Near-Optimal Approximate Shortest Paths and Transshipment in Distributed and Streaming Models*

Ruben Becker[1], Andreas Karrenbauer[1], Sebastian Krinninger[2], and Christoph Lenzen[1]

1 **Max Planck Institute for Informatics, Saarland Informatics Campus, Saarbrücken, Germany**
`first.last@mpi-inf.mpg.de`

2 **University of Vienna, Faculty of Computer Science, Vienna, Austria**
`sebastian.krinninger@univie.ac.at`

Abstract

We present a method for solving the *shortest transshipment* problem—also known as uncapacitated minimum cost flow—up to a multiplicative error of $1 + \varepsilon$ in undirected graphs with non-negative integer edge weights using a tailored gradient descent algorithm. Our gradient descent algorithm takes $\varepsilon^{-3} \operatorname{polylog} n$ iterations, and in each iteration it needs to solve an instance of the transshipment problem up to a multiplicative error of $\operatorname{polylog} n$, where n is the number of nodes. In particular, this allows us to perform a single iteration by computing a solution on a sparse spanner of logarithmic stretch. Using a careful white-box analysis, we can further extend the method to finding approximate solutions for the single-source shortest paths (SSSP) problem. As a consequence, we improve prior work by obtaining the following results:

1. *Broadcast CONGEST model:* $(1+\varepsilon)$-approximate SSSP using $\tilde{O}((\sqrt{n}+D) \cdot \varepsilon^{-O(1)})$ rounds,[1] where D is the (hop) diameter of the network.
2. *Broadcast congested clique model:* $(1+\varepsilon)$-approximate shortest transshipment and SSSP using $\tilde{O}(\varepsilon^{-O(1)})$ rounds.
3. *Multipass streaming model:* $(1+\varepsilon)$-approximate shortest transshipment and SSSP using $\tilde{O}(n)$ space and $\tilde{O}(\varepsilon^{-O(1)})$ passes.

The previously fastest SSSP algorithms for these models leverage sparse hop sets. We bypass the hop set construction; computing a spanner is sufficient with our method. The above bounds assume non-negative integer edge weights that are polynomially bounded in n; for general non-negative weights, running times scale with the logarithm of the maximum ratio between non-zero weights. In case of asymmetric costs for traversing an edge in opposite directions, running times scale with the maximum ratio between the costs of both directions over all edges.

1998 ACM Subject Classification I.1.2 Algorithms, G.1.6 Optimization

Keywords and phrases Shortest Paths, Shortest Transshipment, Undirected Min-cost Flow, Gradient Descent, Spanner

Digital Object Identifier 10.4230/LIPIcs.DISC.2017.7

* Full version available at `https://arxiv.org/abs/1607.05127`.

[1] We use $\tilde{O}(\cdot)$ to hide polylogarithmic factors in n.

© Ruben Becker, Andreas Karrenbauer, Sebastian Krinninger, and Christoph Lenzen;
licensed under Creative Commons License CC-BY

1 Introduction

Single-source shortest paths (SSSP) is a fundamental and well-studied problem in computer science. Thanks to sophisticated algorithms and data structures [20, 23, 41], it has been known for a long time how to obtain (near-)optimal running time in the RAM model. This is not the case in non-centralized models of computation, which become more and more relevant in a big-data world. Despite certain progress for *exact* SSSP algorithms [6, 7, 9, 15, 28, 30, 39, 40], there remain large gaps to the strongest known lower bounds. Close-to-optimal running times have so far only been achieved by efficient approximation schemes [10, 17, 25, 32]. For instance, in the CONGEST model of distributed computing, the state of the art is as follows: Exact SSSP on weighted graphs can be computed in $O(D^{1/3}(n \log n)^{2/3})$ rounds [15], where D is the (hop) diameter of the graph, and $(1+\varepsilon)$-approximate SSSP can be computed in $(\sqrt{n}+D) \cdot 2^{O(\sqrt{\log n \log(\varepsilon^{-1} \log n)})}$ rounds [25].[2] Even for constant ε, the latter exceeds the strongest known lower bound of $\Omega(\sqrt{n}/\log n + D)$ rounds [13] by a super-polylogarithmic factor. As a consequence of the techniques developed in this paper, we make a qualitative algorithmic improvement for $(1+\varepsilon)$-approximate SSSP in this model: we solve the problem in $(\sqrt{n}+D) \cdot \varepsilon^{-O(1)}$ polylog n rounds. We thus narrow the gap between upper and lower bound significantly and additionally improve the dependence on ε. Our new approach achieves its superior running time by leveraging techniques from continuous optimization.

It is inherent to our approach that we actually tackle a problem that seems more general than SSSP. In the *shortest transshipment* problem, we seek to find a cheapest routing for sending units of a single good from sources to sinks along the edges of a graph meeting the nodes' demands. Equivalently, we want to find the minimum-cost flow in a graph where edges have unlimited capacity. The special case of SSSP can be modeled as a shortest transshipment problem by setting the demand of the source to $-n+1$ (thus supplying $-n+1$ units) and the demand of every other node to 1. Unfortunately, this relation breaks when we consider approximation schemes: A $(1+\varepsilon)$-approximate solution to the transshipment problem merely yields $(1+\varepsilon)$-approximations to the distances *on average*. In the special case of SSSP, however, one is interested in obtaining a $(1+\varepsilon)$-approximation to the distance for *each single node* and we show how to extend our algorithm to provide such a guarantee as well.

Techniques from continuous optimization have been key to recent breakthroughs in the combinatorial realm of graph algorithms [8, 11, 12, 27, 31, 33, 37]. In this paper, we apply this paradigm to computing primal and dual $(1+\varepsilon)$-approximate solutions to the shortest transshipment problem in undirected graphs with non-negative edge weights. Accordingly, we perform projected gradient descent for a suitable norm-minimization formulation of the problem, where we approximate the infinity norm by a differentiable soft-max function. To make this general approach work in our setting, we need to add significant problem-specific tweaks. In particular, we develop a gradient descent algorithm that reduces the problem of computing a $(1+\varepsilon)$-approximation to the more relaxed problem of computing, e.g., an $O(\log n)$-approximation. We then exploit that an $O(\log n)$-approximation can be computed very efficiently by solving the problem on a sparse spanner, and that it is well-known how to compute sparse spanners efficiently. To obtain the aforementioned per-node guarantee in the approximate SSSP problem, we additionally exploit specific properties of our gradient descent algorithm. Further effort is required to extract an approximate shortest-path tree

[2] Note that these running times refer to weighted graphs. In unweighted graphs, the SSSP problem can easily be solved in $O(D)$ rounds by performing a BFS tree computation.

(i.e., a primal solution) from the dual solution (i.e., estimated distances to the source).

Our method is widely applicable among a plurality of non-centralized models of computation in a rather straightforward way. We obtain the first non-trivial algorithms for approximate undirected shortest transshipment in the broadcast CONGEST,[3] broadcast congested clique, and multipass streaming models. As a further, arguably more important, consequence, we improve upon prior results for computing approximate SSSP in these models. Our approximate SSSP algorithms are the first to be provably optimal up to polylogarithmic factors.

Our Contributions and Results. We summarize our technical and conceptual contributions as follows:

(C1) We give a problem-specific gradient descent algorithm for approximating the shortest transshipment, which requires access to an oracle computing an α-approximate dual solution for any given demand vector.[4] To compute a $(1+\varepsilon)$-approximation, the algorithm performs $\tilde{O}(\varepsilon^{-3}\alpha^2)$ oracle calls. If the oracle returns primal solutions, so does our algorithm.

(C2) We provide an additional analysis of the gradient descent algorithm that allows us to extend the method to solving SSSP in order to achieve a per-node approximation guarantee.

(C3) We observe that spanners can be used to obtain an efficient shortest transshipment oracle with approximation guarantee $\alpha \in O(\log n)$.

By implementing our method in specific models of computation, we obtain the following concrete algorithmic results in graphs with non-negative polynomially bounded[5] integer edge weights:

(R1) We give faster algorithms for computing $(1+\varepsilon)$-approximate SSSP:

a. *Broadcast CONGEST model:* We obtain a deterministic algorithm for computing $(1+\varepsilon)$-approximate SSSP using $\tilde{O}((\sqrt{n}+D)\cdot\varepsilon^{-O(1)})$ rounds. This improves upon the previous best upper bound of $(\sqrt{n}+D)\cdot 2^{O(\sqrt{\log n \log(\varepsilon^{-1}\log n)})}$ rounds [25]. For $\varepsilon^{-1} \in O(\operatorname{polylog} n)$, we match the lower bound of $\Omega(\sqrt{n}/\log n + D)$ [13] (applying to any (randomized) $(\operatorname{poly} n)$-approximation of the distance between two fixed nodes in a weighted undirected graph) up to polylogarithmic factors in n.

b. *Broadcast congested clique model:* We obtain a deterministic algorithm for computing $(1+\varepsilon)$-approximate SSSP using $\tilde{O}(\varepsilon^{-O(1)})$ rounds. This improves upon the previous best upper bound of $2^{O(\sqrt{\log n \log(\varepsilon^{-1}\log n)})}$ rounds [25].

c. *Multipass streaming model:* We obtain a deterministic algorithm for computing $(1+\varepsilon)$-approximate SSSP using $\tilde{O}(\varepsilon^{-O(1)})$ passes and $O(n\log n)$ space. This improves upon the previous best upper bound of $(2+1/\varepsilon)^{O(\sqrt{\log n \log\log n})}$ passes and $O(n\log^2 n)$ space [17]. By setting ε small enough, we can compute distances up to the value $\log n$ exactly in integer-weighted graphs using $\operatorname{polylog} n$ passes and $O(n\log n)$ space. Thus, up to polylogarithmic factors in n, our result matches a lower bound of $n^{1+\Omega(1/p)}/\operatorname{poly} p$ space for all algorithms that decide in p passes if the distance between two fixed nodes in an unweighted undirected graph is at most $2(p+1)$ for any $p = O(\log n/\log\log n)$ [22].

---

3 Also known as the node-CONGEST model.

4 Note that dual feasibility is crucial here. In particular, this rules out an oracle based on tree embeddings [2, 16], as such trees might have stretch $\Omega(n)$ on individual edges.

5 For general non-negative weights, running times scale by a multiplicative factor of $\log R$, where R is the maximum ratio between non-zero edge weights.

(R2) We give fast algorithms for computing $(1+\varepsilon)$-approximate shortest transshipment:
  a. *Broadcast CONGEST model:* A deterministic algorithm using $\tilde{O}(\varepsilon^{-3}n)$ rounds.
  b. *Broadcast congested clique model:* A deterministic algorithm using $\tilde{O}(\varepsilon^{-3})$ rounds.
  c. *Multipass streaming model:* A deterministic algorithm using $\tilde{O}(\varepsilon^{-3})$ passes and $O(n \log n)$ space.

No non-trivial upper bounds were known before in these three models.

In the case of SSSP, we can deterministically compute a $(1+\varepsilon)$-approximation to the distance from the source for every node. Using a *randomized* procedure, we can additionally compute (with high probability within the same asymptotic running times) a tree on which every node has a path to the source that is within a factor of $(1+\varepsilon)$ of its true distance.

In the case of shortest transshipment, we can (deterministically) return $(1+\varepsilon)$-approximate primal and dual solutions. We can further extend the results to asymmetric weights on undirected edges, where each edge can be used in either direction at potentially different costs. Denoting by $\lambda \geq 1$ the maximum over all edges of the cost ratio between traversing the edge in different directions, our algorithms give the same guarantees if the number of rounds or passes, respectively, is increased by a factor of $\lambda^4 \log \lambda$.

Related Work on Shortest Transshipment. Shortest transshipment is a classic problem in combinatorial optimization [29, 36]. The classic algorithms for directed graphs with non-negative edge weights in the RAM model run in time $O(n(m + n\log n)\log n)$ [35] and $O((m + n\log n)B)$ [14], respectively, where B is the sum of the nodes' demands (when they are given as integers) and the term $m + n\log n$ comes from SSSP computations. If the graph contains negative edge weights, then these algorithms require an additional preprocessing step to compute SSSP in presence of negative edge weights, for example in time $O(mn)$ using the Bellman-Ford algorithm [4, 19] or in time $O(m\sqrt{n}\log N)$ using Goldberg's algorithm [21].[6] The weakly polynomial running time was first improved to $\tilde{O}(m^{3/2}\operatorname{polylog} R)$ [12] and then to $\tilde{O}(m\sqrt{n}\operatorname{polylog} R)$ in a recent breakthrough for minimum-cost flow [31], where R is the ratio between the largest and the smallest edge weight. Independent of our work, Sherman [38] obtained a randomized algorithm for computing a $(1+\varepsilon)$-approximate shortest transshipment in weighted undirected graphs in time $O(\varepsilon^{-2}m^{1+o(1)})$ using a generalized-preconditioning approach. We refer the reader to the full paper for a detailed comparison of Sherman's and our approach. We are not aware of any non-trivial algorithms for computing (approximate) shortest transshipment in non-centralized models of computation, such as distributed and streaming models.

Comparison to Hop Set Based SSSP Algorithms. The state-of-the art SSSP algorithms in the distributed CONGEST model follow the framework developed in [34], where (1) the problem of computing SSSP is reduced to an overlay network of size $N = \tilde{O}(\sqrt{n})$ and (2) a sparse hop set is constructed to speed up computing SSSP on the overlay network. An (h,ε)-hop set is a set of weighted edges that, when added to the original graph, provides sufficient shortcuts to approximate all pairwise distances using paths with only h edges ("hops"). In the algorithm by Nanongkai et al. [25], the upper bound of $(\sqrt{n}+D)\cdot 2^{O(\sqrt{\log n \log(\varepsilon^{-1}\log n)})}$ on the number of rounds is achieved by constructing an (h,ε)-hop set of size $O(N\rho)$ where $h \leq 2^{O(\sqrt{\log n \log(\varepsilon^{-1}\log n)})}$ and $\rho \leq 2^{O(\sqrt{\log n \log(\varepsilon^{-1}\log n)})}$. Elkin's algorithm [15], which takes $O(D^{1/3}(n\log n)^{2/3})$ rounds, uses an exact $(N/\rho, 0)$ hop set of size $O(N\rho)$ similar to

[6] Goldberg's running time bound holds for integer-weighted graphs with most negative weight $-N$.

the one developed by Shi and Spencer in a PRAM algorithm [40]. Elkin's main technical contribution lies in showing how to compute this hop set without constructing the overlay network explicitly. Roughly speaking, in these algorithms, *both* h and ρ enter the running time of the corresponding SSSP algorithms, in addition to the time needed to construct the hop set.

The concept of hop sets has been introduced by Cohen in the context of PRAM algorithms for approximate SSSP [10]. The increased interest in hop sets and their applications in the last years [5, 17, 24, 25, 32] has culminated in the construction of (h, ε)-hop sets of size $O(n^{1+\frac{1}{2^k+1}-1})$ for $h = O\left((\frac{k}{\varepsilon})^k\right)$ [18, 26]. Recent lower bounds by Abboud et al. [1] show that this trade-off is essentially tight: any construction of (h, ε)-hop sets of size $\leq n^{1+\frac{1}{2^k-1}-\delta}$ must have $h = \Omega_k\left((\frac{1}{\varepsilon})^k\right)$ (where $k \geq 1$ is an integer and $\delta > 0$). This implies that the hop set based algorithms, as long as the factor ρ has to be paid in the running time for construction hop sets of size $n\rho$, will never be able to achieve a running time comparable to our SSSP algorithm exclusively by finding better hop sets.

Spanners. In our approach we use a spanner to obtain an efficient shortest transshipment oracle.

▶ **Definition 1** (Spanner). Given $G = (V, E, w)$ and $\alpha \geq 1$, an *α-spanner* of G is a subgraph $(V, E', w|_{E'})$, $E' \subseteq E$, in which distances are at most by factor α larger than in G.

In other words, a spanner removes edges from G while approximately preserving distances. It is well-known that for every undirected graph we can efficiently compute an α-spanner of size $O(n \log n)$ with $\alpha = O(\log n)$ [3].

Structure of this paper. In the following section, we will first describe the gradient descent algorithm for the case of symmetric weights. More precisely, we will describe how to obtain a primal/dual solution pair of approximation ratio $(1+\varepsilon)$ for an oracle yielding both primal and dual solutions; if the oracle provides dual solutions only, so do our algorithms. In Section 2.2, we describe how to obtain $(1+\varepsilon)$-approximate distances for *every* node in the SSSP case. In Section 2.3, we show how to obtain a $(1+\varepsilon)$-approximate primal *tree* solution. In Section 3, we briefly describe how the above framework can be implemented in various distributed and streaming models of computation. Due to space limitations, we refer to the full paper for further details.

The full version also discusses how our techniques can be generalized to asymmetric edge weights. The key observation is that, essentially, the gradient descent algorithm can be guided by basing the oracle on solving the symmetrized variant problem on an (undirected) spanner. The additional inaccuracy of the approximation slows down the progress of the algorithm by a factor of $\lambda^4 \log \lambda$. However, while this generalization does not affect our approach structurally, some technical obstacles need to be overcome. For the sake of a streamlined presentation, we thus confine the discussion to the symmetric problem.

2 General Approach for Solving Shortest Transshipment and SSSP

Let $G = (V, E)$ be a (w.l.o.g. connected) undirected graph with n nodes, m edges, and positive[7] integral edge weights $w \in \mathbb{Z}^m_{\geq 1}$. Furthermore, let $b \in \mathbb{Z}^n$ be a vector of demands.

[7] Note that excluding 0 as an edge weight is a only a mild restriction, because we can always generate new weights w' with $w'_e = 1 + \lceil n/\varepsilon \rceil \cdot w_e$ while preserving at least one of the shortest paths between

W.l.o.g., we restrict to feasible and non-trivial instances, i.e., $b^T\mathbb{1} = 0$ and $b \neq 0$.[8] A common approach to model the *undirected shortest transshipment* problem as a linear program considers the node-arc-incidence matrix $A \in \{-1, 0, 1\}^{n\times 2m}$ of the corresponding bidirected graph,[9] where we substitute each edge e by a forward and a backward arc with the same weight w_e in both directions. While this may seem redundant, it is convenient in terms of notation and generalizes to the case of asymmetric edge weights considered in the full paper. With W being the $2m \times 2m$ diagonal matrix containing the weights, we obtain the following primal/dual pair of linear programs:

$$\min\{\|Wx\|_1 : Ax = b\} = \max\{b^T y : \|W^{-1}A^T y\|_\infty \leq 1\}, \tag{1}$$

where for $z \in \mathbb{R}^d$ we write $\|z\|_1 = \sum_{i=1}^d |z_i|$ and $\|z\|_\infty = \max_{i\in[d]}\{|z_i|\}$. The primal (left) program asks to "ship" the flow given by b from sources (negative demand) to sinks (positive demand) along the edges of the graph, minimizing the cost of the flow, i.e., $\sum_{e\in E} w_e|x_e|$. Note that, without changing that $Ax = b$ or affecting the objective, we can remove "negative" flow $x_{vw} < 0$ by increasing x_{wv} by $|x_{vw}|$ and setting $x_{vw} = 0$. Thus, w.l.o.g., we may assume that $x \geq 0$; in particular, an optimal solution sends flow only in one direction over any edge.

The dual (right) program asks for potentials y such that for each edge $e = (v, w) \in E$, $|y_v - y_w| \leq w_e$, maximizing $b^T y$. Note that, because $b^T\mathbb{1} = 0$, shifting the potential by $r \times \mathbb{1}$ for any $r \in \mathbb{R}$ does neither change $b^T y$ nor $y_v - y_w$ for any $v, w \in V$. The goal of the dual is thus to maximize the differences in potential of sources and sinks (weighted according to b), subject to the constraint that the potentials of neighbors must not differ by more than the weight of their connecting edge.

In the special case of SSSP with source $s \in V$, we have that (i) $b_s = -n + 1$ and $b_v = 1$ for all $v \neq s$, (ii) an optimal primal solution x^* is given by routing, for each $s \neq v \in V$, one unit of flow along a shortest path from s to v, and (iii) optimal potentials y^* are given by setting y^*_v to the distance from s to v.

2.1 Gradient Descent

We now describe a gradient descent method that, given an oracle that computes α-*approximate* primal and dual solutions to the undirected shortest transshipment problem for any specified demand vector $\tilde{b}$, returns primal and dual feasible solutions x and y to the undirected shortest transshipment problem that are $(1+\varepsilon)$-close to optimal, i.e., fulfill $\|Wx\|_1 \leq (1+\varepsilon)b^T y$, using $O(\varepsilon^{-3}\alpha^2 \log\alpha \log n)$ calls to the oracle. We then provide an oracle with $\alpha \in \operatorname{polylog}(n)$. For ease of notation, we assume that $\log\alpha \in \operatorname{polylog} n$ throughout this paper.

As our first step, we relate the dual of the shortest transshipment problem to its "reciprocal" linear program that normalizes the objective to 1 and seeks to minimize $\|W^{-1}A^T y\|_\infty$:

$$\min\{\|W^{-1}A^T\pi\|_\infty : b^T\pi = 1\}. \tag{2}$$

We denote by π^* an optimal solution to this problem, whereas y^* denotes an optimal solution to the dual of the original problem (1). It is easy to see that feasible solutions π of (2) that

each pair of nodes as well as $(1+\varepsilon)$-approximations. As we assume edge weights to be integer we can assume that $\varepsilon \geq 1/(n\|w\|_\infty)$ (as otherwise it is required to compute an exact solution) and thus our asymptotic running time bounds are not affected by this modification.

[8] Here $\mathbb{1}$ denotes the all-ones vector and thus $b^T\mathbb{1} = 0$ simply means that the positive demands equal the negative demands (i.e., the supplies).

[9] The incidence matrix A of a directed graph contains a row for every node and a column for every arc and $A_{i,j}$ is -1 if the j-th arc leaves the i-th vertex, 1 if it enters the vertex, and 0 otherwise.

satisfy $\|W^{-1}A^T\pi\|_\infty > 0$ are mapped to feasible solutions of the dual program in (1) via $f(\pi) := \pi/\|W^{-1}A^T\pi\|_\infty$. Similarly, feasible solutions y of the dual program in (1) that satisfy $b^Ty > 0$ are mapped to feasible solutions of (2) via $g(y) := y/b^Ty$. Moreover, the map $f(\cdot)$ preserves the approximation ratio. Namely, for any $\varepsilon > 0$, if π is a solution of (2) within factor $1+\varepsilon$ of the optimum, then $f(\pi)$ is feasible for (1) and within factor $1+\varepsilon$ of the optimum. In particular, $f(\pi^*)$ is an optimal solution of (1).

We would like to apply gradient descent to (2). However, this is not readily possible, since the objective is not differentiable. Hence, we will change the problem another time by using the so-called soft-max function (a.k.a. log-sum-exp or lse for short), which is a suitable approximation for the maximum entry $(v)_{\max} := \max\{v_i : i \in [d]\}$ of a vector $v \in \mathbb{R}^d$.[10] It is defined as $\mathrm{lse}_\beta(v) := \frac{1}{\beta}\ln\left(\sum_{i\in[d]} e^{\beta v_i}\right)$, where $\beta > 0$ is a parameter that controls the accuracy of the approximation of the maximum at the expense of smoothness. We note that $\mathrm{lse}_\beta(\cdot)$ is a convex function for any $\beta > 0$ and provides the following additive approximation of the maximum:

$$(x)_{\max} = \frac{1}{\beta}\ln e^{\beta\cdot(x)_{\max}} \le \mathrm{lse}_\beta(x) \le \frac{1}{\beta}\ln\sum_{i\in[d]} e^{\beta\cdot(x)_{\max}} = \frac{\ln(d)}{\beta} + (x)_{\max}. \tag{3}$$

A trade-off in the choice of β arises because β also controls the smoothness of the lse-function. Formally, lse_β is β-Lipschitz smooth (i.e., its gradient is β-Lipschitz continuous) w.r.t. to the pair 1-norm/∞-norm:

$$\|\Phi_\beta(x) - \Phi_\beta(y)\|_1 \le \beta\|x-y\|_\infty. \tag{4}$$

Using the soft-max function, we define the *potential function*

$$\Phi_\beta(\pi) := \mathrm{lse}_\beta\left(W^{-1}A^T\pi\right).$$

Recalling that A was defined to represent each edge of the graph by a forward and backward arc, we see that $(W^{-1}A^T\pi)_{\max} = \|W^{-1}A^T\pi\|_\infty$, i.e., $\Phi_\beta(\pi)$ is indeed a smooth approximation of the objective of (2). In order to control the approximation error, β is adapted in the course of the algorithm such that the additive error $\ln(2m)/\beta$ is always at most $\frac{\varepsilon}{4}\Phi_\beta(\pi)$. Thus, we maintain a multiplicative approximation of the dual objective function of (2), i.e.,

$$\|W^{-1}A^T\pi\|_\infty \le \Phi_\beta(\pi) \le \frac{\|W^{-1}A^T\pi\|_\infty}{1-\varepsilon/4}. \tag{5}$$

Our gradient descent algorithm, see Algorithm 1 for a pseudo-code implementation, first computes a starting solution π that is an α-approximate (dual) solution to (2) and an initial β that is appropriate for π as discussed above. This can be done, e.g., by solving the problem on an α-spanner and scaling down (by at most a factor of α) to obtain a feasible solution for the original graph. In each iteration, it updates the potentials π using an α-approximate solution to a shortest transshipment problem with a modified demand vector $\tilde{b}$ that depends on the gradient. Depending on the objective value of this approximation, the algorithm either performs an update to π or terminates, see the check for the value of δ in the algorithm.

The intuition behind the algorithm is the following. As the potential function is differentiable, its gradient exists everywhere and it points in the opposite direction of the steepest

[10] Note the difference to the ∞-norm, which is defined as the maximum of the *absolute values* of the entries of a vector.

Algorithm 1: `gradient_transship` (G, b, ε)

Compute α-approximation π to $\min\{\|W^{-1}A^T\pi\|_\infty : b^T\pi = 1\}$. *// use oracle*
Determine β so that $4\ln(2m) \leq \varepsilon\beta\Phi_\beta(\pi) \leq 5\ln(2m)$.
repeat
  Set $\tilde{b} := P^T\nabla\Phi_\beta(\pi)$, where $P := I - \pi b^T$. *// project to maintain $b^T\pi = 1$*
  if $\tilde{b} = 0$ **then return** π *// Special case: optimal solution found*
  Determine $\tilde{h}$ with $\|W^{-1}A^T\tilde{h}\|_\infty = 1$ and $\tilde{b}^T\tilde{h} \geq \frac{1}{\alpha}\max\{\tilde{b}^T h : \|W^{-1}A^T h\|_\infty \leq 1\}$.
  // $\tilde{h}$ can be obtained from the oracle with demand vector $\tilde{b} = P^T\nabla\Phi_\beta(\pi)$
  Set $\delta := \frac{\tilde{b}^T\tilde{h}}{\|W^{-1}A^T P\tilde{h}\|_\infty}$. *// δ measures closeness to optimality*
  if $\delta > \frac{\varepsilon}{8\alpha}$ **then** $\pi \leftarrow \pi - \frac{\delta}{2\beta\|W^{-1}A^T P\tilde{h}\|_\infty}P\tilde{h}$. *// project to maintain $b^T\pi = 1$*
  while $4\ln(2m) \geq \varepsilon\beta\Phi_\beta(\pi)$ **do** $\beta \leftarrow \frac{5}{4}\beta$. *// find appropriate β*
until $\delta \leq \frac{\varepsilon}{8\alpha}$
return π

descent. However, our update steps must maintain the constraint $b^T\pi = 1$, i.e., they must lie in the orthogonal complement of b. To this end, we consider the projection of the gradient $P^T\nabla\Phi_\beta(\pi)$.[11] Because the gradient, and hence the direction of the steepest descent, changes when we move away from our current solution, we use an adaptive step width restricting the update to a region for which we know that the gradient does not vary too much.

If we had a sufficiently good guarantee on the Lipschitz smoothness of $\Phi_\beta(\cdot)$, using the gradient itself (resp. its projection) as the update direction h (i.e., performing the update $\pi \leftarrow \pi - \eta h$ for an appropriate step width η) would decrease the objective $\Phi_\beta(\pi)$ fast enough. However, we only have such a guarantee on the Lipschitz smoothness of $\mathrm{lse}_\beta(\cdot)$. By the convexity of the objective $\Phi_\beta(\pi)$, we can argue that the (normalized) progress of an update direction h is the ratio $P^T\nabla\Phi_\beta(\pi)/\|W^{-1}A^T h\|_\infty$, which suggests finding h by $\max\{\nabla\Phi_\beta(\pi)^T Ph : \|W^{-1}A^T h\|_\infty \leq 1\}$. Note that this linear program is precisely of the form (1), with demand vector $\tilde{b} := P^T\nabla\Phi_\beta(\pi)$, and is thus not easier to solve as the original problem. However, finding an approximately optimal update direction only mildly affects the number of iterations, i.e., querying the oracle for an α-approximate dual solution with demand $\tilde{b}$ yields the desired guarantee.

We then use the projection P to derive a feasible update and rescale so that the gradient does not change too much, enabling us to prove a sufficiently strong progress guarantee – unless the current solution is already close to the optimum. This is captured by δ, which is guaranteed to be large in case significant progress still can be made. Conversely, a small δ implies that we are close to the optimum. Accordingly, at termination π is a near-optimal solution of (2), and rescaling to $y = \pi/\|W^{-1}A^T\pi\|_\infty$ yields a near-optimal dual solution of (1). Here, scaling up β as the potential decreases ensures that the incurred approximation error is sufficiently small. On the other hand, using large β and having the guarantee that δ is large as long as we are not close to the optimum guarantees that the potential function decreases rapidly and only a small number of iterations is required.

We proceed by formalizing this intuition. First, we show that a primal-dual pair that is $(1+\varepsilon)$-close to optimal in (1) can be constructed from the output potentials π and α-approximate primal and dual solutions, say $\tilde{f}$ and $\tilde{h}$, to the transshipment problem that

[11] As $b^T\pi = 1$, we have that $b^T Ph = b^T(I - \pi b^T)h = b^T h - b^T\pi b^T h = 0$ for all h.

was solved in the last iteration of the algorithm. If one is only interested in a dual solution to (1), then the α-approximate dual solution $\tilde{h}$ is enough and thus only an oracle providing a dual solution is required, as done in Algorithm 1. A primal solution can be obtained from $\tilde{f}$ and the vector $\tilde{x} := W^{-1}\nabla \operatorname{lse}_\beta\left(W^{-1}A^T\pi\right)$. This choice of $\tilde{x}$ is obtained by applying the chain rule of differentiation to $\nabla\Phi_\beta(\pi)$, i.e., $\nabla\Phi_\beta(\pi) = AW^{-1}\nabla \operatorname{lse}_\beta\left(W^{-1}A^T\pi\right)$. In the correctness proof we allow a more general choice of $\tilde{x}$, which we will exploit later on for finding a tree solution for approximate SSSP.

▶ **Lemma 2** (Correctness). *Let $0 < \varepsilon \leq 1/2$,*

- *$\pi \in \mathbb{R}^n$ and $\beta \in \mathbb{R}$ denote the return values of Algorithm* **`gradient_transship`**,
- *$\tilde{f} \in \mathbb{R}^{2m}$ and $\tilde{h} \in \mathbb{R}^n$ be the α-approximate pair returned by the oracle in the last iteration of Algorithm* **`gradient_transship`**, *and*
- *$\tilde{x} \in \mathbb{R}^{2m}$ be such that $A\tilde{x} = \nabla\Phi_\beta(\pi)$ and $\|W\tilde{x}\|_1 \leq 1 + \varepsilon/8$.*

Then $x := \frac{\tilde{x}-\tilde{f}}{\pi^T\nabla\Phi_\beta(\pi)}, y := \frac{\pi}{\|W^{-1}A^T\pi\|_\infty}$ is a $(1+\varepsilon)$-approximate pair, i.e., it holds that $Ax = b$, $\|W^{-1}A^Ty\|_\infty \leq 1$, and $\|Wx\|_1 \leq (1+\varepsilon)b^Ty$.

Proof. First note that $A\tilde{f} = \tilde{b}$ and $\tilde{b} = P^T\nabla\Phi_\beta(\pi) = \nabla\Phi_\beta(\pi) - b\pi^T\nabla\Phi_\beta(\pi)$. Thus $Ax = \frac{\nabla\Phi_\beta(\pi)-\tilde{b}}{\pi^T\nabla\Phi_\beta(\pi)} = b$. Moreover, $\|W^{-1}A^Ty\|_\infty = 1$ follows directly from the definition of y.

It remains to show that $\|Wx\|_1 \leq (1+\varepsilon)b^Ty$.[12] It can be shown (see full paper for details) that convexity of $\Phi_\beta(\cdot)$ and the guarantee on β yield

$$\pi^T\nabla\Phi_\beta(\pi) \geq \left(1 - \frac{\varepsilon}{4}\right)\Phi_\beta(\pi) \geq \left(1 - \frac{\varepsilon}{4}\right)\|W^{-1}A^T\pi\|_\infty > 0. \tag{6}$$

Hence, $|\pi^T\nabla\Phi_\beta(\pi)| = \pi^T\nabla\Phi_\beta(\pi)$. Moreover, $\|W\tilde{x}\|_1 \leq 1 + \varepsilon/8$ by assumption and thus

$$\|Wx\|_1 \overset{\Delta\text{-ineq.}}{\leq} \frac{1 + \frac{\varepsilon}{8} + \|W\tilde{f}\|_1}{\pi^T\nabla\Phi_\beta(\pi)} \overset{\alpha\text{-approx.}}{\leq} \frac{1 + \frac{\varepsilon}{8} + \alpha\tilde{b}^T\tilde{h}}{\pi^T\nabla\Phi_\beta(\pi)} = \frac{1 + \frac{\varepsilon}{8} + \alpha\delta\|W^{-1}A^TP\tilde{h}\|_\infty}{\pi^T\nabla\Phi_\beta(\pi)},$$

where $\delta = \frac{\tilde{b}^T\tilde{h}}{\|W^{-1}A^TP\tilde{h}\|_\infty}$ as in Algorithm `gradient_transship`. By the definition of $P = I - \pi b^T$ and the triangle inequality for the infinity norm, we obtain $\|W^{-1}A^TP\tilde{h}\|_\infty \leq \|W^{-1}A^T\tilde{h}\|_\infty + |b^T\tilde{h}|\|W^{-1}A^T\pi\|_\infty$. Using the upper bound $|b^T\tilde{h}| \leq b^Ty^*$ from the optimality of y^* and $\|W^{-1}A^T\tilde{h}\|_\infty \leq 1$, we obtain $\|W^{-1}A^TP\tilde{h}\|_\infty \leq 1 + \|W^{-1}A^T\pi\|_\infty b^Ty^*$. Using (6) for the denominator, this yields

$$\|Wx\|_1 \leq \frac{1 + \frac{\varepsilon}{8} + \alpha\delta(1 + \|W^{-1}A^T\pi\|_\infty b^Ty^*)}{(1-\frac{\varepsilon}{4})\|W^{-1}A^T\pi\|_\infty} \leq \frac{1 + \frac{\varepsilon}{8} + \frac{\varepsilon}{8}(1 + \frac{\|Wx\|_1}{b^Ty})}{(1-\frac{\varepsilon}{4})}b^Ty,$$

since $b^Ty^* \leq \|Wx\|_1$ by weak duality, $\|W^{-1}A^T\pi\|_\infty = 1/b^Ty$, and $\delta \leq \frac{\varepsilon}{8\alpha}$ at termination of the algorithm. Thus $(1+\frac{\varepsilon}{4})/(1-\frac{3\varepsilon}{8}) \leq (1+\frac{\varepsilon}{4})/(1-\frac{\varepsilon}{2}) \leq (1+\varepsilon)$ yields the result. ◀

It remains to show a bound on the number of iterations until termination. To this end, we establish that the potential function decreases by a multiplicative factor in each iteration.

▶ **Lemma 3** (Multiplicative Decrement of Φ_β). *Let $\pi \in \mathbb{R}^n$, let β satisfy $\varepsilon\beta\Phi_\beta(\pi) \leq 5\ln(2m)$, and let $\tilde{h}$ satisfy $\|W^{-1}A^TP\tilde{h}\|_\infty > 0$, where $P = I - \pi b^T$. Then, for $\delta := \frac{\tilde{b}^T\tilde{h}}{\|W^{-1}A^TP\tilde{h}\|_\infty}$, where $\tilde{b} = P^T\nabla\Phi_\beta(\pi)$, it holds that*

$$\Phi_\beta\left(\pi - \frac{\delta}{2\beta\|W^{-1}A^TP\tilde{h}\|_\infty}P\tilde{h}\right) \leq \left(1 - \frac{\varepsilon\delta^2}{20\ln(2m)}\right)\Phi_\beta(\pi).$$

[12] Here, we omit the special case $\tilde{b} = 0$, which guarantees optimality. See full version for details.

Proof. Let us denote $h := \frac{\delta}{2\beta\|W^{-1}A^TP\tilde{h}\|_\infty}\tilde{h}$. Recall that $\Phi_\beta(\cdot)$ is convex, thus

$$\begin{aligned}
\Phi_\beta(\pi - Ph) - \Phi_\beta(\pi) &\overset{\text{Convexity}}{\leq} -\nabla\Phi_\beta(\pi - Ph)^T Ph + \nabla\Phi_\beta(\pi)^T Ph - \nabla\Phi_\beta(\pi)^T Ph \\
&= \left[\nabla \operatorname{lse}_\beta\left(W^{-1}A^T\pi\right) - \nabla \operatorname{lse}_\beta\left(W^{-1}A^T(\pi - Ph)\right)\right]^T W^{-1}A^TPh - \tilde{b}^T h \\
&\overset{\text{Hölder}}{\leq} \|\nabla \operatorname{lse}_\beta\left(W^{-1}A^T\pi\right) - \nabla \operatorname{lse}_\beta\left(W^{-1}A^T(\pi - Ph)\right)\|_1 \|W^{-1}A^TPh\|_\infty - \tilde{b}^T h \\
&\overset{\beta\text{-Lipschitz}}{\leq} \beta\|W^{-1}A^TPh\|_\infty^2 - \tilde{b}^T h,
\end{aligned}$$

where we used Hölder's inequality[13] and then the fact that the lse_β-function is β-Lipschitz smooth (see (4)). Using the definitions of h and δ yields $\Phi_\beta(\pi - Ph) - \Phi_\beta(\pi) \leq \frac{\delta^2}{4\beta} - \frac{\delta^2}{2\beta} = -\frac{\delta^2}{4\beta}$. Using the upper bound on β yields the result. ◀

This progress guarantee is sufficient to show the following bound on the number of iterations.

▶ **Lemma 4** (Number of Iterations). *Suppose that $0 < \varepsilon \leq 1/2$. Then, it holds that Algorithm* `gradient_transship` *terminates within $O(\varepsilon^{-3}\alpha^2 \log\alpha \log n)$ iterations.*

Proof. Note that for all $x \in \mathbb{R}^n$, $\nabla_\beta \operatorname{lse}_\beta(x) \leq 0$, i.e., lse_β is decreasing as a function of β and thus the while-loop that scales β up does not increase $\Phi_\beta(\pi)$. Denote by β_0 and π_0 the initial values of β and π, respectively, and by β and π the values at termination. By Lemma 3 and the fact that the algorithm ensures $\delta > \varepsilon/(8\alpha)$ as long as it does not terminate, the potential decreases by a factor of $1 - \frac{\varepsilon\delta^2}{20\ln(2m)} \leq 1 - \frac{\varepsilon^3}{1280\alpha^2\ln(2m)}$.[14] Hence, the number of iterations k can be bounded by

$$k \leq \log\left(\frac{\Phi_\beta(\pi)}{\Phi_{\beta_0}(\pi_0)}\right)\left(\log\left(1 - \frac{\varepsilon^3}{1280\alpha^2\ln(2m)}\right)\right)^{-1} \leq \log\left(\frac{\Phi_{\beta_0}(\pi_0)}{\Phi_\beta(\pi)}\right)\frac{1280\alpha^2\ln(2m)}{\varepsilon^3}.$$

As $\ln(2m) \in O(\log n)$, it remains show that $\frac{\Phi_{\beta_0}(\pi_0)}{\Phi_\beta(\pi)} \in O(\alpha)$. Using that π_0 is an α-approximate solution and that β_0 is such that $4\ln(2m) \leq \varepsilon\beta_0\Phi_{\beta_0}(\pi^0)$, we obtain that

$$\Phi_{\beta_0}(\pi^0) = \operatorname{lse}_{\beta_0}\left(W^{-1}A^T\pi^0\right) \overset{(3)}{\leq} \|W^{-1}A^T\pi^0\|_\infty + \frac{\ln(2m)}{\beta_0} \leq \alpha\|W^{-1}A^T\pi^*\|_\infty + \frac{\varepsilon\Phi_{\beta_0}(\pi^0)}{4}$$

and thus $\Phi_{\beta_0}(\pi^0) \leq \alpha\|W^{-1}A^T\pi^*\|_\infty/(1-\varepsilon/4)$. On the other hand, $\Phi_\beta(\pi) \geq \|W^{-1}A^T\pi\|_\infty \geq \|W^{-1}A^T\pi^*\|_\infty$ and thus $\frac{\Phi_{\beta_0}(\pi^0)}{\Phi_\beta(\pi)} \leq \frac{\alpha}{1-\varepsilon/4} = O(\alpha)$ and the bound follows. ◀

We remark that one can first run the gradient descent algorithm with $\varepsilon = 1/2$ and then switch to the desired accuracy. Using this trick, the above bound slightly improves to $O((\varepsilon^{-3} + \log\alpha)\alpha^2 \log n)$. From the discussion so far, we obtain the following result.

▶ **Theorem 5.** *Given an oracle that computes α-approximate solutions to the undirected transshipment problem, using Algorithm* `gradient_transship`*, we can compute primal and dual solutions x, y to the shortest transshipment problem satisfying $\|Wx\|_1 \leq (1+\varepsilon)b^Ty$ with $\tilde{O}(\varepsilon^{-3}\alpha^2)$ oracle calls. If the oracle only returns α-approximate dual solutions, then Algorithm* `gradient_transship` *computes a $(1+\varepsilon)$-approximate dual solution.*

---

[13] Hölder's inequality states that $x^Ty \leq \|x\|_p\|y\|_q$ for p, q satisfying $\frac{1}{p} + \frac{1}{q} = 1$, assuming $\frac{1}{\infty} = 0$.

[14] Here, we omit the technical argument that the condition $\|W^{-1}A^TP\tilde{h}\|_\infty > 0$ of Lemma 3 is always fulfilled when we apply the lemma. See full version for details.

2.2 Single-Source Shortest Paths

In the special case of SSSP, we have $b_v = 1$ for all $v \in V \setminus \{s\}$ and $b_s = 1 - n$ for the source s. In fact, it is the combination of $n-1$ shortest s-t-path problems. Let π be the potentials returned by Algorithm `gradient_transship` and let us assume, w.l.o.g., that $\pi_s = 0$ (otherwise shift $\pi \leftarrow \pi - \pi_s \mathbb{1}$). Recall that in an optimal solution π^* with $\pi^*_s = 0$ the value of π^*_v for any v denotes the distance from s to v. Thus the approximation guarantee from Theorem 5 yields that for the potentials π, it holds that $\sum_{v \neq s} \pi_v \leq (1+\varepsilon) \sum_{v \neq s} \pi^*_v$, i.e., the distances merely approximate the optimal distances *on average* over all sink-nodes, which is unsatisfactory. However, we can obtain potentials π such that for *every* v, it holds that $\pi_v \leq (1+\varepsilon)\pi^*_v$ and equivalently $y^* \geq y_v \geq y^*_v/(1+\varepsilon)$ for the s-v-distances.

Using the tools proposed above, we can show that when running the gradient descent algorithm with higher precision, we can determine "good" nodes for which we know the distance with sufficient accuracy by checking, for every node v, whether the gradient would allow further progress for the s-v shortest path problem. We then argue that a constant fraction of the nodes will be "good" when the algorithm is finished. We then concentrate on the other nodes by adapting the demand vector b accordingly, i.e., setting $b_v = 0$ for all good nodes v. We iterate until all nodes are good. The pseudocode is given in Algorithm `sssp`.

▶ **Theorem 6.** *Let $y^* \in \mathbb{R}^n$ denote the distances of all nodes from the source node s. Algorithm `sssp` computes a vector $y \in \mathbb{R}^n$ with $\|W^{-1}A^T y\|_\infty \leq 1$ such that $y^*_v/(1+\varepsilon) \leq y_v \leq y^*_v$ holds for each $v \in V$, using* $\operatorname{polylog}(n, \|w\|_\infty)$ *calls to Algorithm* `gradient_transship`.

2.3 Finding a Primal Tree Solution

In the following, we explain how to obtain primal tree solutions, for a specific implementation of the transshipment oracle from Section 3, where we solve the subproblem on spanner.

Recall that, as shown in Lemma 2, $x := \frac{\tilde{x} - \tilde{f}}{\pi^T \nabla \Phi_\beta(\pi)}$ is a $(1+\epsilon)$-approximate primal solution, where $\tilde{f}$ is the primal solution computed by the oracle in the last iteration of the algorithm and $\tilde{x} := W^{-1} \nabla \operatorname{lse}_\beta(W^{-1}A^T\pi)$. To also obtain a $(1+\epsilon)$-approximate primal *tree* solution, we first sample a tree, say T_1, from $\tilde{x}$ by sampling for each node among its incident edges a parent edge with probabilities proportional to the values in $\tilde{x}$. Then we compute an optimal

Algorithm 2: `sssp` (G, s, ε)

Let $\hat{y} = 0$, $b = \mathbb{1} - n\mathbb{1}_s$, and $\varepsilon' = \frac{\varepsilon^3}{3840\alpha^2 \ln(2m)}$.
while $b_s < 0$ **do**
  Set $\pi = \texttt{gradient\_transship}(G, b, \varepsilon')$ and $y = \frac{\pi}{\|W^{-1}A^T\pi\|_\infty}$.
  Determine β so that $4\ln(2m) < \varepsilon' \beta \Phi_\beta(y) \leq 5\ln(2m)$ and compute $\nabla \Phi_\beta(y)$.
  for *each* $v \in V$ *with* $b_v = 1$ **do**
    Set $\tilde{b} := P^T \nabla \Phi_\beta(y)$, where $P := [I - \frac{y}{(\mathbb{1}_v - \mathbb{1}_s)^T y}(\mathbb{1}_v - \mathbb{1}_s)^T]$.
    Compute $\tilde{h}$ with $\|W^{-1}A^T\tilde{h}\|_\infty = 1$ and $\tilde{b}^T\tilde{h} \geq \frac{1}{\alpha}\max\{\tilde{b}^T h : \|W^{-1}A^T h\|_\infty \leq 1\}$. // *$\tilde{h}$ can be obtained from the oracle with demand vector $\tilde{b} = P^T \nabla \Phi_\beta(\pi)$*
    Set $\delta := \frac{\tilde{b}^T\tilde{h}}{\|W^{-1}A^T P\tilde{h}\|_\infty}$.
    if $\delta \leq \frac{\varepsilon}{8\alpha}$ **then** set $b_v = 0$, $\hat{y}_v = y_v - y_s$ and $b_s \leftarrow b_s + 1$
return $\hat{y}$

tree solution x_T in the graph $G' = (V, T_1 \cup S)$ consisting of the tree T_1 and the edges of the spanner S. As $\mathrm{E}[\|Wx(T_1)\|_1] = 1$, using Markov's inequality we get that, with probability $\Omega(\varepsilon)$, the tree solution $x(T_1)$ corresponding to T_1 satisfies $\|Wx(T_1)\|_1 \leq 1+\varepsilon/8$. Repeating the sampling $O(\varepsilon \log n)$ times and taking the best result, we obtain such a solution with high probability. Thus, using Lemma 2, we can conclude that the optimal tree solution x_T in G' is $(1+\varepsilon)$-approximate for the problem. To obtain an approximate tree solution in the case of SSSP, we repeat this sampling after every call of the gradient descent algorithm, obtaining a tree T_i in the i-th call. We can then find the approximate shortest path tree in the graph $G' = (V, \bigcup_i T_i \cup S)$ combining the sampled edges of each iteration and the initial spanner. Note that, since the number of calls to the gradient descent algorithm is $\operatorname{polylog}(n, \|w\|_\infty)$, the resulting graph is still of size $O(n \operatorname{polylog}(n, \|w\|_\infty))$ for a spanner of size $O(n \log n)$.

3 Implementation in Various Models of Computation

Common to all our implementations is the use of sparse spanners. An optimal solution of an instance of the shortest transshipment problem on an α-spanner of the input graph is an α-approximate solution to the original problem. Thus, whenever our gradient descent algorithm asks the oracle for an α-approximate solution to a subproblem, we solve the subproblem on a spanner to get an approximation with $\alpha = O(\log n)$.

Broadcast Congested Clique. In the *broadcast congested clique* model, the system consists of n fully connected nodes labeled by unique $O(\log n)$-bit identifiers. Computation proceeds in synchronous rounds, where in each round, nodes may perform arbitrary local computations, broadcast (send) an $O(\log n)$-bit message to the other nodes, and receive the messages from other nodes. The input is distributed among the nodes. The first part of the input of every node consists of its incident edges (given by their endpoints' identifiers) and their weights. The second part of the input is problem specific: for the transshipment problem, every node v knows its demand b_v and for SSSP v knows whether or not it is the source s. In both cases, every node knows $0 < \varepsilon \leq 1/2$ as well. Each node needs to compute its part of the output. For shortest transshipment, every node in the end needs to know a $(1+\varepsilon)$-approximation of the optimum value, and for SSSP every node needs to know a $(1+\varepsilon)$-approximation of its distance to the source. The complexity of the algorithm is measured in the worst-case number of rounds until the computation is complete.

Implementing our approach in this model is straightforward. The key observations are:

- Every node can locally aggregate information about its incident edges (e.g. concerning the "stretches" under the potential of the current solution π) and make it known to all other nodes in a single communication round. Thus, given $\beta > 0$ and $\pi \in \mathbb{R}^n$, it is rather straightforward to evaluate $\Phi_\beta(\pi)$ and $\nabla\Phi_\beta(\pi)$ in a constant number of rounds.
- An $O(\log n)$-spanner of the input graph can be computed and made known to all nodes quickly, following the algorithm of Baswana and Sen [3] (see full paper for details).
- Local computation then suffices to solve (sub)problems on the spanner optimally. In particular, $O(\log n)$-approximations to transshipment problems can be computed easily. It suffices to communicate the demand vector; in cases where the demand vector is known a priori (e.g. when strengthening the approximation guarantee from average to worst-case for each node in the SSSP problem), even this is not necessary.

▶ **Theorem 7.** *For any $0 < \varepsilon \leq 1/2$, in the broadcast congested clique model a deterministic $(1+\varepsilon)$-approximation to the shortest transshipment problem in undirected graphs with non-negative edge weights can be computed in $\varepsilon^{-3} \operatorname{polylog} n$ rounds.*

▶ **Theorem 8.** *For any $0 < \varepsilon \leq 1$, in the broadcast congested clique model a deterministic $(1+\varepsilon)$-approximation to single-source shortest paths in undirected graphs with non-negative edge weights can be computed in ε^{-9} polylog n rounds.*

To compute a tree solution, the main observation is that the sampling of the tree can be performed locally at every node.

Broadcast CONGEST Model. The *broadcast CONGEST* model differs from the broadcast congested clique in that communication is restricted to edges that are present in the input graph. That is, node v receives the messages sent by node w if and only if $\{v, w\} \in E$. All other aspects of the model are identical to the broadcast congested clique. We stress that this restriction has significant impact, however: Denoting the hop diameter of the input graph (i.e., the diameter of the unweighted graph $G = (V, E)$) by D, it is straightforward to show that $\Omega(D)$ rounds are necessary to solve the SSSP problem. Moreover, it has been established that $\Omega(\sqrt{n}/\log n)$ rounds are required even on graphs with $D \in O(\log n)$ [13]. Both of these bounds apply to randomized approximation algorithms.

Our main result for this model is that we can nearly match the above lower bounds for approximate SSSP computation. The solution is based on combining a known reduction to an overlay network on $\tilde{\Theta}(\varepsilon^{-1}\sqrt{n})$ nodes, simulating the broadcast congested clique on this overlay, and applying Theorem 8. Simulating a round of the broadcast congested clique for k nodes is done by pipelining each of the k messages over a breadth-first search tree of the underlying graph, taking $O(D + k)$ rounds.

▶ **Corollary 9.** *For any $0 < \varepsilon \leq 1$, in the broadcast CONGEST model a deterministic $(1+\varepsilon)$-approximation to single-source shortest paths in undirected graphs with non-negative weights can be computed in $\tilde{O}((\sqrt{n} + D) \cdot \varepsilon^{-9})$ rounds.*

Multipass Streaming Model. In the *streaming* model the input graph is presented to the algorithm edge by edge as a "stream" without repetitions. The goal is to design algorithms that use as little space as possible. Space is counted in memory words, where we assume that an edge weight or a node identifier fits into a word. In the *multipass streaming* model, the algorithm may make several such passes over the input stream and the goal is to keep the number of passes small (again using little space). For graph algorithms, the usual assumption is that the edges of the graph are presented to the algorithm in arbitrary order.

The main observation is that we can apply the same approach as before with $O(n \log n)$ space: this enables us to store a spanner throughout the entire computation, and we can keep track of intermediate (node) state vectors. Computations on the spanner are thus "free," while $\Phi_\beta(\pi)$ and $\nabla\Phi_\beta(\pi)$ can be evaluated in a single pass by straightforward aggregation. It follows that $\varepsilon^{-O(1)}$ polylog n passes suffice for completing the computation.

▶ **Theorem 10.** *For any $0 < \varepsilon \leq 1/2$, in the multipass streaming model a deterministic $(1+\varepsilon)$-approximation to the shortest transshipment problem in undirected graphs with non-negative weights can be computed in ε^{-3} polylog n passes with $O(n \log n)$ space.*

▶ **Theorem 11.** *For any $0 < \varepsilon \leq 1$, in the multipass streaming model, a deterministic $(1+\varepsilon)$-approximation to single-source shortest paths in undirected graphs with non-negative weights can be computed in $\varepsilon^{-9} \log(\|w\|_\infty)$ polylog n passes with $O(n \log n)$ space.*

References

1 Amir Abboud, Greg Bodwin, and Seth Pettie. A hierarchy of lower bounds for sublinear additive spanners. In *Symposium on Discrete Algorithms (SODA)*, pages 568–576, 2017.

2 Yair Bartal. On approximating arbitrary metrices by tree metrics. In *Symposium on the Theory of Computing (STOC)*, pages 161–168, 1998.

3 Surender Baswana and Sandeep Sen. A simple and linear time randomized algorithm for computing sparse spanners in weighted graphs. *Random Structures & Algorithms*, 30(4):532–563, 2007. Announced at ICALP'03.

4 Richard Bellman. On a routing problem. *Quarterly of Applied Mathematics*, 16(1):87–90, 1958.

5 Aaron Bernstein. Fully dynamic $(2+\epsilon)$ approximate all-pairs shortest paths with fast query and close to linear update time. In *Symposium on Foundations of Computer Science (FOCS)*, pages 693–702, 2009.

6 Gerth Stølting Brodal, Jesper Larsson Träff, and Christos D. Zaroliagis. A parallel priority queue with constant time operations. *Journal of Parallel and Distributed Computing*, 49(1):4–21, 1998. Announced at IPPS'97.

7 Keren Censor-Hillel, Petteri Kaski, Janne H. Korhonen, Christoph Lenzen, Ami Paz, and Jukka Suomela. Algebraic methods in the congested clique. In *Symposium on Principles of Distributed Computing (PODC)*, pages 143–152, 2015.

8 Paul Christiano, Jonathan A. Kelner, Aleksander Mądry, Daniel A. Spielman, and Shang-Hua Teng. Electrical flows, laplacian systems, and faster approximation of maximum flow in undirected graphs. In *Symp. on Theory of Computing (STOC)*, pages 273–282, 2011.

9 Edith Cohen. Using selective path-doubling for parallel shortest-path computations. *Journal of Algorithms*, 22(1):30–56, 1997. Announced at ISTCS'93.

10 Edith Cohen. Polylog-time and near-linear work approximation scheme for undirected shortest paths. *Journal of the ACM*, 47(1):132–166, 2000. Announced at STOC'94.

11 Michael B. Cohen, Aleksander Madry, Piotr Sankowski, and Adrian Vladu. Negative-weight shortest paths and unit capacity minimum cost flow in $\tilde{O}(m^{10/7} \log W)$ time. In *Symposium on Discrete Algorithms (SODA)*, 2017.

12 Samuel I. Daitch and Daniel A. Spielman. Faster approximate lossy generalized flow via interior point algorithms. In *Symposium on Theory of Computing (STOC)*, pages 451–460, 2008.

13 Atish Das Sarma, Stephan Holzer, Liah Kor, Amos Korman, Danupon Nanongkai, Gopal Pandurangan, David Peleg, and Roger Wattenhofer. Distributed verification and hardness of distributed approximation. *SIAM Journal on Computing*, 41(5):1235–1265, 2012. Announced at STOC'11.

14 Jack Edmonds and Richard M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM*, 19(2):248–264, 1972.

15 Michael Elkin. Distributed exact shortest paths in sublinear time. In *Symposium on the Theory of Computing (STOC)*, pages 757–770, 2017.

16 Michael Elkin, Yuval Emek, Daniel A. Spielman, and Shang-Hua Teng. Lower-stretch spanning trees. *SIAM Journel on Computing*, 38(2):608–628, 2008. Announced at STOC'05.

17 Michael Elkin and Ofer Neiman. Hopsets with constant hopbound, and applications to approximate shortest paths. In *Symposium on Foundations of Computer Science (FOCS)*, pages 128–137, 2016.

18 Michael Elkin and Ofer Neiman. Linear-size hopsets with small hopbound, and distributed routing with low memory. *CoRR*, abs/1704.08468, 2017. `https://arxiv.org/abs/1704.08468`.

19 Lester R. Ford. Network flow theory. Technical Report P-923, The RAND Corporation, 1956.

20 Michael L. Fredman and Robert Endre Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM*, 34(3):596–615, 1987. Announced at FOCS'84.

21 Andrew V. Goldberg. Scaling algorithms for the shortest paths problem. *SIAM Journal on Computing*, 24(3):494–504, 1995. Announced at SODA'93.

22 Venkatesan Guruswami and Krzysztof Onak. Superlinear lower bounds for multipass graph processing. In *Conference on Computational Complexity (CCC)*, pages 287–298, 2013.

23 Thomas Dueholm Hansen, Haim Kaplan, Robert Endre Tarjan, and Uri Zwick. Hollow heaps. In *International Colloquium on Automata, Languages and Programming (ICALP)*, pages 689–700, 2015.

24 Monika Henzinger, Sebastian Krinninger, and Danupon Nanongkai. Decremental single-source shortest paths on undirected graphs in near-linear total update time. In *Symposium on Foundations of Computer Science (FOCS)*, pages 146–155, 2014.

25 Monika Henzinger, Sebastian Krinninger, and Danupon Nanongkai. A deterministic almost-tight distributed algorithm for approximating single-source shortest paths. In *Symposium on Theory of Computing (STOC)*, pages 489–498, 2016.

26 Shang-En Huang and Seth Pettie. Thorup-Zwick emulators are universally optimal hopsets. *CoRR*, abs/1705.00327, 2017. `https://arxiv.org/abs/1705.00327`.

27 Jonathan A. Kelner, Yin Tat Lee, Lorenzo Orecchia, and Aaron Sidford. An almost-linear-time algorithm for approximate max flow in undirected graphs, and its multicommodity generalizations. In *Symposium on Discrete Algorithms (SODA)*, pages 217–226, 2014.

28 Philip N. Klein and Sairam Subramanian. A randomized parallel algorithm for single-source shortest paths. *Journal of Algorithms*, 25(2):205–220, 1997. Announced at STOC'92.

29 Bernhard Korte and Jens Vygen. *Combinatorial Optimization*. Springer, 2000.

30 François Le Gall. Further algebraic algorithms in the congested clique model and applications to graph-theoretic problems. In *International Symposium on Distributed Computing (DISC)*, pages 57–70, 2016.

31 Yin Tat Lee and Aaron Sidford. Path finding methods for linear programming: Solving linear programs in $\tilde{O}(\sqrt{rank})$ iterations and faster algorithms for maximum flow. In *Symposium on Foundations of Computer Science (FOCS)*, pages 424–433, 2014.

32 Gary L. Miller, Richard Peng, Adrian Vladu, and Shen Chen Xu. Improved parallel algorithms for spanners and hopsets. In *Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 192–201, 2015.

33 Aleksander Mądry. Navigating central path with electrical flows: From flows to matchings, and back. In *Symposium on Foundations of Computer Science (FOCS)*, pages 253–262, 2013.

34 Danupon Nanongkai. Distributed approximation algorithms for weighted shortest paths. In *Symposium on Theory of Computing (STOC)*, pages 565–573, 2014.

35 James B. Orlin. A faster strongly polynomial minimum cost flow algorithm. *Operations Research*, 41(2):338–350, 1993. Announced at STOC'88.

36 Alexander Schrijver. *Combinatorial Optimization*. Springer, 2003.

37 Jonah Sherman. Nearly maximum flows in nearly linear time. In *Symposium on Foundations of Computer Science (FOCS)*, pages 263–269, 2013.

38 Jonah Sherman. Generalized preconditioning and network flow problems. In *Symposium on Discrete Algorithms (SODA)*, pages 772–780, 2017.

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

40 Thomas H. Spencer. Time-work tradeoffs for parallel algorithms. *Journal of the ACM*, 44(5):742–778, 1997. Announced at SODA'91 and SPAA'91.

41 Mikkel Thorup. Undirected single-source shortest paths with positive integer weights in linear time. *Journal of the ACM*, 46(3):362–394, 1999. Announced at FOCS'97.