Undirected $(1+\varepsilon)$-Shortest Paths via Minor-Aggregates: Near-Optimal Deterministic Parallel and Distributed Algorithms*

Václav Rozhoň
ETH Zurich
Switzerland

Christoph Grunau
ETH Zurich
Switzerland

Bernhard Haeupler
ETH Zurich
Switzerland
Carnegie Mellon University
USA

Goran Zuzic
ETH Zurich
Switzerland

Jason Li
UC Berkeley
USA

ABSTRACT

This paper presents near-optimal deterministic parallel and distributed algorithms for computing $(1+\varepsilon)$-approximate single-source shortest paths in any undirected weighted graph.

On a high level, we deterministically reduce this and other shortest-path problems to $\widetilde{O}(1)$ [1] Minor-Aggregations. A Minor-Aggregation computes an aggregate (e.g., max or sum) of node-values for every connected component of some subgraph.

Our reduction immediately implies:

- Optimal deterministic parallel (PRAM) algorithms with $\widetilde{O}(1)$ depth and near-linear work.
- Universally-optimal deterministic distributed (CONGEST) algorithms, whenever deterministic Minor-Aggregate algorithms exist. For example, an optimal $\widetilde{O}(\text{HopDiam}(G))$-round deterministic CONGEST algorithm for excluded-minor networks.

Several novel tools developed for the above results are interesting in their own right:

- A local iterative approach for reducing shortest path computations "up to distance D" to computing low-diameter decompositions "up to distance $\frac{D}{2}$". Compared to the recursive vertex-reduction approach of [48], our approach is simpler, suitable for distributed algorithms, and eliminates many derandomization barriers.
- A simple graph-based $\widetilde{O}(1)$-competitive ℓ_1-oblivious routing based on low-diameter decompositions that can be evaluated in near-linear work. The previous such routing [64] was $n^{o(1)}$-competitive and required $n^{o(1)}$ more work.
- A deterministic algorithm to round any fractional single-source transshipment flow into an integral tree solution.
- The first distributed algorithms for computing Eulerian orientations.

CCS CONCEPTS

• **Theory of computation → Distributed algorithms**; **Parallel algorithms**.

KEYWORDS

Distributed Algorithms, Parallel Algorithms, Shortest Path

ACM Reference Format:
Václav Rozhoň, Christoph Grunau, Bernhard Haeupler, Goran Zuzic, and Jason Li. 2022. Undirected $(1+\varepsilon)$-Shortest Paths via Minor-Aggregates: Near-Optimal Deterministic Parallel and Distributed Algorithms. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing (STOC '22), June 20–24, 2022, Rome, Italy.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3519935.3520074

1 INTRODUCTION

This paper gives essentially near-optimal[2] deterministic parallel algorithms for various $(1+\varepsilon)$-approximate shortest-path-type problems in undirected weighted graphs. These problems include computing $(1+\varepsilon)$-approximate shortest paths or shortest path trees from a single source (shortened to $(1+\varepsilon)$-SSSP) and $(1+\varepsilon)$-approximate minimum transshipment (shortened to $(1+\varepsilon)$-transshipment). Our algorithms run in polylogarithmic time and require near-linear

*VR and CG received funding from the European Research Council (ERC) under the European Unions Horizon 2020 research and innovation programme (grant agreement No. 853109). BH was supported in part by NSF grants CCF-1814603, CCF-1910588, NSF CAREER award CCF-1750808, a Sloan Research Fellowship, funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (ERC grant agreement 949272), and the Swiss National Foundation (project grant 200021-184735) GZ was supported in part by the Swiss National Foundation (project grant 200021-184735). JL was supported by the Simons Foundation and the Simons Institute for the Theory of Computing.

[1] We use $\widetilde{O}$-notation to suppress polylogarithmic factors in the number of nodes n, e.g., $\widetilde{O}(m) = m \log^{O(1)} n$. We use the term near-linear to mean $\widetilde{O}(m)$.

STOC '22, June 20–24, 2022, Rome, Italy

ACM ISBN 978-1-4503-9264-8/22/06…$15.00
https://doi.org/10.1145/3519935.3520074

[2] We refer to optimality up to polylogarithmic factors as *near-optimality*, while *almost-optimality* refers to optimality up to subpolynomial, i.e., $n^{o(1)}$, factors.

work. This is optimal up to polylogarithmic factors. All prior parallel deterministic algorithms with polylogarithmic depth for $(1+\varepsilon)$-SSSP required $n^{\Theta(1)}$ more work and, to our knowledge, no subquadratic work NC algorithm was known for $(1+\varepsilon)$-transshipment. We also give deterministic distributed algorithms for the above problems, including the first non-trivial distributed deterministic $(1+\varepsilon)$-transshipment algorithm.

Computing a single-source shortest path is one of the most fundamental problems in combinatorial optimization. Already in 1956, Dijkstra [14] gave a simple deterministic $O(m \log n)$ time algorithm for computing an exact single-source shortest path tree in any directed graph. Parallel and distributed algorithms on the other hand have been subject of over three decades of extensive research [1, 7, 9, 10, 17, 19–25, 31, 36, 39, 40, 44, 47, 49, 59–61, 64, 65] with much remaining unknown. Much recent research has focused on $(1+\varepsilon)$-approximate shortest paths in undirected graphs, the problem solved in this paper. We provide a detailed summary of this prior work next and describe related work on SSSP algorithms in other computational models or on other related problems, including all-pair-shortest-path and SSSP in directed graphs, in the full version of the paper. We focus on randomized and deterministic parallel algorithms with polylogarithmic depth, i.e., RNC and NC algorithms.

Parallel Algorithm. The PRAM model of parallel computation was introduced in 1979. After a decade of intense study, Karp and Ramachandran noted in their influential 1989 survey on parallel algorithms [43] that the "transitive closure bottleneck" for shortest-path-type problem required work "far in excess of the time required to solve the problem sequentially". Indeed, these algorithms build on matrix multiplication or on transitive closure routines and require $O(n^3)$ work. A line of work [12, 44, 59–61] provided algorithms with different work-time trade-offs but without much improvement on the cubic work bound for fast parallel algorithms.

In a major breakthrough, Cohen [10] gave a randomized $(1+\varepsilon)$-SSSP algorithm based on hopsets with $\widetilde{O}_\rho(1)$ time and $O(m^{1+\rho})$ work for any constant $\rho > 0$. Algorithms with slightly improved time-work tradoffs were given [21, 22] but these algorithms still required a polynomial m^ρ factor more work than Dijkstra's algorithm to parallelize shortest path computations to polylogarithmic parallel time.

This remained the state-of-the-art until 2020 when methods from continuous optimization developed by Sherman [56, 58] enabled further progress on shortest-path type problems. In particular, Sherman gave an $m \cdot n^{o(1)}$ sequential algorithm for the $(1+\varepsilon)$-approximate minimum transshipment problem. *Transshipment*, also known as uncapacitated min-cost flow, Wasserstein distance, or optimal transport, is the other major problem solved in this paper. The input to the transshipment problem on some weighted graph G consists of a *demand* b which specifies for each node a positive or negative demand value with these values summing up to zero. The goal is to find a flow of minimum cost which sends flow from nodes with surplus to nodes with positive demands. Here the cost of routing a unit of flow over an edge is proportional to the weight/length of the edge. For example, an optimal transshipment flow for a demand with $b(s) = -1$ and $b(t) = 1$ is simply a unit flow from s to t which decomposes into a distribution over shortest (s,t)-paths. We remark that transshipment is in many ways a powerful generalization of shortest-path problems, including computing SSSP (trees), with the exception that extracting trees or paths from a continuous transshipment flow often remains a hard problem itself which requires highly nontrivial rounding algorithms. While having an innocent feel to them, such rounding steps have a history of being inherently randomized and being algorithmic bottlenecks for shortest-path problems on several occasions.

Sherman's almost-linear time transshipment algorithm did not imply anything new for SSSP at first, but it inspired two independent approaches by [48] and Andoni, Stein, and Zhong [2] that led to near-optimal randomized parallel shortest path algorithms with polylogarithmic time and near-linear work. In particular, Li [48] improved several parts of Sherman's algorithm and combined it with the vertex-reduction framework that Peng [53] had introduced to give the first near-linear time algorithm for $(1+\varepsilon)$-maximum-flow. [48] also adopts a randomized rounding algorithm of [7] based on random walks to extract approximate shortest-path trees from transshipment flows. Overall, this results in randomized PRAM algorithms for both $(1+\varepsilon)$-transshipment and $(1+\varepsilon)$-SSSP with polylogarithmic time and near-linear work. Concurrently, Andoni, Stein, and Zhong [2] achieved the same result for $(1+\varepsilon)$-transshipment by combining Sherman's framework with ideas from hop-sets. They also provide a randomized rounding based on random walks algorithm that can extract an s-t shortest path. Later, this rounding was extended in Zhong's thesis [62] to extract the full $(1+\varepsilon)$-SSSP tree.

It is remarkable that all parallel $(1+\varepsilon)$-SSSP algorithms described above (with the exception of the super-quadratic work algorithms before Cohen's 25 year old breakthrough) crucially rely on randomization for efficiency. Indeed, the only modern deterministic parallel $(1+\varepsilon)$-SSSP algorithm is a very recent derandomization of Cohen's algorithm by Elkin and Matar [19]. This algorithm suffers from the familiar $O(m^{1+\rho})$ work bound for a polylogarithmic time parallel algorithm. For the $(1+\varepsilon)$-transshipment problem even less is known. Indeed we are not aware of any efficient deterministic parallel algorithm before this work and would expect that much larger polynomial work bounds would be required to achieve a deterministic algorithm with polylogarithmic parallel time.

In this paper, we give deterministic parallel algorithms for both the $(1+\varepsilon)$-SSSP and the $(1+\varepsilon)$-transshipment problem in undirected weighted graphs. Both algorithms only require near-linear work and run in polylogarithmic time. This solves these two and various other shortest-path-type problems optimally, up to polylogarithmic factors.

Theorem 1.1 (Deterministic Parallel SSSP and Transshipment). *There is a deterministic parallel algorithm that, given an undirected graph with nonnegative weights, computes a $(1+\varepsilon)$-approximate single-source shortest path tree or a $(1+\varepsilon)$-approximation to minimum transshipment in $\widetilde{O}(m \cdot \varepsilon^{-2})$ work and $\widetilde{O}(1)$ time in the* PRAM *model for any $\varepsilon \in (0,1]$.*

Distributed Algorithm. We also give new distributed $(1+\varepsilon)$-SSSP and $(1+\varepsilon)$-transshipment algorithms in the standard CONGEST model of distributed computing. For distributed algorithms a lower bound of $\Omega(\sqrt{n} + \mathrm{HopDiam}(G))$ rounds is known for worst-case topologies [16, 55], where $\mathrm{HopDiam}(G)$ is the unweighted (i.e., hop) diameter of the network. We refer to this as the *existential*

lower bound since it depends on the parameterization by $(n, \text{HopDiam}(G))$, as opposed to the later-discussed *universal lower bound* which does not.

In his 2004 survey on distributed approximation, Elkin [15] pointed to the distributed complexity of shortest paths approximations as one of two fundamental and wide-open problems in the area—as no non-trivial algorithms complementing the above lower bounds were known.

Since then, the $(1+\varepsilon)$-SSSP problem is one of the most studied problems. We only give a brief summary here: Lenzen and Patt-Shamir gave an $\widetilde{O}(n^{1/2+\varepsilon}+\text{HopDiam}(G))$-round $O(1+\frac{\log 1/\varepsilon}{\varepsilon})$-SSSP algorithm [46] and a $(1+o(1))$-approximation in $\widetilde{O}(n^{1/2}\cdot \text{HopDiam}(G)^{1/4}+\text{HopDiam}(G))$ rounds was given by Nanongkai [49]. Building on Sherman's transshipment framework mentioned before, [7] gave a randomized $\widetilde{O}(\varepsilon^{-3}\cdot(\sqrt{n}+\text{HopDiam}(G)))$ algorithm. The first algorithm improving over the $\Omega(\sqrt{n})$ barrier was given in [36] with a running time of $\text{ShortcutQuality}(G)\cdot n^{o(1)}$ albeit with a bad $n^{o(1)}$-approximation. This was recently improved by [64] to a $(1+\varepsilon)$-approximation with the same round complexity. Using [28], this gives a $(1+\varepsilon)$-approximation algorithm with round complexity $\text{HopDiam}(G)\cdot n^{o(1)}$ for any excluded minor topology (e.g., planar graphs). All the above algorithms are randomized: The only deterministic distributed algorithm is the $(1+o(1))$-SSSP algorithm of [40] with a $(n^{1/2+o(1)}+\text{HopDiam}(G)^{1+o(1)})$ running time. For $(1+\varepsilon)$-transshipment, the only known sublinear distributed algorithm is the randomized $\text{ShortcutQuality}(G)\cdot n^{o(1)}$-round algorithm of [64]. No non-trivial deterministic distributed $(1+\varepsilon)$-transshipment algorithm (with sub-linear round complexity) was known prior to this work.

The distributed results of this paper include deterministic distributed algorithms for $(1+\varepsilon)$-SSSP, its generalization $(1+\varepsilon)$-set-source shortest path (in which we are looking for distances from a subset of nodes), and $(1+\varepsilon)$-transshipment. Our results improve over the previous best running times of deterministic distributed algorithms for each of the above problems, both in the worst-case and for any excluded minor graph. Both the round and the message complexities (in KT_0, see [52] for the definition) of our algorithms are (existentially) optimal, up to $\text{poly}\log n$ factors, in all cases.

Theorem 1.2 (Deterministic Distributed SSSP and Transshipment). *There are deterministic* CONGEST *algorithms that, given an undirected graph with non-negative weights, compute a $(1+\varepsilon)$-approximate set-source shortest path forest or a $(1+\varepsilon)$-transshipment solution for any $\varepsilon\in(0,1]$. Our algorithms have an optimal message complexity of $\widetilde{O}(m)\cdot\varepsilon^{-2}$ and are guaranteed to terminate*

(1) within at most $\widetilde{O}(\sqrt{n}+\text{HopDiam}(G))\cdot\varepsilon^{-2}$ rounds and
(2) within at most $\widetilde{O}(\text{HopDiam}(G))\cdot\varepsilon^{-2}$ rounds if G does not contain any $\widetilde{O}(1)$-dense minor.

Universal optimality. Recently, the pervasive $\widetilde{\Omega}(\text{HopDiam}(G)+\sqrt{n})$ distributed lower bounds have been extended to a near-tight universal lower bound [38] which shows that most optimization problems including all problems studied in this paper require $\widetilde{\Omega}(\text{ShortcutQuality}(G))$ rounds on any communication graph G. Here $\text{ShortcutQuality}(G)$ is a natural graph parameter (we refer the interested reader to [38] for a formal definition). For experts interested in universally-optimal distributed algorithms we remark that the results of this paper imply strong conditional results. We state these results for the interested reader in Section 2.3.

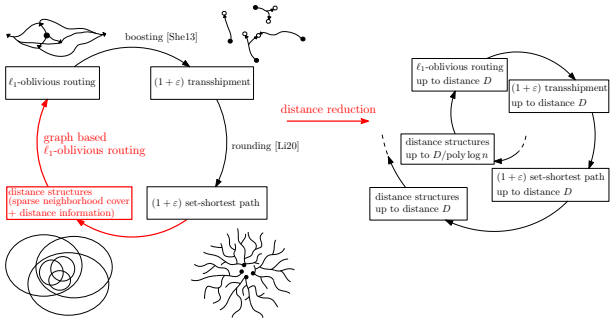

Figure 1: This figure summarizes our approach with our contributions marked in red. It is known that a $\widetilde{O}(1)$-competitive solution to ℓ_1-oblivious routing can be boosted to a $(1+\varepsilon)$-approximate transshipment which can then be rounded to yield a $(1+\varepsilon)$-approximate SSSP tree. We close the circle of reductions by constructing certain distance structures from the approximate SSSP tree. Think of those distance structures as a family of clusterings on different scales, storing some additional distance information. One of our main technical contributions is the efficient construction of a $\widetilde{O}(1)$-approximate oblivious routing from the distance structures, which closes the loop.
This loop of reductions itself is not very useful as the complexity of the problems does not decrease. We break this loop by our *distance reduction* framework by showing one can construct distance structures "up to distance D" using distance structures "up to distance $D/\text{poly}\log n$. After $O(\log n)$ iterations, we build up the desired solution to any of the four problems, in particular to approximate transshipment and set-SSSP.

1.1 Technical Overview

While the results for SSSP and transshipment are important, the main impact of this paper will likely be the new tools and algorithmic ideas developed in this paper. We expect these ideas to be applicable beyond shortest path problems. Next, we give a brief summary of the most relevant parts and ideas of prior work which are needed to understand our work and put in proper context. We then give an informal high-level overview of our new algorithm and some of the new tools developed for it. A readable and more precise technical proof overview is given in Section 2.

1.1.1 Background and Prior Work.

Transshipment Boosting, ℓ_1-Oblivious Routing, and Transshipment Flow Rounding. All modern SSSP-algorithms, including ours, compute shortest paths via the transshipment problem (see text before Theorem 1.1 for a definition). The key idea in this approach of Sherman [56, 58] is that even a rather bad α-approximation algorithm for (dual) transshipment can be *boosted* to a $(1+\varepsilon)$-approximation. This is achieved via the multiplicative weights method (or equivalently: gradient descent) and requires only $\text{poly}(\alpha, \varepsilon^{-1}, \log n)$ invocations

of the α-approximation algorithm [7, 58, 63]. A particularly convenient way of obtaining such a boostable α-approximation is to design a linear matrix R which maps any node-demand b to an α-approximate transshipment flow for b. Such a matrix is called an ℓ_1-oblivious routing because linearity forces each node to route its demand obliviously, i.e., without knowledge of the demand on other nodes. As mentioned before, in order to obtain an actual $(1+\varepsilon)$-SSSP tree from a transshipment flow most algorithms [48] are using an approach of [6] which produces a $(1+\varepsilon)$-SSSP tree after $O(\log n)$ adaptive applications of some rounding algorithm. All parts of this reduction except for the rounding algorithm are deterministic.

Putting all these pieces together proves that all one needs to obtain an efficient (deterministic) algorithm for both the $(1+\varepsilon)$-transshipment and the $(1+\varepsilon)$-SSSP problem is a $\widetilde{O}(1)$-competitive ℓ_1-oblivious routing that can be evaluated efficiently and a (deterministic) rounding algorithm. Our algorithm uses these steps in a black-box fashion except that we need to replace the randomized rounding algorithm by a new deterministic transshipment flow rounding procedure.

Vertex Reduction Framework. Unfortunately, it is clear that ℓ_1-oblivious routing cannot be done without having some approximate shortest path information. This chicken and egg problem is resolved via a clever vertex-reduction framework of Li [48]. The vertex-reduction framework relies on a cyclic sequence of efficient problem reductions. For transshipment [48], these problems are SSSP, transshipment, ℓ_1-oblivious routing, and ℓ_1-embedding. Specifically:

- To solve $(1+\varepsilon)$-SSSP on G, it is sufficient to solve $(1+\varepsilon)$-transshipment on G,
- for which it is sufficient to construct $\widetilde{O}(1)$-competitive ℓ_1-oblivious routing on G (via boosting),
- for which it is sufficient to construct an $\widetilde{O}(1)$-distortion ℓ_1-embedding on G,
- for which it is sufficient to solve $O(\log^2 n)$ instances of $\widetilde{O}(1)$-SSSP on a sequence of graphs $G'_1, G'_2, \ldots, G'_{O(\log^2 n)}$, which are resolved recursively.

As stated, this simply reduced $(1+\varepsilon)$-SSSP on G to multiple $\widetilde{O}(1)$-SSSP on graphs that are slightly larger than G, which isn't particularly helpful on its own. The key idea to transform this "branching cycle" into a branching spiral that terminates (fast) is to add a step into the cycle which applies ultrasparsification on G'_i, which is an operation that transforms a graph with n nodes to a smaller graph with $\frac{n}{\gamma}$ nodes such that distances in the smaller graph $\widetilde{O}(\gamma)$-approximate distances in the larger graph. Applying this step, we can reduce $(1+\varepsilon)$-SSSP on a graph with n nodes to $O(\log^2 n)$ instances of $\widetilde{O}(1)$-SSSPs on smaller graphs with $O\left(\frac{n}{\log^{100} n}\right)$ nodes. It is easy to see that the total size of all recursive calls falls exponentially on each subsequent level, hence the total parallel runtime is $\widetilde{O}(1)$ and the total work $\widetilde{O}(m)$ (since a single sequence of reductions requires $\widetilde{O}(m)$ work).

Minor Aggregates and the Low-Congestion Shortcuts Framework. The low-congestion shortcuts framework was originally intended for designing simple and efficient *distributed* graph algorithms and was developed over a long sequence of works [27–30, 33–35, 37, 38, 45]. However, this paper argues that the framework provides a natural language even for developing fast parallel algorithms. Indeed, we present our parallel SSSP algorithm using the framework and our hope is that this choice simplifies the exposition, even before considering the benefits of immediately obtaining distributed results.

We describe our algorithms in the recently introduced Minor-Aggregation model [32, 64], which offers an intuitive interface to the recent advancements in the low-congestion shortcut framework. In the Minor-Aggregation model, one can (1) contract edges, thereby operating a minor, (2) each (super)node in the contracted graph can compute an aggregate (e.g., min, max, sum) of surrounding neighbors, and (3) add $\widetilde{O}(1)$ arbitrarily-connected virtual nodes over the course of the algorithm. The goal is to design $\widetilde{O}(1)$-round algorithms in this model. Such a Minor-Aggregation algorithm can be compiled to a near-optimal algorithm in both the parallel and distributed settings.

1.1.2 Our New Tools.

Near-Optimal Graph-Based ℓ_1-oblivious routing. The first key contribution of this paper is a new construction of graph-based ℓ_1-oblivious routing with drastically improved guarantees from the so-called sparse neighborhood covers. A sparse neighborhood cover of distance scale D is a collection of $O(\log n)$ clusterings (partitioning of the node set into disjoint *clusters*) such that (1) each cluster has diameter at most D, and (2) each ball in G of radius $D/(\log^C n)$ is fully contained in at least one cluster (for some fixed constant $C > 0$). Specifically, given sparse neighborhood covers for all $O(\log n)$ exponentially-increasing distance scales $\beta, \beta^2, \beta^3, \ldots, \text{poly}(n)$ for some $\beta = \widetilde{O}(1)$ along with some extra distance information, we construct an $\widetilde{O}(1)$-competitive ℓ_1-oblivious routing. The algorithm greatly differs from all prior approaches, as all of them had inherent barriers preventing them from achieving deterministic near-optimality, which we describe below.

Derandomization issues: Efficient constructions of an ℓ_1-oblivious routing either use an ℓ_1-embedding or so-called low-diameter decompositions; this is a clustering problem whose deterministic version is also known as a sparse neighborhood cover. Sparse neighborhood covers were introduced in the seminal work of [4] and applied with great success for many problems, including approximate shortest paths and other distance based problems [5, 13].

Since an efficient deterministic ℓ_1-embedding is not known, we need a deterministic construction of sparse neighborhood covers. Luckily, many of the ideas required to derandomize the computation of sparse neighborhood covers were developed very recently by a sequence of papers that derandomized the closely related network decomposition problem which is, essentially, an unweighted version of the sparse cover problem [8, 26, 54]. In [18], these unweighted results were extended to an algorithm constructing sparse neighborhood covers for weighted graphs and this is the result that we use to solve the shortest path problem here.

Distance-reduction framework: Iterative and Locality-Friendly. As we explained earlier, our new ℓ_1-oblivious routing construction reduces the SSSP problem in G to computing sparse neighborhood covers along with some extra distance information. Clearly, this

requires our SSSP computation to compute some distance information, leading again to a chicken and egg problem. Unfortunately, directly applying the vertex-reduction framework is not compatible with our desire to obtain fast minor-aggregation (or distributed) algorithms. In particular, at the lowest level of the recursion, the vertex-reduction framework generates a polynomial number of constant-size SSSP problems, each of which essentially corresponds to a minor of G. While these problems can each be solved in constant time in the parallel setting, the fact that these minors can be arbitrarily overlapping and each correspond to large diameter subsets in G means that polynomially many instead of the desired polylogarithmic number of minor aggregations are necessary.

This paper therefore designs a novel, completely different, and more locality-friendly complexity-reduction framework: *distance reduction*. On a high-level, we show that obtaining "sparse neighborhood covers up to distance scale D" can be reduced to several "shortest path computations up to distance D" which are computable from "sparse neighborhood covers up to distance $D/\text{poly}\log n$" (see Figure 1). This iterative and local nature of our distance-reduction framework directly translates into a small overall number of minor aggregations; this is in stark contrast to the inherently recursive vertex-reduction framework (which recurses on different graphs, requiring a recursive approach). As a nice little extra, combining our distance-reduction framework with the new ℓ_1-embedding makes for an algorithm that is (in our not exactly unbiased opinion) a good bit simpler than the previous algorithms of [2, 48]. We note that this framework is more reminiscent of older approaches to hopset constructions [10, 20]. In these approaches, hopsets over longer paths which use at most $\ell/2$ edges are used to bootstrap the construction of hopsets over paths using at most ℓ edges.

A key definition to formalize what exactly the "up to distance D" in our distance-reduction framework means is the following. We attach a virtual, so-called, *cheating node* and connect it to all other nodes with an edge of length D. This new graph naturally preserves distance information "up to distance D" and the complexity of computing distances increases as D increases.

Derandomization: Deterministic Transshipment Flow Rounding via Eulerian Tours. All previous transshipment approaches crucially use randomization. The most significant challenge we had to overcome in making our results deterministic stem from the following issue: The only problem that remains to be derandomized in this paper is *rounding* a (fractional) transshipment solution to (a flow supported on) a tree, a crucial step in the distance-reduction framework. We prove the following theorem (only the parallel version is stated for simplicity).

Theorem 1.3. *There is a deterministic parallel algorithm which takes as an input any fractional transshipment flow f satisfying some single-source transshipment demand b in the graph G and in near-linear work and polylogarithmic time outputs a flow f' of equal or smaller cost which is supported on some tree in G.*

While such a flow rounding seems an unlikely bottleneck for a deterministic SSSP algorithm, we remark that even for randomized algorithms a lot of complexity has come from this rounding step in the algorithms of [2, 7, 48]—all these approaches are based on random walks and are inherently randomized.

Note that if the fractional flow f is acyclic (has no *directed* cycles), then there is a trivial randomized rounding which simply samples one outgoing edge for each node in G through which flow is routed, choosing the probability of each edge proportional to the outflow in f. It is easy to see that retaining the edges that are in the connected component of the source truthfully samples a tree-supported flow from f. The complexity comes in once f is not acyclic as the sampled edges can now create many connected components. This requires finding the cycles in these components, contracting the edges and again running a randomized out-edge sampling on the remaining graph.

Deterministically none of the above works. Indeed, we are not aware of any simple(r) way of deterministically obtaining a tree-supported flow even if f is acyclic. Our rounding procedure can be seen as a generalization of an algorithm of Cohen's rounding [11], which can be used to round any fractional transshipment flow to an integral flow by scaling flow values to only leave integral and half-integral flow values and then finding an Eulerian tour covering all half-integral edges. Pushing one half-unit of flow in the cheaper direction of this Eulerian tour makes all flow values fully integral and allows the scaling to be reduced. At the end of this procedure, all flow values are integral but this does not guarantee that the flow is supported on a tree. To eliminate any non-tree like parts of the flow we show how to keep the algorithm running and find further Eulerian tours until the flow becomes tree-supported.

A Distributed Eulerian Tour Algorithm. The problem of computing Eulerian Tours can be stated as follows. Given an undirected graph with all degrees even, direct the edges in such a way that the out-degree of every node equals the in-degree. Note that we do not require connectedness. While computing Eulerian tours is a well-known parallel primitive which can be efficiently computed in near-linear work and polylogarithmic time [3], there are, to our knowledge, no distributed algorithms known for this problem.

Therefore, to also give *distributed* SSSP algorithms in this paper we need to design efficient distributed Eulerian-tour algorithms that can then be used in the Eulerian-tour-based rounding procedure of Theorem 1.3. We build the first algorithm computing such an Eulerian tour orientation by using algorithms from [50, 51] for low-congestion cycle covers. Interestingly, these low-congestion cycle covers were only developed for the completely unrelated purpose of making distributed computation resilient to Byzantine faults introduced by an adversary in a recent line of work [41, 42, 50, 51].

Theorem 1.4 (Informal). *There is a deterministic* CONGEST *algorithm which, given any Eulerian subgraph H of the network G as an input, computes an Eulerian tour orientation in $\widetilde{O}(\sqrt{n}+\text{HopDiam}(G))$ rounds or $\widetilde{O}(\text{HopDiam}(G))$ rounds if G is an excluded-minor graph.*

The most general and fully formal statements of all results proven in this paper are given in Section 2.3.

2 SSSP VIA MINOR-AGGREGATIONS: A LOCAL ITERATIVE REDUCTION CYCLE

In Section 1 and Figure 1, we gave an idealized and informal description of our algorithm. The real set of reductions our algorithm is built on is not quite the perfect cycle from Figure 1, but instead

it looks like Figure 2. In Section 2.1, we give a formal definition for each part of this new "cycle" and explain how the parts of the new cycle correspond to parts in the old cycle. Once we have defined each part of the new cycle, each arrow in Figure 2 corresponds to a formal statement. These formal statements can be found in Section 2.2, along with informal explanations how the statements can be proven.
Finally, Section 2.3 contains statements of our main theorems together with simple proof sketches. The formal proofs for all the results stated in Section 2.2 and Section 2.3 can be found in the full version of the paper.

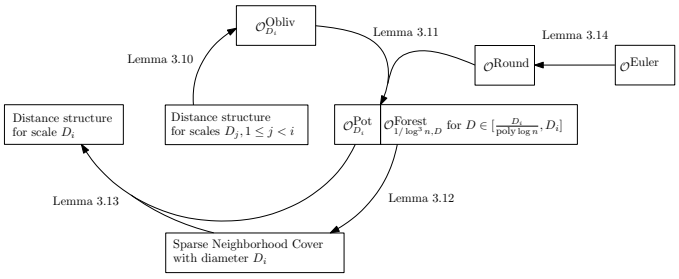

Figure 2: The figure illustrates a single iteration of our local iterative reduction cycle. At the beginning of the cycle, the algorithm has already computed a distance structure for every scale D_j with $D_j < D_i$. After completion of the cycle, the algorithm has computed a distance structure for scale D_i, under the assumption that it has access to the rounding oracle O^{Round} (or the Eulerian-Orientation oracle O^{Euler}). Lemma 2.9 states this result formally. Moreover, each arrow in the cycle corresponds to a formal statement of the following form: Given A (and B), then one can efficiently (in $\widetilde{O}(1)$ Minor-Aggregation rounds) compute C.

2.1 Key Definitions and Oracles

The definition of the oblivious routing oracle $O^{Obliv}_{D_i}$ relies on the definition of the graph $G_{S,D}$ and the definition of distance scales.

We start with the definition of the graph $G_{S,D}$.

Definition 2.1 (Graph $G_{S,D}$)**.** Let $S \subseteq V$ and $D \in [\text{poly}(n)]$. We construct the graph $G_{S,D} = (V \cup \{v_D, s^*\}, E \cup \{\{v_D, u\} \mid u \in V\} \cup \{\{s^*, u\} \mid u \in S\})$ by adding two additional nodes v_D and s^* to G. The node v_D is connected to each node in V and the node s^* is connected to all the nodes in S, with all the new edges having a weight of D. Moreover, we denote with G_S the unweighted graph one obtains by discarding the edge weights in the weighted graph $G_{S,D}$.

Informally speaking, $G_{S,D}$ preserves distance information only up to distance $2D$. More formally, any shortest path between two nodes in G remains a shortest path in $G_{S,D}$ if this path is of length at most $2D$. Shortest paths in G longer than $2D$, on the other hand, are not preserved in $G_{S,D}$, since any two nodes in $G_{S,D}$ have a shortest path of length $2D$ via v_D. Moreover, consider a shortest path tree from vertex s^* up to distance $2D$. If one removes s^* and v_D from this tree, then the resulting forest is a shortest path forest in G from set S up to distance D.

We next give the definition of distance scales, along with defining global parameters which are used throughout the paper.

Definition 2.2 (Distance Scales)**.** We set $\tau = \log^7(n)$, $\beta = 8\tau$ and $D_i = \beta^i$.

A distance scale is a value contained in the set $\{D_i : i \in \mathbb{N}, D_i \le n^2 \max_{e \in E} \ell(e)\}$. In particular, if we denote with i_{max} the largest integer i for which D_i is a distance scale, then $D_{i_{max}} \ge \text{diam}(G)$.

We are finally ready to formally define oracle $O^{Obliv}_{D_i}$, the ℓ_1-Oblivious Routing Oracle for scale D_i. It corresponds to computing an "ℓ_1-oblivious routing up to distance D_i" in Figure 1.

Definition 2.3 (ℓ_1-Oblivious Routing Oracle for scale D_i – $O^{Obliv}_{D_i}$)**.** The ℓ_1-Oblivious Routing Oracle for scale D_i, $O^{Obliv}_{D_i}$, takes as input a set $S \subseteq V$ as well as a demand b and a flow f for G_{S,D_i}. It outputs $R_S b$ and $R_S^T f$ for a fixed $\widetilde{O}(1)$-competitive ℓ_1-oblivious routing R_S for G_{S,D_i}.

Next, we give the formal definition of the forest oracle $O^{Forest}_{\varepsilon,D}$.

Definition 2.4 ($(1+\varepsilon)$-Approximate Forest for distance D rooted at S / Forest Oracle – $O^{Forest}_{\varepsilon,D}$)**.** The forest oracle $O^{Forest}_{\varepsilon,D}$ takes as input a node set $S \subseteq V$. The output is a $(1+\varepsilon)$-approximate forest for distance D rooted at S, which is a forest F rooted at S such that the following holds.

(1) For every $u \in V(G)$ with $\text{dist}_G(S,u) \le D$ we have $u \in V(F)$.
(2) For every $u \in V(F)$, $\text{dist}_F(S,u) \le (1+\varepsilon)D$.

For a given distance scale D_i, having access to $O^{Forest}_{\frac{1}{\log^3(n)}, D}$ for every $D \in \left[\frac{D_i}{\tau}, D_i\right]$ corresponds to "$(1+\varepsilon)$ set-shortest path up to distance D_i" in Figure 1.

The notion of "$(1+\varepsilon)$ transshipment up to distance D" in Figure 1 does not have a direct correspondence in Figure 2. However, the potential oracle $O^{Pot}_{D_i}$ outputs a potential capturing a certain version of (dual) transshipment potentials. The oracle $O^{Pot}_{D_i}$ is defined as follows:

Definition 2.5 (Potential for scale D_i with respect to set S / Potential Oracle – $O^{Pot}_{D_i}$)**.** The potential oracle $O^{Pot}_{D_i}$ takes as input a node set $S \subseteq V$. The output is a potential $\phi_{S,D_i} \in \mathbb{R}^{V(G_{S,D_i})}$ for scale D_i with respect to S. A potential for scale D_i with respect to a set $S \subseteq V$ is a non-negative potential ϕ_{S,D_i} such that:

(1) $\forall v \in S : \phi_{S,D_i}(v) = 0$ and
(2) $\text{dist}_G(v,S) \ge \frac{D_i}{\tau}$ implies $\phi_{S,D_i}(v) \ge 0.5\frac{D_i}{\tau}$.

We next give the definition of a distance structure for scale D_i.

Definition 2.6 (Distance Structure for scale D_i)**.** A distance structure for scale D_i consists of a sparse neighborhood cover with covering radius $\frac{D_i}{\tau}$. Moreover, each cluster C in one of the clusterings comes with

(1) a tree T_C of diameter at most D_i which spans C and is rooted at some node $v_C \in C$. We refer to v_C as the cluster center.
(2) A potential for scale D_i with respect to $V \setminus C$ (known to nodes in C).

In Figure 1, "distance structures up to distance D" correspond to having a distance structure for every scale D_j with $D_j \leq D$.

In Section 1 we outlined the problem of rounding a transshipment flow, and that rounding can be deterministically reduced to solving $\widetilde{O}(1)$ Eulerian-Orientation problems. We now define the oracles for the two problems.

The rounding oracle is defined as follows.

Definition 2.7 (Rounding Oracle — O_ε^{Round})**.** The Rounding Oracle O_ε^{Round} takes as input a weighted graph H with length function ℓ_H and a flow f on H. The weighted graph H needs to be a subgraph of some graph H' that one can obtain from G by adding up to $\widetilde{O}(1)$ virtual nodes and adding edges of arbitrary nonnegative length incident to the virtual nodes. The flow f needs to satisfy the following condition. Let b be the demand that f routes. Then, $b(v) \geq 0$ for all $v \in V(H)$ except for some $s \in V(H)$ called the source. The output is a rooted tree T with root s spanning all vertices of H with non-zero demand such that $\sum_{v:b(v)\neq 0} b(v)\,\mathrm{dist}_T(s,v) \leq (1+\varepsilon')\ell_H(f)$ with $\varepsilon' = \min\left(\varepsilon, \frac{1}{20\log^3(n)\tau}\right)$.

Note that O_ε^{Round} takes as input a weighted graph that can have more vertices than G. This allows us to use O_ε^{Round} to round a transshipment flow defined on the graph $G_{S,D}$. Moreover, we sometimes just write O^{Round}, without a precision parameter ε, which we define as $O^{Round} = O_1^{Round} = O^{Round}_{\frac{1}{20\log^3(n)\tau}}$.

The Eulerian-Orientation oracle is defined next.

Definition 2.8 (Eulerian-Orientation Oracle — O^{Euler})**.** The Eulerian-Orientation oracle O^{Euler} takes as input a Eulerian graph H. The graph H has to be a subgraph of some graph H' that one can obtain from G by adding up to $\widetilde{O}(1)$ virtual nodes and adding any edges incident to the virtual nodes. The output is an orientation of the edges of H such that the in-degree of every node is equal to its out-degree.

2.2 Formal Statements Corresponding to Figure 2

For the sake of this section, we say that we can solve a problem or compute a structure *efficiently* if there exists a Minor-Aggregation algorithm for the task that runs in $\widetilde{O}(1)$ rounds.

In this subsection, we give one formal statement for each arrow in Figure 2, as promised at the beginning of Section 2. Before that, we state a result which captures the main essence of our local iterative reduction cycle.

Lemma 2.9 (Main Lemma)**.** *Assume a distance structure for every scale D_j smaller than scale D_i and oracle O^{Round} are given. A distance structure for scale D_i can be efficiently computed.*

Lemma 2.9 is a simple corollary of the next four lemmas. The formal proof can be found in the full version of the paper.

Lemma 2.10. *Assume a distance structure for every scale D_j smaller than scale D_i is given. Then, $O^{Obliv}_{D_i}$ can be efficiently computed.*

Lemma 2.10 follows from a simple adaption of our $\widetilde{O}(1)$-competitive ℓ_1-oblivious routing construction, explained in the full version of the paper.

Lemma 2.11. *Assume oracle O^{Round} and $O^{Obliv}_{D_i}$ are given. Then, $O^{Pot}_{D_i}$ and $O^{Forest}_{\frac{1}{\log^3(n)},D}$ can be efficiently computed for any $D \in \left[\frac{D_i}{\tau}, D_i\right]$.*

Lemma 2.11 follows mainly from previous work. More precisely, having access to a $\widetilde{O}(1)$-competitive ℓ_1-oblivious routing in G_{S,D_i}, we can compute the following two objects via boosting and rounding [7, 48, 57, 64]. First, a $(1+\varepsilon)$-SSSP-tree in G_{S,D_i} rooted at s^*. Second, an individually good $(1+\varepsilon)$-approximate potential in G_{S,D_i} for the single-source transshipment demand with source s^*. For completeness, we give a summary of these steps in the proof of the full version of the paper.

If one looks at what the aforementioned tree and potential in G_{S,D_i} correspond to in the graph G, then one can relatively straightforwardly transform them to obtain a $\left(1+\frac{1}{\log^3 n}\right)$-approximate forest for D rooted at S and a potential for scale D_i with respect to S, assuming $\varepsilon = \frac{1}{\mathrm{poly}(\log n)}$ is sufficiently small. The details of this transformation can be found in the full version of the paper.

Lemma 2.12. *Assume oracle $O^{Forest}_{\frac{1}{\log^3(n)},D}$ is given for every $D \in \left[\frac{D_i}{\tau}, D_i\right]$. Then, a sparse neighborhood cover with covering radius $\frac{D_i}{\tau}$ together with a rooted spanning tree T_C of diameter at most D_i for every cluster C in the cover can be computed efficiently.*

Lemma 2.12 directly follows from [18, Theorem C.4], which is proven by adapting the algorithms of [8, 54] for the closely-related network decomposition problem.

Lemma 2.13. *Assume we are given an oracle $O^{Pot}_{D_i}$ and a sparse neighborhood cover with covering radius $\frac{D_i}{\tau}$ together with a rooted spanning tree T_C of diameter at most D_i for every cluster C in the cover. Then, a distance structure for scale D_i can be computed efficiently.*

Given a sparse neighborhood cover together with a tree for each cluster, it only remains to compute the potential for scale D_i with respect to $V \setminus C$ for each cluster C in the sparse neighborhood cover. One can compute the potentials for all the clusters in a given clustering $\mathcal{C}$ simultaneously. The simplest approach for computing these potentials is to compute a single potential for scale D_i with respect to all the nodes that are neighboring one of the clusters in $\mathcal{C}$. This approach works as long as there does not exist a node that is both clustered and neighboring a different cluster. Even though this can indeed happen, there is a simple solution that solves this problem. The details can be found in the proof of Lemma 2.13 in the full version of the paper.

Lemma 2.14. *Assume the oracle O^{Euler} is given. Then, O^{Round} can be efficiently computed.*

The main ideas to prove this lemma were already discussed in the introduction. The details can be found in the full version of the paper.

2.3 Main Theorems

In this part, we state the main theorems of this paper.

First of all, a simple induction proof on top of Lemma 2.9 leads to the following result.

Lemma 2.15. *Assume oracle O^{Round} is given. Then, a distance structure for every scale D_i can be efficiently computed.*

Note that for $i = 1$, Lemma 2.9 states that given access to O^{Round}, one can efficiently compute a distance structure for scale D_1. The complete induction proof can be found in the full version of the paper.

Given access to a distance structure for every scale D_i, the theorem below follows directly from our $\widetilde{O}(1)$-competitive ℓ_1-oblivious routing scheme described in the full version of the paper.

Theorem 2.16 (Distance Structures give ℓ_1-Oblivious Routing). *Assume a distance structure for every scale D_i is given. Then, there exists a $\widetilde{O}(1)$-competitive ℓ_1-oblivious routing R for G for which R and R^T can be efficiently evaluated.*

As discussed in Section 1, ℓ_1-oblivious routing and rounding is sufficient to solve the $(1+\varepsilon)$-SSSP tree and the $(1+\varepsilon)$-transshipment problems [7, 48, 57, 64].

Theorem 2.17 (ℓ_1-Oblivious Routing gives SSSP and transshipment). *Assume oracle $O^{Round}_{\varepsilon/2}$ is given for some $\varepsilon \in (0, 1]$ and that there exists an efficient algorithm to evaluate R and R^T for some $\widetilde{O}(1)$-competitive ℓ_1-oblivious routing R for G. Then, the $(1+\varepsilon)$-transshipment problem and the $(1+\varepsilon)$-SSSP-tree problem in G can be solved in $\widetilde{O}(1/\varepsilon^2)$* Minor-Aggregation *rounds.*

Combining Lemma 2.15, Theorem 2.16 and Theorem 2.17 results in the following theorem.

Theorem 2.18. *Assume oracle $O^{Round}_{\varepsilon/2}$ is given for some $\varepsilon \in (0, 1]$. The $(1+\varepsilon)$-transshipment problem and the $(1+\varepsilon)$-SSSP-tree problem in G can be solved in $\widetilde{O}(1/\varepsilon^2)$* Minor-Aggregation *rounds.*

The theorem above together with the fact that O^{Round} can be efficiently implemented given O^{Euler} implies the following result.

Theorem 2.19. *Assume oracle O^{Euler} is given and let $\varepsilon \in (0, 1]$. The $(1+\varepsilon)$-transshipment problem and the $(1+\varepsilon)$-SSSP-tree problem in G can be solved in $\widetilde{O}(1/\varepsilon^2)$* Minor-Aggregation *rounds.*

The Eulerian-Orientation problem can be solved with near-linear work and polylogarithmic depth [3]. Together with the theorem above and the fact that each Minor-Aggregation round can be simulated with near-linear work and polylogarithmic depth, we obtain our main parallel result.

Theorem 1.1 (Deterministic Parallel SSSP and Transshipment). *There is a deterministic parallel algorithm that, given an undirected graph with nonnegative weights, computes a $(1+\varepsilon)$-approximate single-source shortest path tree or a $(1+\varepsilon)$-approximation to minimum transshipment in $\widetilde{O}(m \cdot \varepsilon^{-2})$ work and $\widetilde{O}(1)$ time in the* PRAM *model for any $\varepsilon \in (0, 1]$.*

Moreover, our CONGEST algorithms for the Eulerian-Orientation problem developed in the full version of the paper together with general simulation results for the CONGEST model developed in prior work, we obtain our main result in the CONGEST model.

Theorem 1.2 (Deterministic Distributed SSSP and Transshipment). *There are deterministic* CONGEST *algorithms that, given an undirected graph with non-negative weights, compute a $(1+\varepsilon)$-approximate set-source shortest path forest or a $(1+\varepsilon)$-transshipment solution for any $\varepsilon \in (0, 1]$. Our algorithms have an optimal message complexity of $\widetilde{O}(m) \cdot \varepsilon^{-2}$ and are guaranteed to terminate*

(1) within at most $\widetilde{O}(\sqrt{n} + \text{HopDiam}(G)) \cdot \varepsilon^{-2}$ rounds and
(2) within at most $\widetilde{O}(\text{HopDiam}(G)) \cdot \varepsilon^{-2}$ rounds if G does not contain any $\widetilde{O}(1)$-dense minor.

We finish this section by stating the conditional results on universally optimal SSSP and transshipment algorithms one can obtain from this work:

Theorem 2.20. *Suppose there exists a deterministic algorithm for partwise aggregation that runs in* $\text{ShortcutQuality}(G) \cdot n^{o(1)}$ CONGEST *rounds, then there exist deterministic $(1+\varepsilon)$-SSSP and $(1+\varepsilon)$-transshipment algorithms with a round complexity of* $\text{ShortcutQuality}(G) \cdot n^{o(1)}$*, which is universally-optimal up to a $n^{o(1)}$-factor.*

We get even stronger conditional results if better CONGEST algorithms for computing cycle covers as defined in [50, 51] are given.

Theorem 2.21. *Suppose there exists a deterministic algorithm for partwise aggregation that runs in $\widetilde{O}(\text{ShortcutQuality}(G))$* CONGEST *rounds and a $(\widetilde{O}(1), \widetilde{O}(1))$ cycle cover algorithm for $\widetilde{O}(1)$-diameter graphs which runs in $\widetilde{O}(1)$* CONGEST *rounds. Then, there exist deterministic $(1+\varepsilon)$-SSSP and $(1+\varepsilon)$-transshipment algorithms with a round complexity of $\widetilde{O}(\text{ShortcutQuality}(G))$, which is universally-optimal up to polylogarithmic factors.*

While the polylogarithmically tight algorithmic results assumed in Theorem 2.21 seem out of reach of current techniques, our conditional results show that the problem-specific part towards universally optimal shortest path algorithms, even deterministic ones, are essentially fully understood through the techniques of this paper.

REFERENCES

[1] Noga Alon, Zvi Galil, and Oded Margalit. 1997. On the exponent of the all pairs shortest path problem. *J. Comput. System Sci.* 54, 2 (1997), 255–262.

[2] Alexandr Andoni, Clifford Stein, and Peilin Zhong. 2020. Parallel approximate undirected shortest paths via low hop emulators. In *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy (Eds.). ACM, 322–335.

[3] Mikhail Atallah and Uzi Vishkin. 1984. Finding Euler tours in parallel. *J. Comput. System Sci.* 29, 3 (1984), 330–337.

[4] Baruch Awerbuch and David Peleg. 1990. Sparse partitions. In *Proceedings [1990] 31st Annual Symposium on Foundations of Computer Science*. IEEE, 503–513.

[5] Yair Bartal. 2021. Advances in Metric Ramsey Theory and its Applications. *arXiv preprint arXiv:2104.03484* (2021).

[6] Ruben Becker, Yuval Emek, and Christoph Lenzen. 2019. Low diameter graph decompositions by approximate distance computation. *arXiv preprint arXiv:1909.09002* (2019).

[7] Ruben Becker, Andreas Karrenbauer, Sebastian Krinninger, and Christoph Lenzen. 2017. Near-Optimal Approximate Shortest Paths and Transshipment in Distributed and Streaming Models. In *31st International Symposium on Distributed Computing (DISC)*, Vol. 91. 7:1–7:16.

[8] Yi-Jun Chang and Mohsen Ghaffari. 2021. Strong-Diameter Network Decomposition. 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, 273–281. https://doi.org/10.1145/3465084.3467933

[9] Shiri Chechik and Doron Mukhtar. 2020. Single-Source Shortest Paths in the CONGEST Model with Improved Bound. In *Proceedings of the 39th Symposium on Principles of Distributed Computing (PODC)*. 464–473.

[10] Edith Cohen. 1994. Polylog-time and near-linear work approximation scheme for undirected shortest paths. In *Proceedings of the twenty-sixth annual ACM symposium on Theory of Computing*. 16–26.

[11] Edith Cohen. 1995. Approximate max-flow on small depth networks. *SIAM J. Comput.* 24, 3 (1995), 579–597.

[12] Edith Cohen. 1997. Using selective path-doubling for parallel shortest-path computations. *Journal of Algorithms* 22, 1 (1997), 30–56.

[13] Edith Cohen. 1998. Fast Algorithms for Constructing t-Spanners and Paths with Stretch t. *SIAM J. Comput.* 28, 1 (1998), 210–236. https://doi.org/10.1137/S0097539794261295 arXiv:https://doi.org/10.1137/S0097539794261295

[14] Edsger Dijkstra. 1959. A note on two problems in connexion with graphs. *Numerische mathematik* 1, 1 (1959), 269–271.

[15] Michael Elkin. 2004. Distributed approximation: a survey. *ACM SIGACT News* 35, 4 (2004), 40–57.

[16] 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.

[17] Michael Elkin. 2017. Distributed Exact Shortest Paths in Sublinear Time. *Journal of the ACM (JACM)* (2017), 757–770.

[18] Michael Elkin, Bernhard Haeupler, Václav Rozhoň, and Christoph Grunau. [n.d.]. Deterministic Low-Diameter Decompositions for Weighted Graphs and Distributed and Parallel Applications.

[19] Michael Elkin and Shaked Matar. 2021. Deterministic PRAM Approximate Shortest Paths in Polylogarithmic Time and Slightly Super-Linear Work. In *Proceedings of the 33rd ACM Symposium on Parallelism in Algorithms and Architectures* (Virtual Event, USA) *(SPAA '21)*. Association for Computing Machinery, New York, NY, USA, 198–207. https://doi.org/10.1145/3409964.3461809

[20] Michael Elkin and Ofer Neiman. 2016. Hopsets with Constant Hopbound, and Applications to Approximate Shortest Paths. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*. 128–137.

[21] Michael Elkin and Ofer Neiman. 2017. Linear-Size Hopsets with Small Hopbound, and Distributed Routing with Low Memory. arXiv:1704.08468 [cs.DS]

[22] Michael Elkin and Ofer Neiman. 2019. Hopsets with constant hopbound, and applications to approximate shortest paths. *SIAM J. Comput.* 48, 4 (2019), 1436–1480.

[23] Michael Elkin and Ofer Neiman. 2019. Linear-Size Hopsets with Small Hopbound, and Constant-Hopbound Hopsets in RNC. In *The 31st ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*. 333–341.

[24] Sebastian Forster and Danupon Nanongkai. 2018. A Faster Distributed Single-Source Shortest Paths Algorithm. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*. 686–697.

[25] Zvi Galil and Oded Margalit. 1997. All pairs shortest paths for graphs with small integer length edges. *J. Comput. System Sci.* 54, 2 (1997), 243–254.

[26] Mohsen Ghaffari, Christoph Grunau, and Václav Rozhoň. 2021. Improved Deterministic Network Decomposition. In *Proc. of the 32nd ACM-SIAM Symp. on Discrete Algorithms (SODA)*. Society for Industrial and Applied Mathematics, USA, 2904–2923.

[27] Mohsen Ghaffari and Bernhard Haeupler. 2016. Distributed algorithms for planar networks ii: Low-congestion shortcuts, mst, and min-cut. In *Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms (SODA)*. 202–219.

[28] Mohsen Ghaffari and Bernhard Haeupler. 2021. Low-Congestion Shortcuts for Graphs Excluding Dense Minors. 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, 213–221. https://doi.org/10.1145/3465084.3467935

[29] Mohsen Ghaffari, Bernhard Haeupler, and Harald Räcke. 2021. Hop-Constrained Expander Decompositions, Oblivious Routing, and Universally-Optimal Distributed Algorithms. *arXiv preprint* (2021).

[30] Mohsen Ghaffari, Bernhard Haeupler, and Goran Zuzic. 2021. Hop-constrained oblivious routing. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*. 1208–1220.

[31] Mohsen Ghaffari and Jason Li. 2018. Improved distributed algorithms for exact shortest paths. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*. 431–444.

[32] Mohsen Ghaffari and Goran Zuzic. 2022. Universally-Optimal Distributed Exact Min-Cut. *arXiv preprint* (2022).

[33] Bernhard Haeupler, D Ellis Hershkowitz, and David Wajc. 2018. Round- and message-optimal distributed graph algorithms. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing*. 119–128.

[34] Bernhard Haeupler, Taisuke Izumi, and Goran Zuzic. 2016. Low-congestion shortcuts without embedding. In *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing*. 451–460.

[35] Bernhard Haeupler, Taisuke Izumi, and Goran Zuzic. 2016. Near-optimal low-congestion shortcuts on bounded parameter graphs. In *International Symposium on Distributed Computing*. Springer, 158–172.

[36] Bernhard Haeupler and Jason Li. 2018. Faster distributed shortest path approximations via shortcuts. *arXiv preprint arXiv:1802.03671* (2018).

[37] Bernhard Haeupler, Jason Li, and Goran Zuzic. 2018. Minor excluded network families admit fast distributed algorithms. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing*. 465–474.

[38] Bernhard Haeupler, David Wajc, and Goran Zuzic. 2021. Universally-optimal distributed algorithms for known topologies. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*. 1166–1179.

[39] Monika Henzinger, Sebastian Krinninger, and Danupon Nanongkai. 2016. An almost-tight distributed algorithm for computing single-source shortest paths. 2016. In *STOC*, Vol. 16. 2897518–2897638.

[40] Monika Henzinger, Sebastian Krinninger, and Danupon Nanongkai. 2019. A deterministic almost-tight distributed algorithm for approximating single-source shortest paths. *SIAM J. Comput.* (2019), STOC16–98.

[41] Yael Hitron and Merav Parter. 2021. Broadcast CONGEST algorithms against adversarial edges. In *35th International Symposium on Distributed Computing (DISC 2021)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik.

[42] Yael Hitron and Merav Parter. 2021. General CONGEST Compilers against Adversarial Edges. In *35th International Symposium on Distributed Computing (DISC 2021) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 209)*, Seth Gilbert (Ed.). Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 24:1–24:18. https://doi.org/10.4230/LIPIcs.DISC.2021.24

[43] Richard M Karp and Vijaya Ramachandran. 1989. A survey of parallel algorithms for shared-memory machines.

[44] Philip N Klein and Sairam Subramanian. 1997. A randomized parallel algorithm for single-source shortest paths. *Journal of Algorithms* 25, 2 (1997), 205–220.

[45] Shimon Kogan and Merav Parter. 2021. Low-Congestion Shortcuts in Constant Diameter Graphs. *arXiv preprint arXiv:2106.01894* (2021).

[46] Christoph Lenzen and Boaz Patt-Shamir. 2013. Fast routing table construction using small messages. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*. 381–390.

[47] Christoph Lenzen, Boaz Patt-Shamir, and David Peleg. 2019. Distributed distance computation and routing with small messages. *Distributed Computing* 32, 2 (2019), 133–157.

[48] Jason Li. 2020. Faster parallel algorithm for approximate shortest path. In *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy (Eds.). ACM, 308–321.

[49] Danupon Nanongkai. 2014. Distributed approximation algorithms for weighted shortest paths. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing (STOC)*. 565–573.

[50] Merav Parter and Eylon Yogev. 2019. Low Congestion Cycle Covers and Their Applications. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms* (San Diego, California) *(SODA '19)*. Society for Industrial and Applied Mathematics, USA, 1673–1692.

[51] Merav Parter and Eylon Yogev. 2019. Optimal Short Cycle Decomposition in Almost Linear Time. In *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 132)*, Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi (Eds.). Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 89:1–89:14. https://doi.org/10.4230/LIPIcs.ICALP.2019.89

[52] David Peleg. 2000. *Distributed computing: a locality-sensitive approach*. SIAM.

[53] Richard Peng. 2016. Approximate Undirected Maximum Flows in $O(mpolylog(n))$ Time. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 1862–1867.

[54] Václav Rozhoň and Mohsen Ghaffari. 2020. Polylogarithmic-Time Deterministic Network Decomposition and Distributed Derandomization.

[55] 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.

[56] Jonah Sherman. 2013. Nearly Maximum Flows in Nearly Linear Time. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science (FOCS)*. 263–269.

[57] Jonah Sherman. 2017. Area-convexity, ℓ_∞ regularization, and undirected multicommodity flow. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*. 452–460.

[58] Jonah Sherman. 2017. Generalized Preconditioning and Undirected Minimum-Cost Flow. In *Proceedings of the 2017 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 772–780.

[59] Hanmao Shi and Thomas H Spencer. 1999. Time–work tradeoffs of the single-source shortest paths problem. *Journal of algorithms* 30, 1 (1999), 19–32.

[60] Thomas H Spencer. 1997. Time-work tradeoffs for parallel algorithms. *Journal of the ACM (JACM)* 44, 5 (1997), 742–778.

[61] Jeffrey D Ullman and Mihalis Yannakakis. 1991. High-probability parallel transitive-closure algorithms. *SIAM J. Comput.* 20, 1 (1991), 100–125.

[62] Peilin Zhong. 2021. *New Primitives for Tackling Graph Problems and Their Applications in Parallel Computing*. Ph.D. Dissertation. Columbia University.

[63] Goran Zuzic. 2021. A Simple Boosting Framework for Transshipment. *arXiv preprint arXiv:2110.11723* (2021).

[64] Goran Zuzic, Goramoz Goranci, Mingquan Ye, Bernhard Haeupler, and Xiaorui Sun. 2022. Universally-Optimal Distributed Shortest Paths and Transshipment via Graph-Based L1-Oblivious Routing. In *Proceedings of the 33rd Annual ACM-SIAM*

Symposium on Discrete Algorithms (SODA). SIAM.
[65] Uri Zwick. 1998. All pairs shortest paths in weighted directed graphs-exact and almost exact algorithms. In *Proceedings 39th Annual Symposium on Foundations of Computer Science (Cat. No. 98CB36280)*. IEEE, 310–319.