only have to show that $d_H(s, x) \geq d_G(s, x)$ for every skeleton node $x \in C$. Now observe that for every pair of skeleton nodes $x, y \in C$ we have $\tilde{d}(x, y) \geq d_G(x, y)$ by (3) and thus $w_H(x, y) \geq d_G(x, y)$. This in turn implies that $d_H(s, x) \geq d_G(s, x)$ for every skeleton node $x \in C$, as desired. $\qquad \square$

Lemma 8. *For every node $v \in V$, $d_{G'}^h(s, v) = d_{G'}(s, v)$.*

Proof. The inequality $d_{G'}(s, v) \leq d_{G'}^h(s, v)$ is obvious by the definition of the h-hop distance from s to v. In the remainder of this proof we argue that $d_{G'}^h(s, v) \leq d_{G'}(s, v)$ by constructing

a path in G' with at most h edges and of weight at most $d_{G'}(s, v)$.

Consider a shortest path π' from s to v in G'. We can assume that π' is a simple path and thus π' contains at most one edge $(s, x) \in (\{s\} \times C) \setminus E$ for any $x \in C$, and if so, this edge must be the first edge of π'. We assume in the following that π' starts with some edge $(s, x) \in (\{s\} \times C) \setminus E$, as for the case that all edges of π' are contained in E a simpler version of the argument applies.

Now let π be the shortest path from x to v in G. Subdivide π into consecutive subpaths $\pi_1, \ldots, \pi_k$ such that the subpaths $\pi_2, \ldots, \pi_k$ consist of exactly $\lceil h/2 \rceil$ nodes and the subpath π_1 consists of at most $\lceil h/2 \rceil$ nodes. By the properties of C guaranteed in **Step 1**, we can assume that each of the subpaths $\pi_2, \ldots, \pi_k$ contains a skeleton node of C. Set $y_1 = x$ and for every $2 \leq i \leq k$ let y_i be a skeleton node on π_i. Note that between any pair of consecutive skeleton nodes y_i and y_{i+1} (for $1 \leq i \leq k - 1$) there are at most $h - 1$ edges on π and thus

$$d_G^h(y_i, y_{i+1}) = d_G(y_i, y_{i+1}). \tag{4}$$

We now give an upper bound on the weight of the edge (s, y_k) in G', mainly applying the triangle inequality for the distance metric induced by the skeleton graph H:

$$
\begin{aligned}
w_{G'}(s, y_k) &\leq d_H(s, y_k) \\
&\leq d_H(s, y_1) + d_H(y_1, y_2) + \cdots + d_H(y_{k-1}, y_k) \\
&\leq w_H(s, y_1) + w_H(y_1, y_2) + \cdots + w_H(y_{k-1}, y_k) \\
&= w_H(s, y_1) + \tilde{d}(y_1, y_2) + \cdots + \tilde{d}(y_{k-1}, y_k) \\
&\overset{(3)}{\leq} w_H(s, y_1) + 2 \cdot d_G^h(y_1, y_2) + \cdots + 2 \cdot d_G^h(y_{k-1}, y_k) \\
&\overset{(4)}{=} w_H(s, y_1) + 2 \cdot d_G(y_1, y_2) + \cdots + 2 \cdot d_G(y_{k-1}, y_k) \\
&= w_H(s, y_1) + 2 \cdot d_G(y_1, y_k) \\
&= w_H(s, x) + 2 \cdot d_G(x, y_k)
\end{aligned}
$$

Now consider the path π'' in G' consisting of first the edge from s to y_k of weight $w_{G'}(s, y_k) = d_H(s, y_k)$ and then the subpath of π from y_k to v. We now compare the weight of π' with the weight of π''. Recall that π' consists of first the edge (s, x) and then a path from x to v consisting only of edges contained in E. Therefore, π' has weight at least

$$w_H(s, x) + 2 \cdot d_G(x, v).$$

By the upper bound on $d_H(s, y_k)$ above, π'' has weight at most

$$w_H(s, x) + 2 \cdot d_G(x, y_k) + 2 \cdot d_G(y_k, v) = w_H(s, x) + 2 \cdot d_G(x, v).$$

It follows that the weight of π'' is at most the weight of π', where π' was the shortest path from s to v in G'. Furthermore, π'' consist of at most h edges. Thus, $d_{G'}^h(s, v) \leq d_{G'}(s, v)$ as desired. $\qquad\square$

Lemma 9. *For every node $v \in V$, the distance estimate $\hat{d}(s, v)$ satisfies conditions* (1) *and* (2) *of Theorem 1.*

Proof. By Lemma 7 we have $\frac{1}{2} \cdot d_G(s, v) \leq \frac{1}{2} \cdot d_{G'}(s, v) \leq d_G(s, v)$ for every node $v \in V$. Now observe that by Lemma 8

$\frac{1}{2} \cdot d_{G'}(s, v) = \frac{1}{2} \cdot d_{G'}^h(s, v) = \hat{d}_G(s, v)$ and thus condition (1) is satisfied.

Condition (2) essentially follows from the fact that the distance metric on G' obeys the triangle inequality as together with Lemma 8 we have, for every edge $(u, v) \in E$:

$$
\begin{aligned}
\hat{d}_G(s, v) &= \frac{1}{2} \cdot d_{G'}^h(s, v) \\
&= \frac{1}{2} \cdot d_{G'}(s, v) \\
&\leq \frac{1}{2} \cdot (d_{G'}(s, u) + d_{G'}(u, v)) \\
&\leq \frac{1}{2} \cdot (d_{G'}(s, u) + w_{G'}(u, v)) \\
&\leq \frac{1}{2} \cdot (d_{G'}(s, u) + 2 \cdot w_G(u, v)) \\
&= \frac{1}{2} \cdot d_{G'}(s, u) + w_G(u, v) \\
&= \frac{1}{2} \cdot d_{G'}^h(s, u) + w_G(u, v) \\
&= \hat{d}_G(s, u) + w_G(u, v). \qquad\square
\end{aligned}
$$

IV. FIRST CONGEST MODEL IMPLEMENTATION

In the following we present our first exact SSSP algorithm for the CONGEST model. Its guarantees can be formalized as follows.

Theorem 10. *In the Broadcast CONGEST model, there is a randomized SSSP algorithm for directed graphs with non-negative integer edge weights in the range $\{0, 1, \ldots, W\}$ that performs $\tilde{O}(\sqrt{nD} \log W)$ rounds in expectation.*

In our algorithm, we use the following implementation of the auxiliary algorithm of Section III:

Step 1. We implement the sampling process of Lemma 5: Simultaneously, s adds itself to C and each other node adds itself to C with probability $\min(8 \ln n/h, 1)$. Afterwards, we spend $O(D)$ rounds in a global upcast to determine the size of C. If $|C| > 24n \ln n/h$ (a low-probability event), then abort the algorithm. Now the set C satisfies the condition demanded in **Step 1** of Section III with high probability and thus our implementation of the auxiliary algorithm will also be correct with high probability.

Step 2. Using the algorithm of Lemma 4 as a subroutine, this step can be implemented in $\tilde{O}(h \log W + |C|) = \tilde{O}(h \log W + n/h)$ rounds.

Step 3. The skeleton graph is constructed implicitly – in the sense that each skeleton node only knows its set of incoming edges together with their weight – by globally broadcasting the set of skeleton nodes in the network in $O(|C| + D) = \tilde{O}(n/h + D)$ rounds.

Step 4. We use a message-passing version of Dijkstra's algorithm that in each iteration determines the next node to visit, i.e., the one with minimum tentative distance, by performing a global upcast in the network. This step can thus be implemented in $O(|C|D) = \tilde{O}(nD/h)$ rounds.

Step 5. By performing the corresponding edge weight modifications internally at each node, i.e., without any additional communication, the augmented graph is constructed implicitly in the sense that each node only knows its set of incoming edges together with their weight.

Step 6. We implement this step with the Bellman-Ford algorithm. The first iteration of the Bellman-Ford algorithm can be performed in $O(D)$ rounds as every skeleton node $x \in C$ already knows the weight of the edge (s, x) in G' and thus only needs to be informed about the start of the algorithm. The remaining $h - 1$ iterations take $O(h)$ rounds in a synchronized implementation of Bellman-Ford by Lemma 2, yielding an overall complexity of $O(h + |C| + D) = O(h + D)$ rounds for this step.

Asymptotically, the overall number of rounds is $\tilde{O}(h \log W + nD/h)$. By setting $h = \sqrt{nD}$ we obtain an auxiliary algorithm performing $\tilde{O}(\sqrt{nD} \log W)$ rounds. Theorem 10 now follows as a Corollary from Lemma 6 and Theorem 1.

V. SECOND CONGEST MODEL IMPLEMENTATION

In our second distributed algorithm, we obtain better guarantees for certain parameter ranges by using a different approach for computing exact distances from s on the skeleton graph; the rest of the algorithm is the same as in Section IV. Instead of running Dijkstra's algorithm, we implement **Step 4** by repeating the algorithmic scheme and effectively constructing a skeleton of the skeleton. Intuitively, this somewhat straightforward repetition of the scheme improves the efficiency because computing on the skeleton graph allows slightly different algorithmic techniques than computing on the original network itself as the former is simulated by performing global broadcasts in the network. This gives us some slack to exploit for increased efficiency. We remark that adding more levels of recursion will not boost the efficiency further because computing shortest paths for the skeleton of the skeleton is not a bottleneck in our running time analysis.

In the following, we explicitly separate the two layers – input graph and skeleton graph – mentioned above. We first show how to compute SSSP "on the skeleton graph" and then demonstrate how this can be used for computing SSSP on the input graph.

A. Implementation in Broadcast LOCAL Clique Model

Consider the following *Broadcast LOCAL Clique model* which deviates from the Broadcast CONGEST model in the following ways: (1) the communication network N is a clique, i.e., every message sent by a node is received by *all* other nodes of the network and (2) the size of the message sent per round is arbitrary (and in particular may also be 0). The complexity of an algorithm for a clique network with n nodes is determined by the number of rounds $R(n)$ and the total size of all messages $M(n)$ broadcast by the nodes over the course of the algorithm.

The Broadcast LOCAL Clique model may seem a bit artificial on its own, but it is highly relevant for our CONGEST model implementation of the auxiliary algorithm because of the following straightforward simulation result for computing exact distances on the skeleton graph.

Lemma 11 (Implicit in [7])**.** *Assume there is an exact SSSP algorithm $\mathcal{A}$ for directed graphs with non-negative integer edge*

*weights in the Broadcast LOCAL Clique model spending $R(n)$ rounds and $M(n)$ messages. Then **Step 4** of the auxiliary algorithm can be implemented in $\tilde{O}(M(|C|)/B + R(|C|) \cdot D)$ rounds in the Broadcast CONGEST model, where $B = \Theta(\log n)$ is the bandwidth of the network and D is its diameter.*

Proof Sketch. We simulate a run of the Broadcast LOCAL Clique algorithm $\mathcal{A}$ on the skeleton graph, which is a clique of $|C|$ nodes. We do this by making each message sent by $\mathcal{A}$ global knowledge, which we carry out by globally broadcasting all messages via a breadth-first-search spanning tree of the communication network N. Such a spanning tree can be constructed initially in $O(D)$ rounds. In the i-th of the $R(|C|)$ rounds of $\mathcal{A}$ we have to send some M_i messages and we know that $\sum_{1 \le i \le R(|C|)} M_i = M(|C|)$. In the Broadcast CONGEST model, the total number of rounds for this simulation therefore is $\tilde{O}(\sum_{1 \le i \le R(|C|)}(M_i/B + D)) = \tilde{O}(M(|C|)/B + R(|C|) \cdot D)$ by standard arguments [1]. $\square$

We now show how to obtain an efficient exact SSSP algorithm in the Broadcast LOCAL Clique model by first implementing the auxiliary algorithm of Section III and then extending it to an exact SSSP algorithm using the the recursive-scaling reduction of Theorem 1.

Lemma 12. *In the Broadcast LOCAL Clique model, there is a randomized SSSP algorithm for directed graphs with non-negative edge weights in the range $\{0, 1, \ldots, W\}$ that in expectation spends $R(n) = \tilde{O}(h \log W)$ rounds and $M(n) = \tilde{O}((nh + n^2/h) \log W)$ messages for any integer parameter $1 \le h \le n$.*

Proof. We implement **Step 1**, **Step 3**, and **Step 5** as in Section IV. We implement **Step 2** by simultaneously running an instance $\mathcal{A}_x$ of the algorithm of Lemma 3 from each skeleton node $x \in C$. In each of the $O(h \log W)$ rounds of the algorithm of Lemma 3, we aggregate, for every node, all the messages that it would have to broadcast over all instances $\mathcal{A}_x$ and broadcast them to all other nodes in a single round in the Broadcast LOCAL Clique model. This results in a total size of $\tilde{O}(n|C| \log W) = \tilde{O}(n^2 \log W/h)$ for all these messages in the Broadcast LOCAL Clique model. We implement **Step 4** in the naive way by having each center node broadcast its outgoing edges in H and computing $d_H(s, \cdot)$ internally at every node. This takes $O(1)$ rounds and $O(|C|^2) = \tilde{O}(n^2/h^2)$ messages. Finally, **Step 6** requires a single run of Bellman-Ford, which by Lemma 2 takes $O(h)$ rounds and $O(nh)$ messages. We now apply the reduction of Theorem 1 to obtain the desired SSSP algorithm for the Broadcast LOCAL Clique model. $\square$

We remark that the essential bottleneck of this implementation of the auxiliary turns out to be the final Bellman-Ford computation in **Step 6** when performed in this algorithm for the Broadcast LOCAL Clique model. Recall that this step is necessary to ensure the domination property of (2).

B. Faster CONGEST Model Implementation in High-Diameter Networks

Theorem 13. *In the Broadcast CONGEST model, there is a randomized SSSP algorithm for directed graphs with non-negative edge weights in the range $\{0, 1, \ldots, W\}$ that performs $\tilde{O}((\sqrt{n}D^{1/4} + n^{3/5} + D) \log W)$ rounds in expectation.*

Proof. We implement **Step 1**, **Step 2**, **Step 3**, **Step 5**, and **Step 6** of the auxiliary algorithm as in Section IV. Applying the simulation of Lemma 11 to the algorithm of Lemma 12 (with parameter h'), we can implement **Step 4** in $\tilde{O}(|C|h' + |C|^2/h' + h'D)$ rounds. Now the overall number of rounds is $\tilde{O}(h \log W + |C| + |C|h' + |C|^2/h' + h'D) = \tilde{O}(h \log W + n/h + nh'/h + n^2/(h^2 h') + h'D)$. We use two variants for balancing these terms. In the first variant, we set $h = \sqrt{n}D^{1/4}$ and $h' = \max(1, \sqrt{n}/D^{3/4})$ to get an upper bound of $\tilde{O}(\sqrt{n}D^{1/4} + \max(\sqrt{n}/D^{1/4}, n/D) + D)$ rounds, where the first term dominates the second term when $D \geq n^{2/5}$. In the second variant, we set $h = n^{3/5}$ and $h' = n^{1/5}$ to get an upper bound of $\tilde{O}(n^{3/5} + n^{1/5}D)$ rounds, where the first term dominates the second term when $D \leq n^{2/5}$. Our overall algorithm first computes a 2-approximation to the diameter D of the communication network N in $O(D)$ rounds by performing breadth-first search from an arbitrary node in N and then chooses h and h' according to the approximate value of D to get an upper bound of $\tilde{O}(\sqrt{n}D^{1/4} + n^{3/5} + D)$ rounds in this implementation of the auxiliary algorithm. We now apply the reduction of Theorem 1 to obtain the desired SSSP algorithm for the Broadcast CONGEST model. $\square$

Note that the algorithm of Theorem 13 is (asymptotically) faster than the simpler algorithm of Theorem 10 when $D = \omega(n^{1/5})$.

VI. ADDITIONAL RESULTS

In this section, we work out some additional results, first for approximate SSSP the distributed setting, and then for exact SSSP in the parallel setting.

A. Directed Approximate SSSP

In the following, we give an algorithm for computing approximate SSSP on directed graphs in the CONGEST model that matches the round complexity of the fastest known algorithm for single-source reachability up to polylogarithmic factors.

The algorithmic scheme followed by our algorithm is quite similar to the one of the auxiliary algorithm:

Step 1. Construct a set $C \subseteq V$ of *skeleton nodes* containing (a) the source node s and (b) additionally, for every pair of nodes u and v such that the shortest path from u to v in G consists of exactly $\lceil h/2 \rceil$ nodes, at least one node on one of these shortest paths.

Step 2. For each skeleton node $x \in C$, compute $(1 + \epsilon/3)$-approximate h-hop distances from x, i.e., distance estimates $\tilde{d}(x, \cdot)$ such that

$$d_G(x, v) \leq \tilde{d}(x, v) \leq (1 + \epsilon) \cdot d_G^h(x, v)$$

for every node $v \in V$.

Step 3. Construct the *skeleton graph* $H = (C, C^2, w_H)$ with edge weight $w_H(x, y) = \tilde{d}(x, y)$ for every $(x, y) \in C^2$.

Step 4. Compute $(1 + \epsilon/3)$-approximate distances from s on the skeleton graph H, i.e., $d_H(s, x)$

$$d_H(s, x) \leq d'(s, x) \leq (1 + \epsilon) \cdot d_H(s, x)$$

for every skeleton node $x \in C$.

Step 5. Compute

$$\hat{d}(s, v) := \min_{x \in C}(d'(s, x) + \tilde{d}(x, v))$$

for every node $v \in V$.

Lemma 14. *For any directed input graph $G = (V, E, w_G)$ with fixed source node s, any $0 < \epsilon \leq 1$ and any integer $1 \leq h \leq n$, the algorithm above consisting of **Steps 1–5** computes, for every node $v \in V$, a distance estimate $\hat{d}(s, v)$ such that $d_G(s, v) \leq \hat{d}(s, v) \leq (1 + \epsilon)d_G(s, v)$.*

We omit the proof of this lemma, as the arguments in correctness proof for the algorithm are just a variation of those given in [7]. The $(1 + \epsilon)$-approximation guarantee follows because $(1 + \epsilon/3)^2 \leq 1 + \epsilon$.

We proceed with giving a CONGEST-model implementation of the algorithm above. Here, we use a two-step process similar to the implementation of the auxiliary algorithm in Section V: We first provide an implementation in the Broadcast LOCAL Clique model and then use that algorithm as a black box for the implementation in the Broadcast CONGEST model.

Lemma 15. *In the Broadcast LOCAL Clique model, there is a randomized $(1 + \epsilon)$-approximate SSSP algorithm for directed graphs with non-negative edge weights in the range $\{0, 1, \ldots, W\}$ that spends $R(n) = \tilde{O}(h \log W/\epsilon)$ rounds and $M(n) = \tilde{O}(n^2/(\epsilon h) \log W)$ messages for any integer parameter $1 \leq h \leq n$. The algorithm is correct with high probability.*

Proof. By a standard trick one can transform an instance with non-negative integer edge weights to one with positive integer edge weights: give weight 1 to all 0-weight edges and scale up all other edge weights by a factor of $(1 + \epsilon)n$ (see arXiv version). This increases the maximum edge weight by a factor of $O(n)$ and thus the complexity of the algorithm by a factor of $O(\log n)$.

We implement **Step 1** and **Step 3** as in Section IV. We implement **Step 2** by simultaneously running an instance $\mathcal{A}_x$ of the algorithm of Lemma 3 from each skeleton node $x \in C$. In each of the $O(h \log W/\epsilon)$ rounds of the algorithm of Lemma 3, we aggregate, for every node, all the messages that it would have to broadcast over all instances $\mathcal{A}_x$ and broadcast them to all other nodes in a single round in the Broadcast LOCAL Clique model. This results in a total size of $\tilde{O}(n|C| \log W/\epsilon) = \tilde{O}(n^2 \log W/(\epsilon h))$ for all these messages in the Broadcast LOCAL Clique model. We implement **Step 4** in the naive way by having each center node broadcast its outgoing edges

in H and computing $d'(s, \cdot) := d_H(s, \cdot)$ (i.e., exact distances) internally at every node. This takes $O(1)$ rounds and $O(|C|^2) = \tilde{O}(n^2/h^2)$ messages. Finally, **Step 5** only requires internal computation. $\square$

Theorem 16. *In the Broadcast CONGEST model, there is a randomized $(1 + \epsilon)$-approximate SSSP algorithm for directed graphs with non-negative edge weights in the range $\{0, 1, \ldots, W\}$ that performs $\tilde{O}((\sqrt{n}D^{1/4} + D) \log W/\epsilon)$ rounds. The algorithm is correct with high probability.*

Proof. Again, it is sufficient to provide an algorithm for positive integer edge weights.

We implement **Step 1**, **Step 2**, and **Step 3** as in Section IV. Applying the simulation of Lemma 11, which holds regardless of the approximation ratio, to the algorithm of Lemma 15 (with parameters h' and $\epsilon' = \epsilon/3$), we can implement **Step 4** in $\tilde{O}(|C|^2/(\epsilon h') \log W + h'D \log W/\epsilon)$ rounds. Now the overall number of rounds is $\tilde{O}((h + |C| + |C|^2/h' + h'D) \log W/\epsilon) = \tilde{O}((h + n/h + n^2/(h^2 h') + h'D) \log W/\epsilon)$. By setting $h = \sqrt{n}D^{1/4}$ and $h' = \max(1, \sqrt{n}/D^{3/4})$ this becomes $\tilde{O}((\sqrt{n}D^{1/4} + D) \log W/\epsilon)$ as desired. $\square$

B. PRAM Trade-off

In the following, we show that our approach also gives a new work/depth trade-off for computing SSSP in the PRAM model. In analogy to the RAM model, the PRAM (parallel RAM) allows the possibility of processors computing in parallel with shared memory access. The *work* of a parallel algorithm is the total number of operations performed over all processors and the *depth* is the length of the longest series of operations that have to be performed sequentially due to data dependencies. The depth is essentially the number of parallel computation steps needed until the last processor is finished for any schedule of assigning the algorithm's operations to the processors.

Our contribution in the PRAM model is an exact SSSP algorithm with work $\tilde{O}((n^3/h^3 + mn/h) \log W)$ and depth $\tilde{O}(h)$ for any $1 \le h \le \sqrt{n}$. For *directed* graphs, this specific trade-off was not known before. In particular, the algorithm of Klein and Subramanian [21], who follows the same general framework as we do, has work $\tilde{O}(m\sqrt{n})$ and depth $\tilde{O}(\sqrt{n})$, i.e., it it does not allow for such a trade-off as any choice of h different from $\tilde{\Theta}(\sqrt{n})$ would be sub-optimal. Thus, the conceptual novelties in our algorithm, which were motivated by the application to the CONGEST model, also carry over to an improvement in the PRAM model.

Theorem 17. *In the PRAM model, there is a randomized SSSP algorithm for directed graphs with non-negative edge weights in the range $\{0, 1, \ldots, W\}$ that has work $\tilde{O}((n^3/h^3 + mh + mn/h) \log W)$ with high probability and in expectation and depth $\tilde{O}(h)$ for any given $1 \le h \le n$.*

Proof. To obtain the exact SSSP algorithm, we implement the auxiliary algorithm of Section III as follows:

Step 1. We implement the sampling process of Lemma 5: Each node is added to C independently with probability $\min(8 \ln n/h, 1)$. Afterwards, determine the size of C

and if $|C| > 24n \ln n/h$ (a low-probability event), then abort the algorithm. This has work $O(n)$ and depth $O(\log n)$. Now the set C satisfies the condition demanded in **Step 1** of Section III with high probability and thus our implementation of the auxiliary algorithm will also be correct with high probability.

Step 2. For each node in C, this step can be implemented by computing a shortest path tree up to distance $O(h)$ in a graph with suitably rounded edge weights. Similar to the analysis of bounded breadth-first search, this approach has work $\tilde{O}(m \log W)$ for each node in C and depth $O(h)$ (see Lemma 3.2 in [21], which is similar to Lemma 3 in Section II). Thus the total work of this step is $\tilde{O}((n/h) \cdot m \log W)$ and the depth is $O(h)$.

Step 3. The straightforward approach for constructing and storing the graph H has work $O(|C|^3) = O(n^3/h^3)$ and depth $O(\log n)$.

Step 4. We implement this step by using the all-pairs shortest path algorithm based on repeated squaring with min-plus matrix multiplication. This has work $\tilde{O}(|C|^3) = \tilde{O}(n^3/h^3)$ and depth polylog n.

Step 5. The straightforward approach for constructing and storing the graph G' has work $O(m + |C|) = O(m)$ and depth $O(\log n)$.

Step 6. We implement this step with a parallel version of the Bellman-Ford algorithm that synchronizes the computation in h rounds. This has work $\tilde{O}((m + |C|) \cdot h) = \tilde{O}(mh)$ and depth $\tilde{O}(h)$.

Overall, our auxiliary algorithm has work $\tilde{O}(n^3/h^3 + mh + m(n/h) \log W)$ and depth $\tilde{O}(h)$. By applying the reduction of Klein and Subramanian (see Lemma 4.1 in [21], which is similar to Lemma 1 in Section II), we obtain an exact SSSP algorithm that has work $\tilde{O}((n^3/h^3 + mh + mn/h) \log W)$ and depth $\tilde{O}(h)$. As the auxiliary algorithm is correct with high probability, the work bound of the exact SSSP algorithm holds with high probability (and in expectation). $\square$

VII. CONCLUSION AND DISCUSSION

In this paper, we have given two new algorithms for computing SSSP in the CONGEST model, the first performing $\tilde{O}(\sqrt{nD})$ rounds and the second performing $\tilde{O}(\sqrt{n}D^{1/4} + n^{3/5} + D)$ rounds. Our algorithms follow a general scaling approach for reducing the problem to approximate distance computation. However, not any arbitrary approximate SSSP algorithm is adequate for this reduction as the returned distance estimates must additionally 'dominate' the edge weights of the graph. Our main contribution is a schematic auxiliary algorithm that computes suitable distance estimates, which in particular extends and improves upon prior work in the PRAM model. We then provide different ways of implementing this scheme.

Currently, a natural barrier for directed SSSP algorithms is $\tilde{O}(\sqrt{n}D^{1/4} + D)$ rounds as this is the running time of the fastest known algorithm for single-source reachability [19]. We do meet this barrier in our approximation algorithm, but when solving SSSP exactly, **Step 6** of our auxiliary algorithm is a bottleneck: we cannot implement it fast enough on
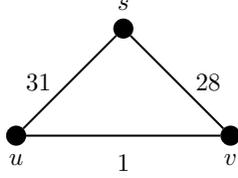
Fig. 1. This graph G shows that **Step 6** of our auxiliary algorithm cannot be omitted. Using the algorithm of Lemma 3 with $\epsilon = 1$ and $h = 2$ we obtain distance estimates $\tilde{d}(s, u) = 28$ and $\tilde{d}(s, v) = 32$. However, $\frac{1}{2} \cdot \tilde{d}(s, v) > \frac{1}{2} \cdot \tilde{d}(s, u) + w_G(u, v)$ and thus these distance estimates do not dominate the edge weights as demanded in the preconditions of Theorem 1.

the skeleton graph to get the $\tilde{O}(\sqrt{n}D^{1/4} + D)$ bound. A simple counter-example (see Fig. 1) shows that **Step 6** cannot simply be omitted. Finding an SSSP algorithm performing only $\tilde{O}(\sqrt{n}D^{1/4} + D)$ rounds therefore remains a major open problem. In the same respect, it is also very interesting to understand whether $\tilde{\Theta}(n^{1/2}D^{1/4} + D)$ is the right bound: it seems both challenging to break the $n^{1/2}D^{1/4}$ bound for the reachability problem or to prove an $n^{1/2}D^c$ lower bound, for any constant c, for the directed exact SSSP problem.

Another challenge lies in finding a *deterministic* SSSP algorithm that outperforms the distributed implementation of the Bellman-Ford algorithm with its round complexity of $O(n)$. In our specific scaling approach, the deterministic procedure of Henzinger, Krinninger, and Nanongkai [8] for computing a skeleton graph cannot be applied. This technique only works for undirected graphs and in our scaling algorithm we necessarily have to deal with directed graphs, even if the input graph is undirected. The recent deterministic construction in the APSP algorithm of Agarwal et al. [34] is also not an option because it takes $\Omega(n)$ rounds.

In general, it is very appealing to understand how fast we can solve graph problems exactly in the CONGEST model. In the past few years there have been many tight approximation algorithms developed in this model, but not much was known for exact algorithms. Two particular problems, as mentioned in [29], are exact maximum matching and minimum cut; the former was recently shown to admit an algorithm performing $O(n \log n)$ rounds [59] while no non-trivial algorithm is known for the latter.

APPENDIX

In the following, we give a proof of Theorem 1, which essentially follows the reduction given by Klein and Subramanian [21]. However, some errors introduced in the typesetting might make the corresponding part of their journal paper hard to read; we therefore include the proof for the readers' convenience. It is sufficient to design an auxiliary algorithm for graphs with positive edge weights as such an algorithm can be extended to work for graphs with non-negative edge weights. This reduction seems to be folklore and we include a formal proof in the arXiv version of our paper.

Lemma 18. *Assume there is an algorithm $\mathcal{A}$ in the CONGEST model that, given a directed graph $G' = (V, E, w_{G'})$ with positive integer edge weights and a source node s, computes a distance estimate $d'(s, \cdot)$ such that*

$$\alpha \cdot d_G(s, v) \leq d'(s, v) \leq d_G(s, v)$$

for every node v, where $\alpha \leq 1$ is known and $1/\alpha$ is integer, and

$$d'(s, v) \leq d'(s, u) + w_G(u, v)$$

for every edge $(u, v) \in E$. Then there is an algorithm $\mathcal{A}'$ in the CONGEST model that, given a directed graph $G = (V, E, w_G)$ with integer weights in the range $w_G(e) \in \{0, 1, \ldots, W\}$ for every edge $e \in E$ and a source node s, computes a distance estimate $\hat{d}(s, \cdot)$ such that

$$\alpha \cdot d_G(s, v) \leq \hat{d}(s, v) \leq d_G(s, v)$$

for every node v and

$$\hat{d}(s, v) \leq \hat{d}(s, u) + w_G(u, v)$$

for every edge $(u, v) \in E$ by making one call to algorithm $\mathcal{A}$ (on a graph $G' = (V, E, w_{G'})$ with integer weights in the range $w_{G'}(e) \in \{1, \ldots, O(nW/\alpha)\}$ for every edge $e \in E$) and performing internal computation.

Now in order to prove Theorem 1, we design a recursive algorithm $\mathcal{B}$ that, given a weighted directed graph $G = (V, E, w_G)$ with source nodes s and an upper bound $\Delta \geq 0$, computes SSSP in a weighted directed graph if the distance from the source node s to any other node is at most Δ, where initially $\Delta = nW$.

If $\Delta < 1$, then the algorithm returns $\tilde{d}(s, v) = 0$ for every node $v \in V$. Otherwise, the algorithm proceeds as follows:

1) Set $\delta := \max(\lfloor \log \Delta - \log n - 1 \rfloor, 0)$.
2) Construct $\hat{G} = (V, \hat{E}, w_{\hat{G}})$ by setting $\hat{E} := \{e \in E : w_G(e) \leq \Delta\}$ and

$$w_{\hat{G}}(e) := \left\lfloor \frac{w_G(e)}{2^\delta} \right\rfloor$$

   for every edge $e \in \hat{E}$.
3) Compute a distance estimate $\hat{d}(s, v)$ for every node $v \in V$ by running algorithm $\mathcal{A}'$ of Lemma 18 on $\hat{G}$.
4) Construct $G' = (V, E, w_{G'})$ by setting

$$w_{G'}(u, v) := w_G(u, v) + 2^\delta \hat{d}(s, u) - 2^\delta \hat{d}(s, v)$$

   for every edge $(u, v) \in E$.
5) Recursively compute $d_{G'}(s, v)$ for every node $v \in V$ by running $\mathcal{B}$ on G' with $\Delta' := \lfloor \frac{3}{4}\Delta \rfloor$.

6) Return $\tilde{d}(s,v) := d_{G'}(s,v) + 2^\delta \hat{d}(s,v)$ for every node $v \in V$.

Observe that Δ reduces by a $\frac{1}{4}$-fraction in each recursive call and thus the number of repetitions is $O(\log nW)$. Furthermore, the weight of each edge $e \in \hat{E}$ consists of $O(\log n)$ bits and thus the algorithm $\mathcal{A}'$ is called on a graph with integer edge weights in the range $\{0, 1, \ldots, O(n)\}$. Obviously, all operations of $\mathcal{B}$, except for the calls to $\mathcal{A}'$ can be implemented in $\tilde{O}(D \log W)$ broadcast rounds in the CONGEST model. To establish the correctness of the algorithm, it is left to prove that the returned distances are correct, that the edge weights in the graph G' are non-negative, and that the distances from the source in G' are indeed bounded by Δ'.

First, we show that $\tilde{d}(s,v) = d_G(s,v)$ for every node $v \in V$. Consider a path π from s to v in G.

$$
\begin{aligned}
w_{G'}(\pi) &= \sum_{e \in \pi} w_{G'}(e) \\
&= \sum_{e \in \pi} \left[w_G(e) + 2^\delta \hat{d}(s,u) - 2^\delta \hat{d}(s,v) \right] \\
&= 2^\delta \hat{d}(s,s) - 2^\delta \hat{d}(s,v) + \sum_{e \in \pi} w_G(e) \\
&= w_G(\pi) - 2^\delta \hat{d}(s,v)
\end{aligned}
$$

Now let π^* be the shortest path from s to v in G. By the arguments above we have $w_{G'}(\pi^*) = w_G(\pi^*) - 2^\delta \hat{d}(s,v) = d_G(s,v) - 2^\delta \hat{d}(s,v)$ and for every other path π from s to v we have $w_{G'}(\pi) = w_G(\pi) - 2^\delta \hat{d}(s,v) \geq w_G(\pi^*) - 2^\delta \hat{d}(s,v)$. Thus, π^* is also a shortest path from s to v in G', i.e. $d_{G'}(s,v) = w_{G'}(\pi^*)$. We therefore have

$$
d_G(s,v) = d_{G'}(s,v) + 2^\delta \hat{d}(s,v) \tag{5}
$$

as desired.

Second, we show that the weights of the graph on which the algorithm is called recursively are non-negative, i.e., $w_{G'}(u,v) \geq 0$ for every edge $(u,v) \in E'$. By (2) we have

$$
\hat{d}(s,v) \leq \hat{d}(s,u) + w_{\hat{G}}(u,v)
$$

which is equivalent to

$$
2^\delta \hat{d}(s,v) \leq 2^\delta \hat{d}(s,u) + 2^\delta w_{\hat{G}}(u,v) \,.
$$

We now have

$$
\begin{aligned}
w_{G'}(u,v) &= w_G(u,v) + 2^\delta \hat{d}(s,u) - 2^\delta \hat{d}(s,v) \\
&\geq w_G(u,v) - 2^\delta w_{\hat{G}}(u,v) \geq w_G(u,v) - w_G(u,v) = 0
\end{aligned}
$$

as desired.

Third, we show that $d_{G'}(s,v) \leq \Delta'$ for every node $v \in V$. Let π be a shortest path from s to v in $\hat{G}$. By (1), and the definition of the edge weights in $\hat{G}$, we have

$$
\begin{aligned}
\hat{d}(s,v) &\geq \tfrac{1}{2} \cdot d_{\hat{G}}(s,v) = \tfrac{1}{2} \cdot w_{\hat{G}}(\pi) = \tfrac{1}{2} \cdot \sum_{e \in \pi} w_{\hat{G}}(e) \\
&= \tfrac{1}{2} \cdot \sum_{e \in \pi} \left\lfloor \tfrac{w_G(e)}{2^\delta} \right\rfloor \,.
\end{aligned}
$$

If $\delta > 0$, $\delta = \lfloor \log \Delta - \log n - 1 \rfloor$ and $2^\delta \leq 2^{\log(\Delta/(2n))} = \frac{\Delta}{2n}$. We thus have

$$
\begin{aligned}
2^\delta \cdot \sum_{e \in \pi} \left\lfloor \tfrac{w_G(e)}{2^\delta} \right\rfloor &\geq 2^\delta \cdot \sum_{e \in \pi} \left(\tfrac{w_G(e)}{2^\delta} - 1 \right) \\
&\geq \sum_{e \in \pi} w_G(e) - 2^\delta n \\
&= w_G(\pi) - 2^\delta n \\
&\geq w_G(\pi) - \tfrac{1}{2}\Delta
\end{aligned}
$$

If $\delta = 0$, then $2^\delta \cdot \sum_{e \in \pi} \left\lfloor \tfrac{w_G(e)}{2^\delta} \right\rfloor = \sum_{e \in \pi} w_G(e) = w_G(\pi)$. In any case we have

$$
2^\delta \cdot \sum_{e \in \pi} \left\lfloor \tfrac{w_G(e)}{2^\delta} \right\rfloor \geq w_G(\pi) - \tfrac{1}{2}\Delta \,.
$$

Thus, by (5), we get

$$
\begin{aligned}
d_{G'}(s,v) &= d_G(s,v) - 2^\delta \hat{d}(s,v) \\
&\leq d_G(s,v) - \tfrac{1}{2} \cdot 2^\delta \cdot \sum_{e \in \pi} \left\lfloor \tfrac{w_G(e)}{2^\delta} \right\rfloor \\
&\leq d_G(s,v) - \tfrac{1}{2} \cdot w_G(\pi) + \tfrac{1}{4}\Delta \\
&\leq d_G(s,v) - \tfrac{1}{2} \cdot d_G(s,v) + \tfrac{1}{4}\Delta \\
&= \tfrac{1}{2} \cdot d_G(s,v) + \tfrac{1}{4}\Delta \\
&\leq \tfrac{1}{2}\Delta + \tfrac{1}{4}\Delta \\
&= \tfrac{3}{4}\Delta = \Delta'
\end{aligned}
$$

as desired.

The claimed Las Vegas guarantee follows from standard arguments.

References

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

[2] L. R. Ford, "Network flow theory," The RAND Corporation, Tech. Rep. P-923, 1956.

[3] R. Bellman, "On a routing problem," *Quarterly of Applied Mathematics*, vol. 16, no. 1, pp. 87–90, 1958.

[4] E. F. Moore, "The shortest path through a maze," in *International Symposium on the Theory of Switching*, 1959, pp. 285–292.

[5] 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*, vol. 30, no. 5, pp. 1427–1442, 2000, announced at FOCS'99.

[6] C. Lenzen and B. Patt-Shamir, "Fast routing table construction using small messages," in *Symposium on Theory of Computing (STOC)*, 2013, pp. 381–390.

[7] D. Nanongkai, "Distributed approximation algorithms for weighted shortest paths," in *Symposium on Theory of Computing (STOC)*, 2014, pp. 565–573.

[8] M. Henzinger, S. Krinninger, and D. Nanongkai, "A deterministic almost-tight distributed algorithm for approximating single-source shortest paths," *SIAM Journal on Computing*, 2018, to appear, announced at STOC'16.

[9] M. Elkin and O. Neiman, "Hopsets with constant hopbound, and applications to approximate shortest paths," in *Symposium on Foundations of Computer Science (FOCS)*, 2016, pp. 128–137.

[10] R. Becker, A. Karrenbauer, S. Krinninger, and C. Lenzen, "Near-optimal approximate shortest paths and transshipment in distributed and streaming models," in *International Symposium on Distributed Computing (DISC)*, 2017, pp. 7:1–7:16.

[11] B. Haeupler and J. Li, "Faster distributed shortest path approximations via shortcuts," in *International Symposium on Distributed Computing (DISC)*, 2018.

[12] B. Haeupler, D. E. Hershkowitz, and D. Wajc, "Round- and message-optimal distributed graph algorithms," in *Symposium on Principles of Distributed Computing (PODC)*, 2018, pp. 119–128.

[13] 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," *SIAM Journal on Computing*, vol. 41, no. 5, pp. 1235–1265, 2012, announced at STOC'11.

[14] M. Elkin, "An unconditional lower bound on the time-approximation trade-off for the distributed minimum spanning tree problem," *SIAM Journal on Computing*, vol. 36, no. 2, pp. 433–456, 2006, announced at STOC'04.

[15] L. Kor, A. Korman, and D. Peleg, "Tight bounds for distributed minimum-weight spanning tree verification," *Theory of Computing Systems*, vol. 53, no. 2, pp. 318–340, 2013, announced at STACS'11.

[16] M. Elkin, H. Klauck, D. Nanongkai, and G. Pandurangan, "Can quantum communication speed up distributed computation?" in *Symposium on Principles of Distributed Computing (PODC)*, 2014, pp. 166–175.

[17] M. Elkin, "Distributed exact shortest paths in sublinear time," in *Symposium on the Theory of Computing (STOC)*, 2017, pp. 757–770.

[18] M. Ghaffari and J. Li, "Improved distributed algorithms for exact shortest paths," in *Symposium on the Theory of Computing (STOC)*, 2018, pp. 431–444.

[19] M. Ghaffari and R. Udwani, "Brief announcement: Distributed single-source reachability," in *Symposium on Principles of Distributed Computing (PODC)*, 2015, pp. 163–165.

[20] T. H. Spencer, "Time-work tradeoffs for parallel algorithms," *Journal of the ACM*, vol. 44, no. 5, pp. 742–778, 1997, announced at SODA'91 and SPAA'91.

[21] P. N. Klein and S. Subramanian, "A randomized parallel algorithm for single-source shortest paths," *Journal of Algorithms*, vol. 25, no. 2, pp. 205–220, 1997.

[22] E. Cohen, "Using selective path-doubling for parallel shortest-path computations," *Journal of Algorithms*, vol. 22, no. 1, pp. 30–56, 1997, announced at ISTCS'93.

[23] G. S. Brodal, J. L. Träff, and C. D. Zaroliagis, "A parallel priority queue with constant time operations," *Journal of Parallel and Distributed Computing*, vol. 49, no. 1, pp. 4–21, 1998, announced at IPPS'97.

[24] H. Shi and T. H. Spencer, "Time-work tradeoffs of the single-source shortest paths problem," *Journal of Algorithms*, vol. 30, no. 1, pp. 19–32, 1999.

[25] E. Cohen, "Polylog-time and near-linear work approximation scheme for undirected shortest paths," *Journal of the ACM*, vol. 47, no. 1, pp. 132–166, 2000, announced at STOC'94.

[26] U. Meyer and P. Sanders, "Δ-stepping: a parallelizable shortest path algorithm," *Journal of Algorithms*, vol. 49, no. 1, pp. 114–152, 2003, announced at ESA'98.

[27] G. L. Miller, R. Peng, A. Vladu, and S. C. Xu, "Improved parallel algorithms for spanners and hopsets," in *Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2015, pp. 192–201.

[28] G. E. Blelloch, Y. Gu, Y. Sun, and K. Tangwongsan, "Parallel shortest paths using radius stepping," in *Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2016, pp. 443–454.

[29] C. Huang, D. Nanongkai, and T. Saranurak, "Distributed exact weighted all-pairs shortest paths in $\tilde{O}(n^{5/4})$ rounds," in *Symposium on Foundations of Computer Science (FOCS)*, 2017, pp. 168–179.

[30] S. Frischknecht, S. Holzer, and R. Wattenhofer, "Networks cannot compute their diameter in sublinear time," in *Symposium on Discrete Algorithms (SODA)*, 2012, pp. 1150–1162.

[31] D. Peleg, L. Roditty, and E. Tal, "Distributed algorithms for network diameter and girth," in *International Colloquium on Automata, Languages, and Programming (ICALP)*, 2012, pp. 660–672.

[32] S. Holzer and R. Wattenhofer, "Optimal distributed all pairs shortest paths and applications," in *Symposium on Principles of Distributed Computing (PODC)*, 2012, pp. 355–364.

[33] C. Lenzen and D. Peleg, "Efficient distributed source detection with limited bandwidth," in *Symposium on Principles of Distributed Computing (PODC)*, 2013, pp. 375–382.

[34] U. Agarwal, V. Ramachandran, V. King, and M. Pontecorvi, "A deterministic distributed algorithm for exact weighted all-pairs shortest paths in $\tilde{O}(n^{3/2})$ rounds," in *Symposium on Principles of Distributed Computing (PODC)*, 2018, pp. 199–205.

[35] K. Censor-Hillel, P. Kaski, J. H. Korhonen, C. Lenzen, A. Paz, and J. Suomela, "Algebraic methods in the congested clique," in *Symposium on Principles of Distributed Computing (PODC)*, 2015, pp. 143–152.

[36] E. Gafni, "Improvements in the time complexity of two message-optimal election algorithms," in *Symposium on Principles of Distributed Computing (PODC)*, 1985, pp. 175–185.

[37] F. Y. L. Chin and H. F. Ting, "An almost linear time and $O(n \log n + e)$ messages distributed algorithm for minimum-weight spanning trees," in *Symposium on Foundations of Computer Science (FOCS)*, 1985, pp. 257–266.

[38] B. Awerbuch, "Optimal distributed algorithms for minimum weight spanning tree, counting, leader election and related problems," in *Symposium on Theory of Computing (STOC)*, 1987, pp. 230–240.

[39] J. A. Garay, S. Kutten, and D. Peleg, "A sublinear time distributed algorithm for minimum-weight spanning trees," *SIAM Journal on Computing*, vol. 27, no. 1, pp. 302–316, 1998, announced at FOCS'93.

[40] S. Kutten and D. Peleg, "Fast distributed construction of small k-dominating sets and applications," *Journal of Algorithms*, vol. 28, no. 1, pp. 40–66, 1998, announced at PODC'95.

[41] M. Ghaffari and F. Kuhn, "Distributed minimum cut approximation," in *Symposium on Distributed Computing (DISC)*, 2013, pp. 1–15.

[42] M. Ghaffari, "Near-optimal distributed approximation of minimum-weight connected dominating set," in *International Colloquium on Automata, Languages, and Programming (ICALP)*, 2014, pp. 483–494.

[43] D. Nanongkai and H. Su, "Almost-tight distributed minimum cut algorithms," in *International Symposium on Distributed Computing (DISC)*, 2014, pp. 439–453.

[44] M. Ghaffari, A. Karrenbauer, F. Kuhn, C. Lenzen, and B. Patt-Shamir, "Near-optimal distributed maximum flow: Extended abstract," in *Symposium on Principles of Distributed Computing (PODC)*, 2015, pp. 81–90.

[45] G. Pandurangan, P. Robinson, and M. Scquizzato, "A time- and message-optimal distributed algorithm for minimum spanning trees," in *Symposium on Theory of Computing (STOC)*, 2017, pp. 743–756.

[46] M. Elkin, "A simple deterministic distributed MST algorithm, with near-optimal time and message complexities," in *Symposium on Principles of Distributed Computing (PODC)*, 2017, pp. 157–163.

[47] R. Bar-Yehuda, K. Censor-Hillel, M. Ghaffari, and G. Schwartzman, "Distributed approximation of maximum independent set and maximum matching," in *Symposium on Principles of Distributed Computing (PODC)*, 2017, pp. 165–174.

[48] H. N. Gabow, "Scaling algorithms for network problems," *Journal of Computer and System Sciences*, vol. 31, no. 2, pp. 148–168, 1985, announced at FOCS'83.

[49] H. N. Gabow and R. E. Tarjan, "Faster scaling algorithms for network problems," *SIAM Journal on Computing*, vol. 18, no. 5, pp. 1013–1036, 1989.

[50] A. V. Goldberg, "Scaling algorithms for the shortest paths problem," *SIAM Journal on Computing*, vol. 24, no. 3, pp. 494–504, 1995, announced at SODA'93.

[51] N. Linial, "Locality in distributed graph algorithms," *SIAM Journal on Computing*, vol. 21, no. 1, pp. 193–201, 1992, announced at FOCS'87.

[52] E. Cohen, "Fast algorithms for constructing t-spanners and paths with stretch t," *SIAM Journal on Computing*, vol. 28, no. 1, pp. 210–236, 1998, announced at FOCS'93.

[53] U. Zwick, "All pairs shortest paths using bridging sets and rectangular matrix multiplication," *Journal of the ACM*, vol. 49, no. 3, pp. 289–317, 2002, announced at FOCS'98.

[54] A. Bernstein, "Fully dynamic $(2 + \epsilon)$ approximate all-pairs shortest paths with fast query and close to linear update time," in *Symposium on Foundations of Computer Science (FOCS)*, 2009, pp. 693–702.

[55] A. Madry, "Faster approximation schemes for fractional multicommodity flow problems via dynamic graph algorithms," in *Symposium on Theory of Computing (STOC)*, 2010, pp. 121–130.

[56] A. Bernstein, "Maintaining shortest paths under deletions in weighted directed graphs," *SIAM Journal on Computing*, vol. 45, no. 2, pp. 548–574, 2016, announced at STOC'13.

[57] C. Lenzen and B. Patt-Shamir, "Fast partial distance estimation and applications," in *Symposium on Principles of Distributed Computing (PODC)*, 2015, pp. 153–162.

[58] J. D. Ullman and M. Yannakakis, "High-probability parallel transitive-closure algorithms," *SIAM Journal on Computing*, vol. 20, no. 1, pp. 100–125, 1991, announced at SPAA'90.

[59] M. Ahmadi, F. Kuhn, and R. Oshman, "Distributed approximate maximum matching in the congest model," in *International Symposium on Distributed Computing (DISC)*, 2018.