Computing Minimum Weight Cycle in the CONGEST Model

Vignesh Manoharan
The University of Texas at Austin
Austin, TX, USA
vigneshm@cs.utexas.edu

Vijaya Ramachandran
The University of Texas at Austin
Austin, TX, USA
vlr@cs.utexas.edu

ABSTRACT

Minimum Weight Cycle (MWC) is the problem of finding a simple cycle of minimum weight in a graph $G = (V, E)$. This is a fundamental graph problem with classical sequential algorithms that run in $\tilde{O}(n^3)$ and $\tilde{O}(mn)$ time[†] where $n = |V|$ and $m = |E|$. In recent years this problem has received significant attention in the context of fine-grained sequential complexity [3, 50] as well as in the design of faster sequential approximation algorithms [13, 26, 32, 33], though not much is known in the distributed CONGEST model.

We present near-optimal $\tilde{\Omega}(n)$ CONGEST lower bounds on the round complexity of computing exact and $(2-\epsilon)$-approximate MWC in undirected weighted graphs and in directed graphs even if unweighted. We complement these lower bounds with sublinear-round algorithms for computing 2-approximation of MWC. Our algorithms use a variety of techniques in non-trivial ways, such as in our approximate directed unweighted MWC algorithm that efficiently computes BFS from all vertices restricted to certain implicitly computed neighborhoods in sublinear rounds, and in our weighted approximation algorithms that use unweighted MWC algorithms on scaled graphs combined with a fast and streamlined method for computing multiple source approximate SSSP.

CCS CONCEPTS

• **Theory of computation → Distributed algorithms**; **Graph algorithms analysis**; **Shortest paths**.

KEYWORDS

Distributed Algorithms, Graph Algorithms

ACM Reference Format:
Vignesh Manoharan and Vijaya Ramachandran. 2024. Computing Minimum Weight Cycle in the CONGEST Model. In *ACM Symposium on Principles of Distributed Computing (PODC '24), June 17–21, 2024, Nantes, France.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3662158.3662801

1 INTRODUCTION

We present algorithms and lower bounds to compute a minimum weight cycle in the distributed CONGEST model. Given a graph $G = (V, E)$ with a non-negative weight $w(e)$ on each edge $e \in E$, the minimum weight cycle problem (MWC) asks for a cycle of minimum weight in G. An α-approximation algorithm ($\alpha > 1$) for MWC must find a cycle whose weight is within an α multiplicative factor of the true MWC. In the distributed setting, cycles are an important feature in network analysis with connections to deadlock detection and computing a cycle basis [22, 42, 44], and a shortest cycle can model the likelihood of deadlocks in routing or in database applications [38].

In the sequential context, MWC is a fundamental and well-studied problem on both directed and undirected graphs, both weighted and unweighted. MWC has classical sequential algorithms running in $\tilde{O}(n^3)$ time and $\tilde{O}(mn)$ time[†], where $|V| = n$ and $|E| = m$. There are also sequential fine-grained hardness results: MWC is in the n^3 class [50] and in the mn class [3] for hardness in graph path problems. In the distributed setting there are near-optimal results in the CONGEST model for most of the graph path problems in the sequential n^3 and mn classes including APSP [8, 41], radius and eccentricities [1, 2], betweenness centrality [27], replacement paths and second simple shortest path [39], but very little was known for MWC prior to our work.

In directed graphs, exact MWC in the CONGEST model can be computed in $\tilde{O}(n)$ rounds by computing APSP [8, 37] and computing the minimum among cycles formed by concatenating a v-u shortest path and a single edge (u, v). In this paper, we show a nearly optimal $\tilde{\Omega}(n)$ lower bound for weighted and unweighted directed graphs, to compute even a $(2-\epsilon)$-approximation of MWC (for any constant $\epsilon > 0$). For an arbitrarily large α-approximation (constant $\alpha \geq 2$), we show an $\tilde{\Omega}(\sqrt{n})$ lower bound. We complement the lower bounds with sublinear approximation algorithms, with a non-trivial $\tilde{O}(n^{4/5} + D)$-round algorithm for computing a 2-approximation of directed unweighted MWC, and a $(2+\epsilon)$-approximation of directed weighted MWC.

In undirected unweighted graphs, where MWC is also known as *girth*, the current best upper and lower bounds for exact computation in the CONGEST model are $O(n)$ [28] and $\tilde{\Omega}(\sqrt{n})$ [23] respectively. For 2-approximation the previous best upper bound was $\tilde{O}(\sqrt{ng} + D)$ [44] (g is the weight of MWC), which we improve in this paper to $\tilde{O}(\sqrt{n} + D)$, which is nearly optimal. We show a lower bound of $\tilde{\Omega}(n^{1/3})$ for $(2.5-\epsilon)$-approximation and $\tilde{\Omega}(n^{1/4})$ for arbitrarily large constant α-approximation.

For undirected weighted graphs, exact MWC can be computed in the CONGEST model using a reduction to APSP in $\tilde{O}(n)$ rounds [3, 50]. Our lower bounds results are similar to the directed case: we show near linear lower bound for $(2-\epsilon)$-approximation and $\tilde{\Omega}(\sqrt{n})$ lower bound for α-approximation (for any constant $\alpha \geq 2$). We complement these bounds with an $\tilde{O}(n^{2/3} + D)$-round algorithm for $(2+\epsilon)$-approximation of MWC.

Our approximation algorithms use a procedure to compute directed BFS or approximate SSSP from multiple sources, for which we provide a streamlined algorithm that is significantly more efficient than repeating the current best approximate SSSP algorithm.

[†]We use $\tilde{O}$, $\tilde{\Omega}$ and $\tilde{\Theta}$ to absorb $polylog(n)$ factors.

PODC '24, June 17–21, 2024, Nantes, France

ACM ISBN 979-8-4007-0668-4/24/06
https://doi.org/10.1145/3662158.3662801

Table 1: MWC results for CONGEST. Approximation results hold for approximation ratio α or $(1+\epsilon)$, where $\alpha > 1$ is an arbitrarily large constant, and $\epsilon > 0$ is an arbitrarily small constant.

Problem	Lower Bound	Ref.	Upper Bound	Ref.
Directed MWC weighted/unweighted	$(2-\epsilon), \Omega\left(\frac{n}{\log n}\right)$	Thm 1.2.A	$1, \tilde{O}(n)$	[8]
			$2, \tilde{O}(n^{4/5}+D)$ (unweighted)	Thm 1.2.C
	$\alpha, \Omega\left(\frac{\sqrt{n}}{\log n}\right)$	Thm 1.2.B	$(2+\epsilon), \tilde{O}(n^{4/5}+D)$ (weighted)	Thm 1.2.D
Undirected weighted MWC	$(2-\epsilon), \Omega\left(\frac{n}{\log n}\right)$	Thm 1.4.A	$1, \tilde{O}(n)$	[8]
	$\alpha, \Omega\left(\frac{\sqrt{n}}{\log n}\right)$	Thm 1.4.B	$(2+\epsilon), \tilde{O}(n^{2/3}+D)$	Thm 1.4.C
Undirected unweighted MWC (Girth)	$(2-\epsilon), \Omega\left(\frac{\sqrt{n}}{\log n}\right)$	[23]	$1, O(n)$	[28]
	$(2-O(1/g)), \Omega\left(\frac{\sqrt{(n/g)}}{\log n}\right)$	[23]	$(2-\frac{1}{g}), \tilde{O}(\sqrt{ng}+D)$	[44]
	$\alpha, \Omega\left(\frac{n^{1/4}}{\log n}\right)$	Thm 1.3.A	$(2-\frac{1}{g}), \tilde{O}(\sqrt{n}+D)$	Thm 1.3.B

1.1 Preliminaries

The CONGEST Model. In the CONGEST model [43], a communication network is represented by a graph $G = (V, E)$ where nodes model processors and edges model bounded-bandwidth communication links between processors. Each node has a unique identifier in $\{0, 1, \ldots n-1\}$ where $n = |V|$, and each node only knows the identifiers of itself and its neighbors in the network. Each node has infinite computational power. The nodes perform computation in synchronous rounds, where each node can send a message of up to $\Theta(\log n)$ bits to each neighbor and can receive the messages sent to it by its neighbors. The complexity of an algorithm is measured by the number of rounds until the algorithm terminates.

The graph G can be directed or undirected but the communication links are always bi-directional (undirected) and unweighted; this follows the convention for CONGEST algorithms [4, 6, 14, 20]. We consider algorithms on both weighted and unweighted graphs G in this paper, where in weighted graphs each edge has a weight assignment $w : E(G) \rightarrow \{0, 1, \ldots W\}$ where $W = poly(n)$, and the weight of an edge is known to the vertices incident to it. Our algorithms readily generalize to larger edge weights in networks with bandwidth $\Theta(\log n + \log W)$, with our round complexities for weighted graphs having an additional factor of $\log(nW)$ for arbitrary integer W. The undirected diameter of the network, which we denote by D, is an important parameter in the CONGEST model.

In our algorithms, we frequently use the well-known broadcast and convergecast CONGEST operations [43]: Broadcasting M messages in total to all nodes, where each message could originate from any node, can be performed in $O(M + D)$ rounds. In the convergecast operation, each node holds an $O(\log n)$-bit value and we want to compute an associative operation (such as minimum or maximum) over all values. This can be performed in $O(D)$ rounds, after which all nodes know the result of the operation. We now define the minimum weight cycle problem considered in this paper.

Definition 1.1. **Minimum Weight Cycle problem (MWC)**: Given an n-node graph $G = (V, E)$ (G may be directed or undirected, weighted or unweighted), compute the weight of a shortest simple cycle in G. In the case of α-approximation algorithms, we need to compute the weight of a cycle that is within a factor α of the minimum (for $\alpha > 1$).

In our distributed CONGEST algorithms for MWC, at the end of execution, every node in the network knows the computed weight of MWC (or approximate weight in case of approximation algorithm). Our CONGEST lower bounds for MWC apply even when only one node is required to know the weight of MWC. Our algorithms also allow us to construct the cycle by storing the next vertex on the cycle at each vertex that is part of the MWC.

1.2 Our Results

Table 1 summarizes our upper and lower bound results. All of our lower bounds hold for randomized algorithms, and the algorithms we present are also randomized – which are correct with high probability in n.

1.2.1 Directed Graphs. We show a strong $\Omega(\frac{n}{\log n})$ lower bound for the exact computation of directed MWC in both weighted and unweighted graphs, which is nearly optimal since we have a matching upper bound. We show that this lower bound also holds for any algorithm computing a $(2-\epsilon)$-approximation of directed MWC, where $\epsilon > 0$ is an arbitrarily small constant. Implicit in our MWC lower bound is an $\tilde{\Omega}(n)$ lower bound for detecting if a given graph has a directed cycle of length q, for any $q \geq 4$. We also address the problem of larger approximations, with an $\tilde{\Omega}(\sqrt{n})$ lower bound for α-approximation of directed MWC, for arbitrarily large constant $\alpha \geq 2$.

Our major algorithmic result is a sublinear round algorithm for computing 2-approximation of MWC in directed unweighted graphs that runs in $\tilde{O}(n^{4/5} + D)$ rounds, which we extend to a $(2+\epsilon)$-approximation algorithm in directed weighted graphs with the same round complexity. These results show that a linear lower bound is not possible for $\alpha \geq 2$ approximations.

Theorem 1.2. *Let $G = (V, E)$ be a directed graph. In the CONGEST model, for any constants $\epsilon > 0, \alpha \geq 2$:*

A. *Computing a $(2-\epsilon)$-approximation of directed MWC (weighted or unweighted) requires $\Omega(\frac{n}{\log n})$ rounds, even on graphs with constant diameter D.*
B. *Computing an α-approximation of directed MWC (weighted or unweighted) requires $\Omega(\frac{\sqrt{n}}{\log n})$ rounds, even on graphs with diameter $D = \Theta(\log n)$.*
C. *We can compute 2-approximation of unweighted MWC in $\tilde{O}(n^{4/5} + D)$ rounds.*
D. *We can compute $(2+\epsilon)$-approximation of weighted MWC in $\tilde{O}(n^{4/5} + D)$ rounds.*

1.2.2 Undirected Unweighted Graphs. Girth (undirected unweighted MWC) can be computed in $O(n)$ rounds [28] and an algorithm for $(2 - \frac{1}{g})$-approximation that takes $\tilde{O}(\sqrt{ng} + D)$ rounds is given in [44] (where g is the girth). A lower bound of $\Omega(\frac{\sqrt{n}}{\log n})$ for $(2-\epsilon)$-approximation of girth is given in [23], and this proof also implies an $\Omega(\frac{\sqrt{(n/g)}}{\log n})$ lower bound for $(2-O(1/g))$-approximation of girth.

We improve on the result in [44] by presenting a faster $\tilde{O}(\sqrt{n}+D)$-round algorithm to compute $(2 - \frac{1}{g})$-approximation of girth. For larger approximation ratios, we show a lower bound of $\tilde{\Omega}(n^{1/4})$ for arbitrarily large constant approximation.

Theorem 1.3. *Consider an undirected unweighted graph $G = (V, E)$. In the CONGEST model:*

A. *For any constant $\alpha \geq 2$, computing an α-approximation of girth requires $\Omega(\frac{n^{1/4}}{\log n})$ rounds, even on graphs with diameter $D = \Theta(\log n)$.*
B. *We can compute a $(2-\frac{1}{g})$-approximation of girth in $\tilde{O}(\sqrt{n}+D)$ rounds, where g is the girth.*

1.2.3 Undirected Weighted Graphs. For computing MWC in undirected *weighted* graphs, we present a near-linear lower bound similar to the directed case, and the lower bound also applies to $(2-\epsilon)$-approximation. This bound is optimal up to a polylog factor. Building on our method for the unweighted case, we present an algorithm for $(2+\epsilon)$-approximation of MWC that runs in $\tilde{O}(n^{2/3} + D)$ rounds.

Theorem 1.4. *let $G = (V, E)$ be an undirected weighted graph $G = (V, E)$. In the CONGEST model, for any constants $\epsilon > 0, \alpha \geq 2$:*

A. *Computing a $(2-\epsilon)$-approximation of MWC requires $\Omega(\frac{n}{\log n})$ rounds, even on graphs with constant diameter.*
B. *Computing an α-approximation of MWC requires $\Omega(\frac{\sqrt{n}}{\log n})$ rounds, even on graphs with diameter $D = \Theta(\log n)$.*
C. *We can compute a $(2+\epsilon)$-approximation of MWC in $\tilde{O}(n^{2/3} + D)$ rounds.*

1.2.4 Approximate k-source SSSP. A key subroutine in our approximation algorithms for MWC computes shortest paths efficiently from k sources.

Definition 1.5. **k-source BFS, SSSP problem**: Given an n-node graph $G = (V, E)$ and a set of k vertices $U \subseteq V$, compute at each $v \in V$ the shortest path distance $d(u, v)$ for each $u \in U$. The problem is **k-source BFS** in unweighted graphs and **k-source SSSP** in weighted graphs.

Optimal algorithms to compute k-source BFS taking $O(k + D)$ rounds are known for undirected unweighted graphs [27, 37]. For undirected weighted graphs, an algorithm in [18] computes $(1 + \epsilon)$-approximate SSSP in $\tilde{O}(\sqrt{nk} + D)$ rounds. When the number of sources is large, $k \geq n^{1/3}$, we present a fast and streamlined algorithm for directed exact BFS that runs in $\tilde{O}(\sqrt{nk}+D)$ rounds. We extend the algorithm to weighted graphs, computing approximate SSSP in both directed and undirected weighted graphs in $\tilde{O}(\sqrt{nk} + D)$ rounds. Our algorithm is more efficient than adapting the result in [18] to directed weighted graphs for $k \geq n^{1/3}$, and matches their complexity for undirected graphs. In our applications to MWC algorithms, we only use the streamlined algorithm for $k \geq n^{1/3}$. We present our results for the complete range of $1 \leq k \leq n$ in the full version [40]. In the following theorem, $SSSP = \tilde{O}(\sqrt{n} + n^{2/5+o(1)}D^{2/5} + D)$ refers to round complexity of computing SSSP (from a single source) [9].

Theorem 1.6. A. *We can compute exact directed BFS from k sources in directed unweighted graphs with round complexity:*

$$\begin{cases} \tilde{O}(\sqrt{nk} + D) & ; k \geq n^{1/3} \\ \min\left(\tilde{O}\left(\frac{n}{k} + D\right), k \cdot SSSP\right) & ; k < n^{1/3} \end{cases} \quad (1)$$

B. *We can compute $(1+\epsilon)$-approximate weighted SSSP from k sources in directed weighted graphs for any constant $\epsilon > 0$ with round complexity:*

$$\begin{cases} \tilde{O}(\sqrt{nk} + D) & ; k \geq n^{1/3} \\ \tilde{O}(\sqrt{nk} + k^{2/5}n^{2/5+o(1)}D^{2/5} + D) & ; k < n^{1/3} \end{cases} \quad (2)$$

1.3 Significance of our Results

In the distributed setting, cycles are an important network feature, with applications to deadlock detection and cycle basis computation [22, 42, 44]. In the sequential context, MWC is a fundamental graph problem that is well-studied. The $\tilde{O}(n^3)$ and $\tilde{O}(mn)$ time sequential algorithms for MWC have stood the test of time. MWC is in the sequential n^3 time fine-grained complexity class [50] and plays a central role as the starting point of hardness for the mn time fine-grained complexity class [3].

The n^3 and mn time fine-grained complexity classes contain important graph problems, which have been studied in the CONGEST model and for which nearly optimal upper and lower bounds have been obtained: All Pairs Shortest Paths (APSP) [8], Radius and Eccentricities [1, 6], Betweenness Centrality [27], Replacement Paths (RP) and Second Simple Shortest Path (2-SiSP) [39]. However, there is a conspicuous lack of results for MWC (except for girth [23, 28, 44] and reductions to APSP for exact MWC algorithms).

In this paper, we make significant progress on this problem with a variety of results, including nearly optimal linear lower bounds for exact MWC and algorithms and lower bounds for approximate MWC. While we show that linear lower bounds hold for $(2-\epsilon)$-approximation, we present sublinear algorithms for computing 2 or $(2+\epsilon)$-approximation of MWC. Our algorithms use a variety of techniques in non-trivial ways, such as our directed unweighted MWC algorithm that computes BFS from all vertices restricted to certain implicitly computed neighborhoods in sublinear rounds, and our weighted algorithms that use unweighted MWC algorithms

on scaled graphs combined with multiple source approximate SSSP. We also present lower bounds for larger approximation factors, with $\tilde{\Omega}(\sqrt{n})$ bounds for arbitrarily large constant factor (≥ 2).

Our $\tilde{\Omega}(n)$ lower bound for directed MWC also gives a linear lower bound for directed 4-cycle detection (in fact for any ≥ 4 length cycle), which is surprising given that triangle detection can be performed optimally in $\tilde{O}(n^{1/3})$ rounds in directed and undirected graphs [12, 45].

1.4 Techniques

Lower Bounds. Our lower bounds use reductions from set disjointness, which has an unconditional communication lower bound. Set Disjointness is a two-party communication problem, where two players Alice and Bob are given k-bit strings S_a and S_b respectively. Alice and Bob need to communicate and decide if the sets represented by S_a and S_b are disjoint, i.e., whether there is no bit position i, $1 \leq i \leq k$, with $S_a[i] = 1$ and $S_b[i] = 1$. A classical result in communication complexity states that Alice and Bob must exchange $\Omega(k)$ bits even if they are allowed shared randomness [7, 35, 46]. Lower bounds using such a reduction also hold against randomized algorithms. To show inapproximability for arbitrarily large constant factors, we make use of reductions from problems with known CONGEST lower bounds: s-t connectivity and s-t undirected shortest path [49].

We use a reduction from set disjointness to establish a $\tilde{\Omega}(n)$ lower bound in directed weighted and unweighted graphs, for exact MWC and for $(2-\epsilon)$-approximation (Theorem 1.2.A). We establish $\tilde{\Omega}(\sqrt{n})$ lower bounds for *any* constant factor approximation algorithms for MWC in directed graphs by adapting a general lower bound graph used for MST, SSSP and other graph problems [17, 49] (Theorem 1.2.B). We also adapt these constructions to obtain a similar lower bound for undirected weighted graphs (Theorem 1.4.A,B). For undirected unweighted graphs, a lower bound of $\tilde{\Omega}(\sqrt{n})$ is known for $(2-\epsilon)$-approximation [23], and we obtain an $\tilde{\Omega}(n^{1/4})$ lower bound for *any* constant factor approximation (Theorem 1.3.A).

Due to space constraints, we defer our lower bound constructions and proofs to the full version [40].

Approximate MWC Upper Bounds. Our upper bounds use a framework of computing long cycles of high hop length and short cycles separately. Computing long cycles typically involves random sampling followed by computing shortest paths through sampled vertices. The sampling probability is chosen such that there is a sampled vertex on any long cycle with high probability, and we compute minimum weight cycles passing through sampled vertices.

Computing short cycles requires a variety of techniques for each of our algorithms, with our method for directed unweighted MWC being the most involved. In a directed unweighted graph, we define a specific neighborhood for each vertex v which contains a minimum weight cycle through v if the MWC does not pass through any sampled vertex, a method inspired by the sequential algorithm of [13]. In order to explore these neighborhoods efficiently, we perform a BFS computation with random scheduling from each vertex that is hop-restricted and restricted to the neighborhood. Additionally, to address congestion, we separately handle bottleneck vertices that send or receive a large number of messages.

The idea of bottleneck vertices has been used in [5, 29] to control congestion while communicating information. This technique is used in the context of computing exact weighted APSP, in a subroutine to send distance information from all vertices to a set of sinks (common to all vertices) through shortest path trees that have been implicitly computed. They identify bottleneck vertices through which too many messages are to be sent, and compute distances through the bottleneck vertex separately. These algorithms can afford to use a super-linear number of rounds as they involve expensive Bellman-Ford operations, and they identify bottleneck vertices one at a time. On the other hand in our algorithm, each vertex needs to send messages to a different set of vertices, i.e., the neighborhood containing short cycles. Another key difference is that our restricted BFS is performed on the fly simultaneously with identifying bottleneck vertices in a distributed manner, as we do not know the shortest path trees beforehand. Finally, distances through all bottleneck vertices are computed with a pipelined hop-restricted BFS to maintain our sublinear round bound.

For undirected unweighted graphs, we compute the $\sqrt{n}$ closest neighbors efficiently using pipelining and compute cycles contained within them. We prove that for any cycle that extends outside a $\sqrt{n}$-neighborhood, a 2-approximation of this cycle is computed with a BFS from sampled vertices.

For weighted graphs, both directed and undirected, we use hop-bounded versions of the unweighted algorithms to compute approximations of short cycles. We use a scaling technique from [41], where we construct a series of graphs with scaled weights such that distance-bounded shortest paths in the original graphs are approximated by some hop-bounded shortest path in a scaled graph.

1.5 Prior Work

Sequential Minimum Weight Cycle. The problem of computing MWC has been extensively studied in the sequential setting, for directed and undirected graphs, both weighted and unweighted. It can be solved by computing All Pairs Shortest Paths (APSP) in the given graph in $O(n^3)$ time and in $\tilde{O}(mn)$ time. The hardness of computing MWC in the fine-grained setting was shown by [50] for the n^3 class and MWC mn-hardness is the hypothesis used for establishing hardness for the mn class [3]. Fast approximation algorithms for computing MWC have been studied: 2-approximation of directed MWC can be computed in $\tilde{O}(\min(n^2, m\sqrt{n}))$ time [13] and 4-approximation can be computed in $\tilde{O}(mn^{1/3})$ time [26]. For undirected unweighted graphs, an α-approximation can be computed in $\tilde{O}(n^{1+1/\alpha})$ time [32]. For undirected weighted graphs, $\frac{4}{3}$-approximation can be computed in $\tilde{O}(n^2)$ time [47] and a general $\frac{4}{3}\alpha$-approximation can be computed in $\tilde{O}(n^{1+1/\alpha})$ time [33].

Distributed Minimum Weight Cycle. An $O(n)$ algorithm for computing girth was given in [28], and a $\tilde{\Omega}(\sqrt{n})$ lower bound for computing girth was given in [23] which applies to any $(2-\epsilon)$-approximation algorithm. An $\tilde{O}(\sqrt{ng} + D)$-round algorithm was given in [44] to compute $(2 - \frac{1}{g})$-approximation of girth(where g is the girth). Computing girth in low-treewidth graphs has been studied in [31]. For exact computation of girth, the gap between lower and upper bounds has been a longstanding open problem. The related problem of cycle detection has been studied with both

upper and lower bounds for undirected graphs [11, 15, 16, 21]. Tight bounds of $\tilde{\Theta}(n^{1/3})$ are known for triangle detection in undirected and directed graphs [12, 30, 45].

Other than reductions of exact MWC to APSP for directed and undirected graphs [3, 50], there are no prior results for computing MWC in directed graphs or weighted graphs, despite its significance in the sequential setting. Approximation algorithms and lower bounds for MWC also have not been studied except for girth.

CONGEST results for APSP and related problems. The CONGEST round complexity of APSP [41] has been studied extensively, with near-optimal upper and lower bounds of $\tilde{O}(n)$ [8] and $\Omega(\frac{n}{\log n})$ [41] respectively. Upper and lower bounds for some related problems that have sequential $O(n^3)$ and $O(mn)$ algorithms have been studied in the CONGEST model, such as for diameter [1, 6], replacement paths and second simple shortest paths [39], radius and eccentricities [1, 6], and betweenness centrality [27].

The round complexity of both exact and approximate SSSP has been extensively researched [9, 10, 14, 17, 20, 41]. For exact or $(1+\epsilon)$-approximate SSSP, the current best upper and lower bounds are $\tilde{O}(n^{2/5+o(1)}D^{2/5}+\sqrt{n}+D)$ [9] and $\Omega(\sqrt{n}+D)$ [17, 49] respectively. Multiple source SSSP has been studied in [18, 19], with an algorithm taking $\tilde{O}(\sqrt{nk}+D)$ for approximate k-source SSSP in undirected graphs in [18].

1.6 Roadmap

We start by presenting algorithms for computing k-source directed BFS and approximate SSSP in Section 2, which are used as subroutines in our MWC algorithms. Our main algorithmic result for 2-approximation of directed unweighted MWC in $\tilde{O}(n^{4/5}+D)$ rounds is in Section 3. For undirected unweighted graphs, we present a near-optimal algorithm for $(2-\frac{1}{g})$-approximation of undirected unweighted MWC in $\tilde{O}(\sqrt{n}+D)$ rounds in Section 4 (here g is the length of MWC). We compute $(2+\epsilon)$-approximation of weighted MWC in Section 5, taking $\tilde{O}(n^{2/3}+D)$ rounds for undirected and $\tilde{O}(n^{4/5}+D)$ for directed graphs. We conclude with some avenues for further research in Section 6.

2 k-SSSP FROM $k \geq n^{1/3}$ SOURCES

We present algorithms to compute directed k-source BFS in unweighted graphs and k-source SSSP in weighted graphs. For k-source directed unweighted BFS our algorithm uses techniques of sampling and constructing a skeleton graph on sampled vertices, which are methods used in CONGEST single source reachability and SSSP algorithms [25, 41]. For k-source approximate directed SSSP, we utilize a recent directed hopset construction from [10] in conjunction with some techniques used in [18] for computing approximate k-source SSSP in undirected graphs.

We start by presenting Algorithm 1 that computes an $n^{1/3}$-source *exact* directed BFS in $\tilde{O}(n^{2/3}+D)$ rounds, and later generalize our result to $k \geq n^{1/3}$ sources. We then extend our algorithm to weighted graphs to compute k-source $(1+\epsilon)$-approximate SSSP.

Let $U \subseteq V$ be the set of sources. Algorithm 1 first randomly samples a vertex set $S \subseteq V$ of size $\tilde{\Theta}(n^{1/3})$ in line 1. We define a (virtual) skeleton graph on this vertex set S, where for vertices $u, v \in S$, an edge (u, v) is added iff there is a directed path from u to

Algorithm 1 Exact $n^{1/3}$-source Directed BFS algorithm

Input: Directed unweighted graph $G = (V, E)$, set of sources $U \subseteq V$ with $|U| = k = n^{1/3}$.

Output: Every vertex v computes $d(u, v)$ for each source $u \in U$.

1: Let $h = n^{2/3}$. Construct set $S \subseteq V$ by sampling each vertex $v \in V$ with probability $\Theta(\frac{\log n}{h})$. W.h.p. in n, $|S| = \tilde{\Theta}(n^{1/3})$.
2: Compute h-hop directed BFS from each vertex in S. Repeat this computation in the reversed graph. This takes $O(|S|+h) = \tilde{O}(n^{2/3})$ rounds. ▷ *Computes shortest path distances of $\leq h$ hops.*
3: ▷ *The following lines compute ($> h$)-hop shortest path distances.* ◁
4: Construct a skeleton graph on vertex set S: For each directed h-hop shortest path in the underlying graph G between sampled vertices found in line 2, add a directed edge with weight equal to shortest path distance. ▷ *Internal computation*
5: Share all edges of the skeleton graph by broadcast, node v broadcasts all its outgoing edges. We broadcast up to $|S|^2$ values in total, which takes $O(|S|^2+D) = \tilde{O}(n^{2/3}+D)$ rounds.
6: Each sampled vertex internally computes all pairs shortest paths in the skeleton graph using the broadcast values.
7: Perform h-hop directed BFS from each source $u \in U$, in $O(h+k) = \tilde{O}(n^{2/3})$ rounds. If any sampled vertex $s \in S$ is visited during this BFS, s broadcasts distance $d(u, s)$. We broadcast up to $k \cdot |S| = \tilde{\Theta}(n^{2/3})$ values, taking $\tilde{O}(n^{2/3}+D)$ rounds.
8: Using the broadcast information, sampled vertices determine their shortest path distance to sources in U: if distance $d(u, t)$ was broadcast for some $u \in U, t \in S$, each sampled vertex $s \in S$ locally sets $d(u, s) \leftarrow \min(d(u, s), d(u, t)+d(t, s))$.
9: Each sampled vertex $s \in S$ propagates distance $d(u, s)$ for each $u \in U$ through h-hop BFS trees computed in line 2. Using random scheduling [24], this takes $\tilde{O}(h+k|S|) = \tilde{O}(n^{2/3})$ rounds.
10: Each vertex v receives distance $d(u, s)$ for source u from a sampled vertex s that contains v in its h-hop BFS tree, and v computes $d(u, v) \leftarrow \min_{s \in S}(d(u, s)+d(s, v))$.

v of at most $h = n^{2/3}$ hops in G. The skeleton graph is directed and weighted, with the weight of each skeleton graph edge being the h-hop bounded shortest path distance in G. The skeleton graph edges are determined using an h-hop directed BFS from each sampled vertex in line 2, and each sampled vertex uses this information to internally determine its outgoing edges in line 4. These skeleton graph edges are then broadcast to all vertices in line 5. All pairs shortest path distances in the skeleton graph can be computed locally at each vertex in line 6 using these broadcast distances, due to the chosen sampling probability.

In line 7, an h-hop BFS is performed from each source, and each vertex that is at most h hops from a source can compute its distance from that source. We now compute distances from each source to sampled vertices (regardless of hop-length) in line 8 using the h-hop bounded distances from line 7 along with skeleton graph distances from line 2. Finally distances from each source to all vertices are computed in line 9, by propagating the distances computed in line 8 through the h-hop BFS trees rooted at each sampled vertex. This allows all vertices to locally compute their distance from all sources.

Computing h-hop BFS from k sources takes $O(h+k)$ rounds [37], and broadcasting $|S|^2 = O(n^{2/3})$ values in line 5 takes $\tilde{O}(n^{2/3}+D)$

rounds. Line 9 involves $k|S| = \tilde{O}(n^{2/3})$ values propagated through $(h = n^{2/3})$-hop BFS trees, which takes $\tilde{O}(n^{2/3})$ rounds using random scheduling [24, 36]. So, the total round complexity of Algorithm 1 is $\tilde{O}(n^{2/3} + D)$ rounds. Detailed proofs are in the full version [40].

Algorithm 1 for $k = n^{1/3}$ sources can be readily generalized to $k \geq n^{1/3}$ sources to obtain an algorithm that takes $\tilde{O}(\sqrt{nk} + D)$ rounds by using parameter $h = \sqrt{nk}$, proving result (1) in Theorem 1.6.A.

Weighted Graphs. We extend our results to weighted graphs to compute k-source $(1+\epsilon)$-approximate SSSP. We replace the h-hop directed BFS computations in Algorithm 1 with h-hop approximate SSSP algorithm from [41] that takes $\tilde{O}(h+k)$ rounds for k sources (which only increases rounds by a polylog factor). Thus, we obtain an algorithm to compute $(1+\epsilon)$-approximate SSSP from $k \geq n^{1/3}$ sources in $\tilde{O}(\sqrt{nk}+D)$ rounds in directed and undirected weighted graphs, proving result (2) in Theorem 1.6.B.

Multiple Source SSSP from $k < n^{1/3}$ sources. In directed unweighted graphs for $k < n^{1/3}$, Algorithm 1 computes exact k-source BFS in $\tilde{O}(\frac{n}{k}+D)$ rounds by choosing parameter $h = \sqrt{nk}$. For small k, the simple algorithm of repeating SSSP computation in sequence from each source taking $k \cdot SSSP$ rounds could be more efficient (threshold for k depends on value of D), and this gives the result in Theorem 1.6.A. See the full version [40] for more details.

In the full version [40], we present an algorithm for approximate SSSP and BFS from $k < n^{1/3}$ sources with round complexity $\tilde{O}(\sqrt{nk} + k^{2/5}n^{2/5+o(1)}D^{2/5} + D)$, proving Theorem 1.6.B. Our algorithm improves on the simple method of repeating the current best (approximate) SSSP algorithm [10] k times for the entire range of $1 < k \leq n$.

3 APPROXIMATE DIRECTED MWC

We present a CONGEST algorithm for 2-approximation of directed unweighted MWC in Algorithm 2. Our algorithm uses sampling combined with multiple source exact directed BFS (result (1) of Theorem 1.6.A) to exactly compute the weight of MWC among long cycles of hop length $\geq h = n^{3/5}$. We handle the case when MWC is short with hop length $< h$ in Section 3.1.

In line 2 of Algorithm 2, we sample $\tilde{\Theta}(n^{2/5})$ vertices uniformly at random and in line 3, we perform a directed BFS computation from each of them in $\tilde{O}(n^{7/10} + D)$ rounds (Theorem 1.6.A). Using these computed distances, each sampled vertex locally computes a minimum weight cycle through itself in line 4, thus computing MWC weight among long cycles. We use Algorithm 3 (see Section 3.1) in line 6 to handle short cycles. Computing short MWC requires the distances between all pairs of sampled vertices as input: so in line 5 of Algorithm 2 we broadcast the h-hop shortest path distances between sampled vertices found during the BFS of line 3 and use these distances to locally compute shortest paths between all pairs of sampled vertices at each vertex. Thus, we exactly compute MWC weight if a minimum weight cycle passes through a sampled vertex in line 4, and a 2-approximation of MWC weight in line 6 otherwise. We now address the short cycle subroutine used in line 6.

Algorithm 2 2-Approximation Algorithm for Directed Unweighted MWC

Input: Directed unweighted graph $G = (V, E)$
Output: μ, 2-approximation of weight of a MWC in G

1: Let $h = n^{3/5}$. Set $\mu_v \leftarrow \infty$ for all $v \in V$. ▷ *μ_v will track the minimum weight cycle through v found so far.*
2: Construct set S by sampling each vertex $v \in G$ with probability $\Theta(\frac{1}{h} \cdot \log^3 n)$. W.h.p. in n, $|S| = \Theta(n^{2/5} \cdot \log^2 n)$.
3: Compute $d(s, v)$ for $s \in S, v \in V$ using multiple source exact directed BFS (Theorem 1.6.A, Algorithm 1) from S. This takes $\tilde{O}(\sqrt{n|S|} + D)$ rounds as $|S| > n^{1/3}$.
4: Compute cycles through $s \in S$: For each edge (v, s), $\mu_v \leftarrow min(\mu_v, w(v, s) + d(s, v))$. ▷ *Locally compute lengths of long cycles and all cycles passing through some sampled vertex.*
5: Broadcast all pairs distances between sampled vertices: Each $t \in S$ broadcasts $d(s, t)$ for all $s \in S$. There are at most $|S|^2$ such distances, which takes $O(|S|^2 + D)$ rounds.
6: Run Algorithm 3 to compute approximate short MWC if it does not contain a sampled vertex, updating μ_v for each $v \in V$. ▷ *See Section 3.1.*
7: Return $\mu \leftarrow \min_{v \in V} \mu_v$, computed by a convergecast operation [43] in $O(D)$ rounds.

3.1 Computing Approximate Short MWC

We present a method to compute 2-approximation of weight of minimum weight cycle among cycles of at most $h = n^{3/5}$ hops that do not pass through any sampled vertex in S. Our method is detailed in Algorithm 3 and runs in $\tilde{O}(n^{4/5})$ rounds. As mentioned in the previous section, each vertex v knows the distances $d(v, s), d(s, v)$ for each vertex $s \in S$, and distances $d(s, t)$ for all pairs $s, t \in S$.

Description of $P(v)$ and $R(v)$. For each vertex $v \in V$, we define a neighborhood $P(v) \subseteq V$ such that $P(v)$ contains (w.h.p. in n) a cycle whose length is at most a 2-approximation of a minimum weight cycle C through v if C does not pass through any sampled vertex. The construction of $P(v)$ is inspired by a sequential algorithm for directed MWC in [13], which uses the following lemma.

Fact 1 (Lemma 5.1 of [13]). Let C be a minimum weight cycle that goes through vertices v, y in a directed weighted graph G. For any vertex t, if $d(y, t) + 2d(v, y) \geq d(t, y) + 2d(v, t)$, then a minimum weight cycle containing t and v has weight at most $2w(C)$.

Suppose we determine that $d(y, t) + 2d(v, y) \geq d(t, y) + 2d(v, t)$ for a vertex $y \in V$ and some sampled vertex $t \in S$. Then by Fact 1 the minimum weight cycle through v and t is at most twice the minimum weight cycle through v and y. Since we compute MWC through all sampled vertices, we can exclude y from $P(v)$. We choose a subset $R(v) \subseteq S$ and use only $t \in R(v)$ to eliminate vertices from $P(v)$ using Fact 1. In particular, we will construct a set $R(v) \subseteq S$ of size $\log n$ such that the size of $P(v)$ is reduced to at most $\frac{n}{|S|} = \tilde{O}(n^{3/5})$ (as we show later).

Definition 3.1. Given a subset $R(v) \subseteq S$, define

$$P(v) = \{y \in V \mid \forall t \in R(v), d(y, t) + 2d(v, y) \leq d(t, y) + 2d(v, t)\}$$

Our algorithm computes a directed BFS from each vertex v restricted to $P(v)$. For the BFS to reach all of $P(v)$, the graph induced by $P(v)$ must be connected, which we prove below in Lemma 3.2.

Lemma 3.2. *$P(v)$ induces a connected subgraph in the shortest path out-tree rooted at v.*

Proof. Let vertex $y \in P(v)$. Then, we will prove that for any vertex z on a shortest path from v to y, $z \in P(v)$ thus proving our claim.

We will use the fact that $d(v,y) = d(v,z) + d(z,y)$ by our assumption. Assume $y \in P(v)$, that is $\forall t \in R(v), d(y,t) + 2d(v,y) \leq d(t,y) + 2d(v,t)$. Fix any $t \in R(v)$, we need to prove that $d(z,t) + 2d(v,z) \leq d(t,z) + 2d(v,t)$. We will use the triangle inequalities $d(z,t) \leq d(z,y) + d(y,t)$ and $d(t,y) \leq d(t,z) + d(z,y) \implies d(t,y) - d(z,y) \leq d(t,z)$.

$$\begin{aligned} d(z,t) + 2d(v,z) &\leq d(z,y) + d(y,t) + 2d(v,z) \\ &= d(y,t) + (2d(v,z) + 2d(z,y)) - d(z,y) \\ &= d(y,t) + 2d(v,y) - d(z,y) \\ &\leq d(t,y) + 2d(v,t) - d(z,y) \ \ (\text{since } y \in P(v)) \\ &= (d(t,y) - d(z,y)) + 2d(v,t) \\ &\leq d(t,z) + 2d(v,t) \end{aligned}$$

□

To construct the set $R(v)$ at vertex v, Algorithm 3 partitions the sampled vertices into $\beta = \log n$ sets $S_1, \ldots S_\beta$ in line 2. In lines 3-8 of Algorithm 3, we construct $R(v)$ iteratively by adding at most one vertex from each S_i, so that $R(v)$ has size $\leq \log n$. In the i'th iteration, we identify the vertices in S_i that have not been eliminated from $P(v)$ by any of the $(i-1)$ previously chosen vertices in $R(v)$ and choose one of these vertices at random to add to $R(v)$. This entire computation is done locally at v using distances between all pairs of sampled vertices sent to v in line 5 of Algorithm 2.

Restricted BFS. In lines 13-22 of Algorithm 3, we compute h-hop BFS from all vertices v restricted to neighborhood $P(v)$: the BFS proceeds for h steps, and at each step the BFS message is forwarded only to neighbors in $P(v)$. To test membership in $P(v)$ using Definition 3.1 at an intermediate vertex before propagating, we use distances between sampled vertices that are part of the input along with information about $R(v)$ that is included in the BFS message. Note that the BFS message has size $O(\log n)$ since $R(v)$ has size $\log n$ (see line 16).

The restricted BFS from every vertex needs to be carefully scheduled to obtain our sublinear round bound. We first implement random delays using ideas in [24, 36], where the start of BFS for a source v is delayed by an offset δ_v chosen uniformly from range $[1, \rho = n^{4/5}]$ at random by v. Here, the parameter $\rho = n^{4/5}$ is chosen based on the maximum number of messages allowed throughout the BFS for a single vertex, as we shall see later. With this scheduling, all BFS messages for a particular source v are synchronous even though messages from different sources may not be. We organize the BFS into phases, each phase involving at most $\Theta(\log n)$ messages. However, the graph may contain bottleneck vertices u that are in the neighborhood $P(v)$ for many v: such a vertex u has to process up to n messages, requiring $\Omega(n)$ phases even with random scheduling.

We define u to be a *phase-overflow vertex* if it has to send or receive more than $\Theta(\log n)$ messages in a single phase of the restricted BFS. During the BFS, we identify any such phase-overflow vertex and terminate BFS computation through it. The BFS runs for $O(n^{4/5})$ phases and each BFS message contains $O(\log n)$ words, and hence the BFS takes a total of $\tilde{O}(n^{4/5})$ rounds. After this restricted BFS is completed, each vertex u knows its shortest path distance from all vertices v such that $u \in P(v)$ and there is a v-u shortest path that has at most h hops and contains no phase-overflow vertex.

Now, it remains to compute h-hop shortest path distances for paths that contain phase-overflow vertices. We prove a bound of $\tilde{O}(n^{4/5})$ on the number of phase-overflow vertices as follows. We define u to be a *bottleneck vertex* if $u \in P(v)$ for more than $\rho = n^{4/5}$ vertices $v \in V$, i.e., u may have to handle messages for more than $n^{4/5}$ sources across all phases of the BFS. We prove the following claims in Lemma 3.3: (i) Vertex u can be a phase-overflow vertex only if u is a bottleneck vertex, and (ii) the number of bottleneck vertices is at most $\tilde{O}(n^{4/5})$. The bound on the number of bottleneck vertices is obtained using the bound of $\tilde{O}(n^{3/5})$ on the size of each $P(v)$. We compute h-hop directed BFS from the $\tilde{O}(n^{4/5})$ phase-overflow vertices in $O(h + n^{4/5})$ rounds.

After computing all distances from each $v \in V$ to vertices y in $P(v)$, we locally compute the minimum among discovered cycles through v: at vertex v, a discovered cycle is formed by concatenating a v-y shortest path and an incoming edge (v,y).

We now prove some results in order to argue correctness. We first argue that $P(v)$ has size at most $\frac{n}{|S|} = \tilde{\Theta}(n^{3/5})$ w.h.p in n (adapting Lemma 6.2 of [13]). When we add a vertex t to $R(v)$ in line 8, we expect t to cover cycles through half the remaining uncovered vertices, since the condition we check (as in Definition 3.1) is symmetric. At any iteration i of lines 7-8, if the number of uncovered vertices is larger than $\tilde{\Theta}(n^{3/5})$, then w.h.p. in n there is some vertex in S_i (which has size $\Theta(n^{2/5} \log n)$) that is not covered. This vertex is then added to $R(v)$, reducing the remaining number of uncovered vertices by half. So, the probability that the number of uncovered vertices $P(v)$ remains larger than $\tilde{\Theta}(n^{3/5})$ after $\log n$ such iterations is polynomially small.

Lemma 3.3. *(i) Vertex $u \in V$ is a phase-overflow vertex only if u is a bottleneck vertex.*
(ii) There are at most $\tilde{O}(n^{4/5})$ bottleneck vertices w.h.p. in n.
(iii) There are at most $\tilde{O}(n^{4/5})$ phase-overflow vertices w.h.p. in n.

Proof. We define $P^{-1}(u) = \{v \in V \mid u \in P(v)\}$ to be the set of vertices for which u is a part of their neighborhood. By definition, u is a bottleneck vertex if $|P^{-1}(u)| \geq \rho = n^{4/5}$.
Proof of (i): A message is sent to u from source v by some neighbor x only if $u \in P(v)$, so $P^{-1}(u)$ is the set of sources for which u has to send and receive BFS messages. Assume that u is not a bottleneck vertex, $|P^{-1}(u)| < \rho$, we will prove that u is not a phase-overflow vertex w.h.p. in n, i.e., u sends or receives at most $\Theta(\log n)$ messages in a single phase of BFS.

By assumption, u receives messages from at most $|P^{-1}(u)| < \rho$ sources throughout all phases of the restricted BFS. Additionally, each incoming edge to u receives at most ρ messages throughout the BFS since a single BFS sends at most one message through a single edge. Fix one such edge, that receives messages from sources

Algorithm 3 Approximate Short Cycle Subroutine

Input: Directed unweighted graph $G = (V, E)$, set of sampled vertices $S \subseteq V$. Each vertex v knows distances $d(v, s), d(s, v)$ for $s \in S$ and distances $d(s, t)$ for $s, t \in S$.

Output: For each v, return μ_v which is a 2-approximation of minimum weight of cycles through v among cycles that are short($< n^{3/5}$ hops) and do not pass through any vertex in S.

1: $h = n^{3/5}, \rho = n^{4/5}$.
2: Partition S into $\beta = \log n$ sets $S_1, \ldots S_\beta$ of size $\Theta(n^{2/5} \cdot \log n)$.
3: **for each** vertex $v \in G$ **do**
4: ▷ ***Initial Setup***: *Compute set $R(v) \subseteq S$, which is used to restrict BFS to neighborhood $P(v) \subseteq V$ (defined in Section 3.1).* ◁
5: $R(v) \leftarrow \phi$
6: **for** $i = 1 \ldots \beta$ **do** ▷ *Local computation at v.*
7: Let $T(v) = \{s \in S_i \mid \forall t \in R(v), d(s, t) + 2d(v, s) \leq d(t, s) + 2d(v, t)\}$.
8: If $T(v)$ is not empty, select a random vertex $s^* \in T(v)$ and add it to $R(v)$.
9: δ_v is chosen uniformly at random from $\{1, \ldots, \rho\}$. ▷ *Choose BFS delay.*
10: $Z(v) \leftarrow 0$ ▷ *$Z(v)$ is a flag that determines whether v is a phase-overflow vertex.*
11: Send $\{(d(v, s), d(s, v)) \mid s \in S\}$ to each neighbor u in $O(|S|)$ rounds.
12: ▷ ***Restricted BFS from all vertices***: *Computation is organized into phases where each vertex receives and sends at most $\log n$ BFS messages along its edges. Each BFS message contains $O(\log n)$ words and hence each phase takes $O(\log^2 n)$ CONGEST rounds.* ◁
13: **for** phase $r = 1 \ldots (h + \rho)$ **do**
14: **for each** vertex $v \in G$ **do**
15: **if** $r = \delta_v$ **then** ▷ *This is the first phase for the BFS rooted at v.*
16: Construct message $Q(v) = (R(v), \{d(v, t) \mid \forall t \in R(v)\})$ to be sent along the BFS rooted at v. $Q(v)$ contains $O(\log n)$ words ($|R(v)| \leq \beta = \log n$) and can be sent in $O(\log n)$ rounds.
17: Send BFS message $(Q(v), d(v, v) = 0)$ to each out-neighbor of v.
18: ▷ *Process and propagate messages from other sources y. We restrict the number of messages sent/received by a vertex by $\Theta(\log n)$ and identify phase-overflow vertices($Z(v) \leftarrow 1$) exceeding this congestion. Phase-overflow vertices are processed separately in line 24.* ◁
19: Receive at most $\log n$ messages $(Q(y), d^*(y, v))$ from each in-neighbor. If more than $\Theta(\log n)$ messages are received from an edge, set $Z(v) \leftarrow 1$ and terminate.
20: If message $(Q(y), d^*(y, v))$ is not the first message received for source y, discard it. Let $Y^r(v)$ denote the remaining set of sources y with first time messages, and set $d(y, v) \leftarrow d^*(y, v)$ for $y \in Y^r(v)$.
21: If $|Y^r(v)| > \Theta(\log n)$, set $Z(v) \leftarrow 1$ and terminate.
22: For each $y \in Y^r(v)$, and for each outgoing neighbor u, set estimate $d^*(y, u) \leftarrow d(y, v) + 1$. If $\forall t \in R(y), d(u, t) + 2d^*(y, u) \leq d(t, u) + 2d(y, t)$, send message $(Q(y), d^*(y, u))$ to u. ▷ *Note that $R(y), d(y, t)$ are known to v from $Q(y)$ and $d(u, t), d(t, u)$ from line 11.*
23: ▷ *Process phase-overflow vertices.* ◁
24: Let $Z = \{v \in V \mid Z(v) = 1\}$. Perform directed h-hop BFS with sources Z in $O(|Z| + h)$ rounds. For each $v \in Z$ and edge (x, v), set $\mu_x \leftarrow min(\mu_x, d(v, x) + w(x, v))$.
25: **for** vertex $v \in V$ **do**
26: $\mu_v \leftarrow \min(\mu_v, d(v, y) + 1)$, for each $y \in V$ such that edge (y, v) exists and $d(v, y)$ was computed during this algorithm.

$v_1, v_2, \ldots v_\gamma$ for $\gamma < \rho$, and let the distance from v_i to u be h_i. The BFS messages from source v are offset by a random delay $\delta_v \in \{1, 2, \ldots \rho\}$ and thus the BFS message from v_i is received at u at phase $h_i + \delta_{v_i}$. For a fixed phase r, the message from v_i is sent to u at phase r iff $r = h_i + \delta_{v_i}$ which happens with probability $\frac{1}{\rho}$ since δ_{v_i} is chosen uniformly at random. Using a Chernoff bound, we can show that w.h.p. in n, there are at most $\Theta(\log n)$ of the γ messages that are sent at phase r through the chosen edge.

Vertex u sends an outgoing message for the BFS rooted at v only if $u \in P(v)$ and it received a message from source v. So, u sends at most $|P^{-1}(u)| < \rho$ outgoing messages through a single outgoing edge, and we can repeat the same argument above to argue that at most $\Theta(\log n)$ messages are sent out at a single phase. So, u is not a phase-overflow vertex.

Proof of (ii): We use the fact that w.h.p. in n, for each $v \in V$, $|P(v)| \leq n^{3/5}$. By definition of $P^{-1}(u)$, we have $\sum_{u \in V} |P^{-1}(u)| = \sum_{v \in V} |P(v)|$ (counting pairs of vertices $v, u \in P(v)$). Using the bound $|P(v)| \leq \tilde{O}(n^{3/5})$, we get $\sum_{u \in V} |P^{-1}(u)| \leq n \cdot \tilde{O}(n^{3/5})$.

Let B denote the set of bottleneck vertices. Then, $\sum_{u \in V} |P^{-1}(u)| \geq \sum_{u \in B} |P^{-1}(u)| \geq |B| \cdot \rho$ and hence $|B| \leq (n/\rho) \cdot \tilde{O}(n^{3/5}) = \tilde{O}(n^{4/5})$.

Proof of (iii): By *(i)*, the number of phase-overflow vertices is $\leq |B|$ and $|B| \leq \tilde{O}(n^{4/5})$ by *(ii)*. □

We now present details of the proof of correctness and round complexity of Algorithm 2.

Lemma 3.4. *Algorithm 2 correctly computes a 2-approximation of MWC weight in a given directed unweighted graph $G = (V, E)$ in $\tilde{O}(n^{4/5} + D)$ rounds.*

Proof. **Correctness:** Whenever we update μ_v for any $v \in V$, we use a shortest path from x to v along with an edge (v, x), which means we only record weights of valid directed cycles. Let C be a MWC of G with weight $w(C)$ and let v refer to an arbitrary vertex on C. In the following cases for C, Cases 1 and 2 are handled in Algorithm 2, and Cases 3, 4 are handled by the subroutine in line 6 using Algorithm 3.

Case 1: $w(C) \geq h$: C contains at least h vertices, and hence w.h.p. in n, C contains a sampled vertex in S by our choice of sampling probability. If $s \in S$ is on C, then the computation in line 4 exactly computes $w(C)$.

Case 2: $w(C) < h$ and C extends outside $P(v)$: Let u be a vertex on C such that $u \notin P(v)$, then we have $d(u, t) + 2d(v, u) > d(t, u) + 2d(v, t)$ for some $t \in R(v)$. By Fact 1, this means that a minimum weight cycle containing t and v has weight at most $2w(C)$ since C is a minimum weight cycle containing v and u. Since $R(v) \subseteq S$, t is a sampled vertex and hence $\mu_t \leq 2w(C)$ by the computation in line 4. Thus, we compute a 2-approximation of the weight of C.

Case 3: $w(C) < h$, C is contained in $P(v)$, $\exists u \in C, Z(u) = 1$: In this case, u is in the set of phase-overflow vertices Z constructed in line 24 of Algorithm 3. After the BFS computation through vertices in Z, a minimum weight cycle through u is computed in line 24. Thus, $w(C)$ is computed exactly.

Case 4: $w(C) < h$, C is contained in $P(v)$, $\forall u \in C, Z(u) = 0$: In Algorithm 3, u is not a phase-overflow vertex as $Z(u) = 0$ and u never terminates its execution in line 19. In fact, none of the vertices on C terminate their execution and forward messages from all sources, including v. Thus, the vertex z furthest from v receives message $d(v, z)$ and records a cycle of weight $w(C)$ in line 26.

Round complexity: We first address the running time of Algorithm 2 apart from line 6 which invokes the subroutine Algorithm 3.

We choose our sampling probability such that $|S| = \tilde{\Theta}(n/h) = \tilde{\Theta}(n^{2/5})$, so the multiple source SSSP in line 3 of Algorithm 2 takes time $\tilde{O}(\sqrt{n|S|} + D) = \tilde{O}(n^{7/10} + D)$ using Theorem 1.6.A since we have $\tilde{\Theta}(n^{2/5}) > n^{1/3}$ sources. In line 5, we broadcast $|S|^2$ values taking $O(|S|^2 + D) = \tilde{O}(n^{4/5} + D)$ rounds. Line 7 involves a convergecast operation among all vertices, which takes $O(D)$ rounds [43].

Round complexity of Algorithm 3: We now show that Algorithm 3 takes $\tilde{O}(n^{4/5})$ rounds.

The computation in lines 1-10 is done locally at each vertex v. The local computation of $R(v)$ (lines 3-8) only uses distances $d(v, t)$ and distances $d(s, t)$ for $s, t \in S$ that are part of the input. In line 11, vertex v sends $O(|S|)$ words of information to each neighbor, which takes $O(|S|) = \tilde{O}(n^{2/5})$ rounds.

We now address the round complexity of the restricted BFS of Lines 13-22. The restricted BFS computation is organized into $(h + \rho)$ phases (recall $h = n^{3/5}, \rho = n^{4/5}$). Each phase runs for $O(\log^2 n)$ rounds in which each vertex receives and sends up to $\Theta(\log n)$ BFS messages. Each message of the BFS is of the form $(Q(v), d(v, w))$ as in line 16. Since $Q(v)$ has at most $\beta = \log n$ words, the BFS message can be sent across an edge in $O(\log n)$ rounds. The round bound for each phase is enforced in lines 19,21 where propagation through a vertex is terminated if it has to send or receive more than $\Theta(\log n)$ messages in a single round, i.e., it is a phase-overflow vertex. The membership test in line 22 is done using distances known to v along with information from the BFS message, without additional communication. Thus, lines 13-22 take a total of $O\left((h + \rho) \cdot \log^2 n\right) = \tilde{O}(n^{4/5})$ rounds.

We bound the round complexity of line 24 using Lemma 3.3 to bound the number of phase-overflow vertices by $\tilde{O}(n^{4/5})$, i.e., $|Z| \leq \tilde{O}(n^{4/5})$. Now, the h-hop directed BFS in line 24 from $|Z|$ sources takes $O(|Z| + h) = \tilde{O}(n^{4/5})$ rounds [37]. Finally in line 26, after all BFS computations are completed, we locally compute the minimum discovered cycle through each vertex v formed by a w-v shortest path along with edge (v, w). □

4 UNDIRECTED UNWEIGHTED MWC

In this section, we present an algorithm for computing $(2 - \frac{1}{g})$-approximation of girth (undirected unweighted MWC) in $\tilde{O}(\sqrt{n}+D)$ rounds, where g is the girth. We outline our method here, and present pseudocode in the full version [40].

We first sample a set of $\tilde{O}(\sqrt{n})$ vertices and perform a BFS with each sampled vertex as source. For each non-tree edge (x, y) in T, the BFS tree from sampled vertex w, we record a candidate cycle of weight $d(w, x) + d(w, y) + 1$. Note that this may overestimate the size of the simple cycle C created by edge (x, y) and paths in T from x and y by at most $2d(w, v)$ where v is the closest vertex to w on C (i.e., v is $lca(x, y)$ in T). For cycles where $d(w, v)$ is small relative to the size of C, we get a good approximation of the weight of the cycle. We prove that w.h.p. in n, the only cycles for which this method fails to give a good approximation for any source w are cycles entirely contained within the $\sqrt{n}$ neighborhood of each vertex in the cycle. We efficiently compute shortest path distances within each neighborhood using a source detection algorithm [37], and then compute MWC within the neighborhood.

If a minimum weight cycle extends outside the $\sqrt{n}$ neighborhood of even one of the vertices in the cycle, we show that a sampled vertex w exists in this neighborhood. Thus, when we compute distances from each sampled vertex, we compute a 2-approximation of the weight of such a cycle. We use a more precise approach to obtain our $(2 - \frac{1}{g})$-approximation, by computing lengths of cycles such that exactly one vertex is outside the neighborhood. The source detection procedure for $\sqrt{n}$-neighborhood takes $O(\sqrt{n} + D)$ rounds [37] and BFS from $\tilde{O}(\sqrt{n})$ sampled vertices takes $\tilde{O}(\sqrt{n}+D)$ rounds [37] giving us our total round complexity of $\tilde{O}(\sqrt{n} + D)$.

Computing h-hop limited MWC. (used in Section 5.1). If we are only required to compute approximate h-hop limited MWC, i.e. compute 2-approximation of minimum weight among cycles of $\leq h$ hops, we can restrict our BFS computations to h hops to obtain an $\tilde{O}(\sqrt{n} + h + D)$ round algorithm. This does not improve the running time for unweighted graphs as $h \leq D$, but in Section 5.1 we will apply this procedure to weighted graphs using the following notion of stretched graph: given a network $G = (V, E)$ with weights on edges, a *stretched* unweighted graph G^s is obtained by mapping each edge of G with weight w to a unweighted path of w edges. If G is directed, the path is directed as well.

Given edge-weighted network $G = (V, E)$, we can efficiently simulate the corresponding stretched graph G^s on the network by simulating all but the last edge of the path corresponding to a weighted edge at one of the endpoints. The diameter of the stretched graph may be much larger than that of G but convergecast operations cost only $R_{cast} = O(D)$ rounds where D is the undirected diameter of G. Thus, we can compute h-hop limited unweighted MWC in G^s in $\tilde{O}(\sqrt{n} + h + R_{cast})$ rounds. Note that a cycle of hop length h in G^s corresponds to a cycle of weight h in G. We use this idea in the next section with scaled-down weights so that even a cycle of large weight in G can be approximated by a cycle of low hops in an appropriate stretched graph. See the full version [40] for details.

Corollary 4.1. *Given a network $G = (V, E)$ with edge weights, we can compute a $(2 - 1/g)$-approximation of h-hop limited MWC of G^s (stretched unweighted graph of G) in $\tilde{O}(\sqrt{n} + h + R_{cast})$ rounds, where g is the h-hop limited MWC value in G_s and R_{cast} is the round complexity of convergecast.*

5 WEIGHTED MWC

In this section, we present algorithms to compute $(2+\epsilon)$-approximate weighted MWC in $\tilde{O}(n^{2/3} + D)$ rounds for undirected graphs and

$\tilde{O}(n^{4/5} + D)$ rounds for directed graphs. Our algorithms use the k-source approximate SSSP algorithm from Section 2 along with unweighted MWC approximation algorithms of Sections 3 and 4 on scaled graphs to approximate weighted MWC.

5.1 Approximate Undirected Weighted MWC

We sketch our method to compute $(2+\epsilon)$-approximation in $\tilde{O}(n^{2/3}+D)$ rounds, proving Theorem 1.4.C. Details are in the full version [40].

Let $h = n^{2/3}$. For long cycles with $\geq h$ hops, we sample $k = \tilde{\Theta}(n^{1/3})$ vertices so that w.h.p. in n there is at least one sampled vertex on the cycle. We compute $(1+\epsilon)$-approximate k-SSSP from sampled vertices (result (2) of Theorem 1.6.B) and use this to obtain $(1+\epsilon)$-approximate MWC among long cycles.

For small cycles (hop length $< h$), we use scaling along with a hop-limited version of 2-approximate undirected unweighted MWC algorithm from Corollary 4.1 of Section 4. We use the scaling technique in [41], where it was used in the context of computing approximate shortest paths. We construct $O(\log n)$ scaled versions of the graph G, denoted G^i for $1 \leq i \leq \log(hW)$ with edge-weight w scaled to have weight $\lceil \frac{2hw}{\epsilon 2^i} \rceil$. Each h-hop limited shortest path P in G is approximated by a path of weight at most $h^* = ((1 + (2/\epsilon)) \cdot h)$ in some G^{i^*} — this i^* is in fact $\lceil \log w(P) \rceil$ where $w(P)$ is the weight of P in G, as proven in [41].

We run an h^*-hop limited version of the unweighted approximate MWC algorithm on the stretched scaled graph (see Section 4). The stretched graph may have large edge weights and such edges may not always be traversed in h^* rounds, but a $(1+\epsilon)$-approximation of any h-hop shortest path is traversed within h^* rounds in at least one of the G^i. We apply Corollary 4.1 to compute 2-approximation of h^*-hop limited MWC in each stretched G^i, and take the minimum to compute $(2+\epsilon)$-approximate h-hop limited MWC in G.

The round complexity of our algorithm is $\tilde{O}(n^{2/3} + D)$ rounds. Computing long cycles takes $\tilde{O}(n^{2/3}+D)$ rounds using sampling and $n^{1/3}$-source approximate SSSP algorithm (Theorem 1.6.B). Using the method in Corollary 4.1, computing short cycles takes $\tilde{O}((\sqrt{n}+h^*+R_{cast}) \cdot \log(nW))$ rounds, where $R_{cast} = O(D)$ and $h^* = \tilde{O}(n^{2/3})$. For detailed proofs, see the full version [40].

5.2 Approximate Directed Weighted MWC

We use the above framework for undirected graphs to compute $(2+\epsilon)$-approximation of directed weighted MWC, by replacing the hop-limited undirected unweighted MWC computation with a directed version. We can compute h-hop limited 2-approximation of MWC in stretched directed unweighted graphs in $\tilde{O}(n^{4/5} + h + R_{cast})$ rounds by applying the modifications in Corollary 4.1 to our directed unweighted MWC algorithm (Algorithm 2 from Section 3). The overall algorithm runs in $\tilde{O}(n^{4/5} + D)$ rounds, dominated by the cost of the directed unweighted MWC subroutine.

6 CONCLUSION AND OPEN PROBLEMS

We have presented several CONGEST upper and lower bounds for computing MWC in directed and undirected graphs, both weighted and unweighted. While many of our results are close to optimal, here are some topics for further research.

For $(2-\epsilon)$-approximation of MWC, we have shown nearly optimal near-linear lower bounds in all cases except girth, with near-linear round algorithms for exact computation.

For girth we present a near-optimal $\tilde{O}(\sqrt{n} + D)$-round algorithm for $(2-(1/g))$-approximation. For larger approximations, we showed a lower bound of $\tilde{\Omega}(n^{1/4})$ for arbitrarily large constant α-approximation, and a $\tilde{\Omega}(n^{1/3})$ lower bound for $(2.5-\epsilon)$-approximation in the full version [40]. The current best upper bound for girth is still $O(n)$ [11, 28], and the CONGEST complexity of exact girth remains an open problem.

We have studied larger constant ($\alpha \geq 2$) approximation of MWC in directed graphs and weighted graphs, and presented sublinear algorithms for 2-approximation ($(2+\epsilon)$ for weighted graphs) beating the linear lower bounds for $(2-\epsilon)$-approximation. Our results include: for directed unweighted MWC, $\tilde{O}(n^{4/5} + D)$-round 2-approximation algorithm; for directed weighted MWC, $(2+\epsilon)$-approximation algorithm with the same $\tilde{O}(n^{4/5} + D)$ complexity; for undirected weighted MWC, $\tilde{O}(n^{2/3} + D)$-round algorithm for $(2+\epsilon)$-approximation. For these three graph types, we showed lower bounds of $\tilde{\Omega}(\sqrt{n})$ for any α-approximation of MWC, for arbitrarily large constant α. Whether we can bridge these gaps between upper and lower bounds, or provide tradeoffs between round complexity and approximation quality is a topic for further research.

Our approximation algorithms for weighted MWC (directed and undirected) are based on scaling techniques, which introduce an additional multiplicative error causing our algorithms to give $(2+\epsilon)$-approximation instead of the 2-approximation obtained in the unweighted case. The main roadblock in obtaining a 2-approximation is an efficient method to compute exact SSSP from multiple sources, on which we elaborate below.

When $k \geq n^{1/3}$, we have presented a fast and streamlined $\tilde{O}(\sqrt{nk}+D)$-round algorithm for k-source exact directed BFS, where the key to our speedup is sharing shortest path computations from different sources using skeleton graph constructions. Using scaling techniques, we extended this to an algorithm for k-source directed SSSP in weighted graphs, but only for $(1+\epsilon)$-approximation. While there have been recent techniques for a single source to compute exact SSSP from approximate SSSP algorithms [9, 48] building on [34], it is not clear how to extend them to multiple sources seems difficult. These techniques involve distance computations on graphs whose edge-weights depend on the source. As a result, we can no longer construct a single weighted graph where we can share shortest computations for k sources. Providing an exact k-source SSSP algorithm that matches the round complexity of our k-source approximate weighted SSSP algorithm is a topic for further research.

ACKNOWLEDGMENTS

This work was supported in part by NSF grant CCF-2008241. We thank the anonymous reviewers for their helpful comments.

REFERENCES

[1] Amir Abboud, Keren Censor-Hillel, and Seri Khoury. 2016. Near-Linear Lower Bounds for Distributed Distance Computations, Even in Sparse Networks. In *Distributed Computing - 30th International Symposium, DISC 2016 (Lecture Notes in Computer Science, Vol. 9888)*. Springer, Paris, France, 29–42.

[2] Amir Abboud, Virginia Vassilevska Williams, and Joshua R. Wang. 2016. Approximation and Fixed Parameter Subquadratic Algorithms for Radius and Diameter

in Sparse Graphs. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016*. SIAM, Arlington, VA, USA, 377–391.
[3] Udit Agarwal and Vijaya Ramachandran. 2018. Fine-grained complexity for sparse graphs. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018*. ACM, Los Angeles, CA, USA, 239–252.
[4] Udit Agarwal and Vijaya Ramachandran. 2019. Distributed Weighted All Pairs Shortest Paths Through Pipelining. In *2019 IEEE International Parallel and Distributed Processing Symposium, IPDPS 2019*. IEEE, Rio de Janeiro, Brazil, 23–32.
[5] Udit Agarwal and Vijaya Ramachandran. 2020. Faster Deterministic All Pairs Shortest Paths in Congest Model. In *SPAA '20: 32nd ACM Symposium on Parallelism in Algorithms and Architectures, 2020*. ACM, Virtual Event, USA, 11–21.
[6] Bertie Ancona, Keren Censor-Hillel, Mina Dalirrooyfard, Yuval Efron, and Virginia Vassilevska Williams. 2020. Distributed Distance Approximation. In *24th International Conference on Principles of Distributed Systems, OPODIS 2020 (LIPIcs, Vol. 184)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Strasbourg, France (Virtual Conference), 30:1–30:17.
[7] Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, and D. Sivakumar. 2004. An information statistics approach to data stream and communication complexity. *J. Comput. Syst. Sci.* 68, 4 (2004), 702–732.
[8] Aaron Bernstein and Danupon Nanongkai. 2019. Distributed exact weighted all-pairs shortest paths in near-linear time. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019*. ACM, Phoenix, AZ, USA, 334–342.
[9] Nairen Cao and Jeremy T. Fineman. 2023. Parallel Exact Shortest Paths in Almost Linear Work and Square Root Depth. In *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023*. SIAM, Florence, Italy, 4354–4372.
[10] Nairen Cao, Jeremy T. Fineman, and Katina Russell. 2021. Brief Announcement: An Improved Distributed Approximate Single Source Shortest Paths Algorithm. In *Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing* (Virtual Event, Italy) *(PODC'21)*. Association for Computing Machinery, New York, NY, USA, 493–496.
[11] Keren Censor-Hillel, Orr Fischer, Tzlil Gonen, François Le Gall, Dean Leitersdorf, and Rotem Oshman. 2020. Fast Distributed Algorithms for Girth, Cycles and Small Subgraphs. In *34th International Symposium on Distributed Computing, DISC 2020 (LIPIcs, Vol. 179)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Virtual Conference, 33:1–33:17.
[12] Yi-Jun Chang, Seth Pettie, Thatchaphol Saranurak, and Hengjie Zhang. 2021. Near-optimal Distributed Triangle Enumeration via Expander Decompositions. *Journal of the ACM (JACM)* 68, 3 (2021), 1–36.
[13] Shiri Chechik and Gur Lifshitz. 2021. Optimal Girth Approximation for Dense Directed Graphs. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021*. SIAM, Virtual Conference, 290–300.
[14] Shiri Chechik and Doron Mukhtar. 2022. Single-source shortest paths in the CONGEST model with improved bounds. *Distributed Comput.* 35, 4 (2022), 357–374.
[15] Andrew Drucker, Fabian Kuhn, and Rotem Oshman. 2014. On the power of the congested clique model. In *ACM Symposium on Principles of Distributed Computing, PODC '14*. ACM, Paris, France, 367–376.
[16] Talya Eden, Nimrod Fiat, Orr Fischer, Fabian Kuhn, and Rotem Oshman. 2022. Sublinear-time distributed algorithms for detecting small cliques and even cycles. *Distributed Comput.* 35, 3 (2022), 207–234.
[17] Michael Elkin. 2006. An unconditional lower bound on the time-approximation trade-off for the distributed minimum spanning tree problem. *SIAM J. Comput.* 36, 2 (2006), 433–456.
[18] Michael Elkin and Ofer Neiman. 2019. Hopsets with Constant Hopbound, and Applications to Approximate Shortest Paths. *SIAM J. Comput.* 48, 4 (2019), 1436–1480.
[19] Michael Elkin and Chhaya Trehan. 2022. $(1+\epsilon)$-Approximate Shortest Paths in Dynamic Streams. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2022 (LIPIcs, Vol. 245)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, USA (Virtual Conference), 51:1–51:23.
[20] Sebastian Forster and Danupon Nanongkai. 2018. A Faster Distributed Single-Source Shortest Paths Algorithm. In *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018*. IEEE Computer Society, Paris, France, 686–697.
[21] Pierre Fraigniaud, Mael Luce, Frederic Magniez, and Ioan Todinca. 2024. *Even-Cycle Detection in the Randomized and Quantum CONGEST Model*. Technical Report. arXiv. arXiv:2402.12018 [cs.DC]
[22] Pierre Fraigniaud and Dennis Olivetti. 2019. Distributed Detection of Cycles. *ACM Trans. Parallel Comput.* 6, 3 (2019), 12:1–12:20.
[23] Silvio Frischknecht, Stephan Holzer, and Roger Wattenhofer. 2012. Networks cannot compute their diameter in sublinear time. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012*. SIAM, Kyoto, Japan, 1150–1162.
[24] Mohsen Ghaffari. 2015. Near-Optimal Scheduling of Distributed Algorithms. In *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing, PODC 2015*. ACM, Donostia-San Sebastián, Spain,, 3–12.
[25] Mohsen Ghaffari and Rajan Udwani. 2015. Brief Announcement: Distributed Single-Source Reachability. In *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing, PODC 2015*. ACM, Donostia-San Sebastián, Spain, 163–165.
[26] Alina Harbuzova, Ce Jin, Virginia Vassilevska Williams, and Zixuan Xu. 2024. Improved Roundtrip Spanners, Emulators, and Directed Girth Approximation. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM, Alexandria, Virginia, USA, 4641–4669.
[27] Loc Hoang, Matteo Pontecorvi, Roshan Dathathri, Gurbinder Gill, Bozhi You, Keshav Pingali, and Vijaya Ramachandran. 2019. A round-efficient distributed betweenness centrality algorithm. In *Proceedings of the 24th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, PPoPP 2019*. ACM, Washington, DC, USA, 272–286.
[28] Stephan Holzer and Roger Wattenhofer. 2012. Optimal distributed all pairs shortest paths and applications. In *ACM Symposium on Principles of Distributed Computing, PODC '12*. ACM, Madeira, Portugal, 355–364.
[29] Chien-Chung Huang, Danupon Nanongkai, and Thatchaphol Saranurak. 2017. Distributed Exact Weighted All-Pairs Shortest Paths in $\tilde{O}(n^{5/4})$ Rounds. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017*. IEEE Computer Society, Berkeley, CA, USA, 168–179.
[30] Taisuke Izumi and François Le Gall. 2017. Triangle Finding and Listing in CONGEST Networks. In *Proceedings of the ACM Symposium on Principles of Distributed Computing, PODC 2017*. ACM, Washington, DC, USA, 381–389.
[31] Taisuke Izumi, Naoki Kitamura, Takamasa Naruse, and Gregory Schwartzman. 2022. Fully Polynomial-Time Distributed Computation in Low-Treewidth Graphs. In *SPAA '22: 34th ACM Symposium on Parallelism in Algorithms and Architectures*. ACM, Philadelphia, PA, USA, 11–22.
[32] Avi Kadria, Liam Roditty, Aaron Sidford, Virginia Vassilevska Williams, and Uri Zwick. 2022. Algorithmic trade-offs for girth approximation in undirected graphs. In *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022*. SIAM, Virtual Conference / Alexandria, VA, USA, 1471–1492.
[33] Avi Kadria, Liam Roditty, Aaron Sidford, Virginia Vassilevska Williams, and Uri Zwick. 2023. Improved girth approximation in weighted undirected graphs. In *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023*. SIAM, Florence, Italy, 2242–2255.
[34] Philip N Klein and Sairam Subramanian. 1997. A randomized parallel algorithm for single-source shortest paths. *Journal of Algorithms* 25, 2 (1997), 205–220.
[35] Eyal Kushilevitz and Noam Nisan. 1996. *Communication Complexity*. Cambridge University Press, Cambridge.
[36] Frank Thomson Leighton, Bruce M Maggs, and Satish B Rao. 1994. Packet routing and job-shop scheduling in O (congestion+ dilation) steps. *Combinatorica* 14, 2 (1994), 167–186.
[37] Christoph Lenzen, Boaz Patt-Shamir, and David Peleg. 2019. Distributed distance computation and routing with small messages. *Distributed Computing* 32, 2 (2019), 133–157.
[38] Lev B. Levitin, Mark G. Karpovsky, and Mehmet Mustafa. 2010. Minimal Sets of Turns for Breaking Cycles in Graphs Modeling Networks. *IEEE Trans. Parallel Distributed Syst.* 21, 9 (2010), 1342–1353.
[39] Vignesh Manoharan and Vijaya Ramachandran. 2024. Computing Replacement Paths in the CONGEST Model. In *Structural Information and Communication Complexity (SIROCCO 2024)*. (to appear).
[40] Vignesh Manoharan and Vijaya Ramachandran. 2024. *Improved Approximation Bounds for Minimum Weight Cycle in the CONGEST Model*. Technical Report 2308.08670. arXiv. https://arxiv.org/abs/2308.08670 Full version of this paper.
[41] Danupon Nanongkai. 2014. Distributed approximation algorithms for weighted shortest paths. In *Symposium on Theory of Computing, STOC 2014*. ACM, New York, NY, USA, 565–573.
[42] Gabriele Oliva, Roberto Setola, Luigi Glielmo, and Christoforos N. Hadjicostis. 2018. Distributed Cycle Detection and Removal. *IEEE Trans. Control. Netw. Syst.* 5, 1 (2018), 194–204.
[43] David Peleg. 2000. *Distributed computing: a locality-sensitive approach*. SIAM, USA.
[44] David Peleg, Liam Roditty, and Elad Tal. 2012. Distributed Algorithms for Network Diameter and Girth. In *Automata, Languages, and Programming - 39th International Colloquium, ICALP 2012 (Lecture Notes in Computer Science, Vol. 7392)*. Springer, Warwick, UK, 660–672.
[45] Seth Pettie. 2022. Personal Communication.
[46] A. A. Razborov. 1992. On the Distributional Complexity of Disjointness. *Theor. Comput. Sci.* 106, 2 (Dec. 1992), 385–390.
[47] Liam Roditty and Roei Tov. 2013. Approximating the Girth. *ACM Trans. Algorithms* 9, 2, Article 15 (mar 2013), 13 pages.
[48] Václav Rozhon, Bernhard Haeupler, Anders Martinsson, Christoph Grunau, and Goran Zuzic. 2023. Parallel Breadth-First Search and Exact Shortest Paths and Stronger Notions for Approximate Distances. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023*. ACM, Orlando, FL, USA, 321–334.

[49] Atish Das Sarma, Stephan Holzer, Liah Kor, Amos Korman, Danupon Nanongkai, Gopal Pandurangan, David Peleg, and Roger Wattenhofer. 2012. Distributed verification and hardness of distributed approximation. *SIAM J. Comput.* 41, 5 (2012), 1235–1265.

[50] Virginia Vassilevska Williams and R. Ryan Williams. 2018. Subcubic Equivalences Between Path, Matrix, and Triangle Problems. *J. ACM* 65, 5 (2018), 27:1–27:38.