High-Accuracy Multicommodity Flows via Iterative Refinement

Li Chen
Carnegie Mellon University, Pittsburgh, PA, USA

Mingquan Ye
University of Illinois at Chicago, IL, USA

Abstract

The multicommodity flow problem is a classic problem in network flow and combinatorial optimization, with applications in transportation, communication, logistics, and supply chain management, etc. Existing algorithms often focus on low-accuracy approximate solutions, while high-accuracy algorithms typically rely on general linear program solvers. In this paper, we present efficient high-accuracy algorithms for a broad family of multicommodity flow problems on undirected graphs, demonstrating improved running times compared to general linear program solvers. Our main result shows that we can solve the $\ell_{q,p}$-norm multicommodity flow problem to a $(1+\varepsilon)$ approximation in time $O_{q,p}(m^{1+o(1)}k^2\log(1/\varepsilon))$, where k is the number of commodities, and $O_{q,p}(\cdot)$ hides constants depending only on q or p. As q and p approach to 1 and ∞ respectively, $\ell_{q,p}$-norm flow tends to maximum concurrent flow.

We introduce the first iterative refinement framework for $\ell_{q,p}$-norm minimization problems, which reduces the problem to solving a series of decomposable residual problems. In the case of k-commodity flow, each residual problem can be decomposed into k single commodity convex flow problems, each of which can be solved in almost-linear time. As many classical variants of multicommodity flows were shown to be complete for linear programs in the high-accuracy regime [Ding-Kyng-Zhang, ICALP'22], our result provides new directions for studying more efficient high-accuracy multicommodity flow algorithms.

2012 ACM Subject Classification Theory of computation → Continuous optimization

Keywords and phrases High-accuracy multicommodity flow, Iterative refinement framework, Convex flow solver

Digital Object Identifier 10.4230/LIPIcs.ICALP.2024.45

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/pdf/2304.11252`

Funding *Li Chen*: Supported by NSF grants CCF-2106444 and CCF-2330255.
Mingquan Ye: Supported by NSF grant CCF-2240024.

Acknowledgements We thank Richard Peng and Xiaorui Sun for helpful discussions.

1 Introduction

The multicommodity flow problem is a classic challenge in network flow and combinatorial optimization, where the goal is to optimally route multiple commodities through a network from their respective sources to their respective sinks, subject to flow conservation constraints. This problem has significant applications in various fields such as transportation, communication, logistics, and supply chain management [43, 6, 57, 9, 71]. Currently, the fastest algorithms for computing high-accuracy solutions involve formulating these problems as linear programs and employing generic linear program solvers [44, 39, 62, 22]. Notably, linear programs can be reduced to multicommodity flow problems with near-linear overhead [35, 26].

51st International Colloquium on Automata, Languages, and Programming (ICALP 2024).
Editors: Karl Bringmann, Martin Grohe, Gabriele Puppis, and Ola Svensson;
Article No. 45; pp. 45:1–45:19


Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Existing research predominantly focuses on obtaining $(1+\varepsilon)$-approximate solutions for maximum concurrent k-commodity flows [49, 48, 73, 27, 37, 30, 56, 51], as summarized in Table 1. However, these low-accuracy algorithms feature running times that are polynomial in $1/\varepsilon$ for computing $(1+\varepsilon)$-approximate solutions. In contrast, high-accuracy algorithms exhibit running times that are polynomial in $\log(1/\varepsilon)$. Importantly, [26] demonstrated that any enhancement in the high-accuracy algorithm for the 2-commodity flow problem would result in a faster general linear program solver.

In this paper, we investigate multicommodity flow problems on undirected graphs, which possess more structure than their directed counterparts. Prior work has shown that maximum concurrent 2-commodity flow on undirected graphs can be reduced to two instances of maximum flow problems, both solvable in almost-linear time [63, 15]. More generally, maximum concurrent k-commodity flows can be reduced to 2^{k-1} maximum flows[1]. Additionally, researchers have discovered that $(1+\varepsilon)$-approximate algorithms for undirected graphs are considerably faster than those for directed graphs [42, 41, 66]. Nevertheless, the fastest high-accuracy algorithms still rely on general linear program solvers. Given these advancements, we pose the following natural question:

> *Is it possible to solve multicommodity flow problems on undirected graphs to high accuracy more efficiently than with general linear program solvers?*

This paper gives an affirmative answer and presents high-accuracy algorithms for a large family of multicommodity flow problems that run in time $m^{1+o(1)}\mathrm{poly}(k, \log(1/\varepsilon))$. Our main result is an algorithm that, for any $1 \le q \le 2 \le p$ with $p = \widetilde{O}(1)$[2] and $\frac{1}{q-1} = O(1)$, given edge weights $\mathbf{w} \in \mathbb{R}^E_+$ and k vertex demands $\mathbf{D} = [\mathbf{d}_1, \cdots, \mathbf{d}_k] \in \mathbb{R}^{V \times k}$, solves the following optimization problem to a $(1+\varepsilon)$-approximation with high probability[3] in time $O_{q,p}(m^{1+o(1)}k^2 \log(1/\varepsilon))$:

$$\min_{k\text{-commodity flow } \mathbf{F} \text{ with residue } \mathbf{D}} \|\mathbf{WF}\|_{q,p}^{pq} \stackrel{\text{def}}{=} \sum_{e \in E} w_e^{pq} \left(\sum_{j=1}^{k} |F_{ej}|^q \right)^p.$$

The problem generalizes the maximum concurrent flow problem by setting the edge weights w_e as the reciprocal of edge capacities and letting $q \to 1, p \to \infty$. Therefore, $\ell_{q,p}$-norm flows are natural relaxations of the combinatorial maximum concurrent flows. However, unlike the typical relaxations using the exponential function (exp(congestion)) in previously efficient approximation schemes, we show that $\ell_{q,p}$-norm problems themselves admit high-accuracy solutions. Thus, we provide a large family of multicommodity flows that admit high-accuracy solutions in almost linear time.

From a technical standpoint, our exploration of multicommodity flows reflects the research trajectory of ℓ_p-norm single commodity flows in recent years [46, 1]. This line of study has led to the development of several novel algorithmic components, some of which have proven beneficial for classical single-commodity flow as well [40, 7]. More significantly, examining ℓ_p-norm flows, particularly their weighted variants, has directed attention towards the core challenges of flow problems. We posit that further investigation of $\ell_{q,p}$-norm flows is likely to yield similar insights, potentially for variants of multicommodity flows not known to be hard, such as unit-capacity maximum concurrent flows on undirected graphs.

---

[1] To see this, the edge capacity constraint $\sum_j |F_{ej}| \le u_e$ is equivalent to $|\sum_j \varepsilon_j F_{ej}| \le u_e$ for any $\varepsilon \in \{\pm 1\}^k$.

[2] We use $\widetilde{O}(f(n))$ to hide $\mathrm{poly}\log f(n)$ factors.

[3] We use "with high probability" (w.h.p.) throughout to say that an event happens with probability at least $1 - n^{-C}$ for any constant $C > 0$.

To summarize, this paper introduces the first iterative refinement framework for solving $\ell_{q,p}$-norm minimization problems. The proposed framework reduces the problem to approximately resolving $O_{p,q}(k\log(1/\varepsilon))$ instances of decomposable residual problems. For the k-commodity flow case, each residual problem can be divided into k single commodity flow problems and solved in $km^{1+o(1)}$-time using the almost-linear time convex flow solver [15]. We are the first to combine the iterative refinement framework [2, 3] with the convex flow solver [15], and give a non-trivial application to $\ell_{q,p}$-norm multicommodity flow problem. As many classical variants of multicommodity flows were shown to be complete for linear programs in the high-accuracy regime [26], our result provides new directions for studying more efficient high-accuracy multicommodity flow algorithms.

1.1 Main Result

Given an undirected graph $G=(V,E)$ and parameters q and p, the *$\ell_{q,p}$-norm multicommodity flow* problem asks for a multicommodity flow $\mathbf{F}=[\mathbf{f}_1,\cdots,\mathbf{f}_k]\in\mathbb{R}^{E\times k}$ that routes the given demands while minimizing the following objective:

$$\min_{\mathbf{B}^\top\mathbf{F}=\mathbf{D}}\|\mathbf{WF}\|_{q,p}^{pq}\stackrel{\text{def}}{=}\sum_{e\in E}w_e^{pq}\left(\sum_{j=1}^{k}|F_{ej}|^q\right)^p. \tag{1}$$

Here, we have

1. $\mathbf{B}\in\mathbb{R}^{E\times V}$, the edge-vertex incidence matrix of G;
2. $\mathbf{D}=[\mathbf{d}_1,\cdots,\mathbf{d}_k]\in\mathbb{R}^{V\times k}$, representing the set of k vertex demands, where $\langle\mathbf{d}_j,\mathbf{1}_V\rangle=0$ for each $j\in[k]$;
3. $\mathbf{w}\in\mathbb{R}_+^E$, the vector of edge weights, and $\mathbf{W}=\operatorname{diag}(\mathbf{w})$.

This problem can be seen as a generalization of the classical maximum concurrent flow problem, which aims to find a feasible flow $\mathbf{F}$ that minimizes edge congestion:

$$\min_{\mathbf{B}^\top\mathbf{F}=\mathbf{D}}\max_{e\in E}\frac{1}{c_e}\sum_{j\in[k]}|F_{ej}|=\left\|\mathbf{C}^{-1}\mathbf{F}\right\|_{1,\infty}. \tag{2}$$

Here, $\mathbf{c}\in\mathbb{R}_+^E$ denotes the vector of edge capacities and $\mathbf{C}=\operatorname{diag}(\mathbf{c})$. Our generalization allows for fractional vertex demands, meaning that for each commodity j, the demand $\mathbf{d}_j$ is not restricted to source-sink pairs.

The main technical result of this paper is presented below.

▶ **Theorem 1.** *Given any $1<q\le 2\le p$ with $p=\widetilde{O}(1)$ and $\frac{1}{q-1}=O(1)$ and an error parameter $\varepsilon>\exp(-\widetilde{O}(1))$, let* OPT *be the optimal value to Problem* (1)*. There is a randomized algorithm that computes a feasible flow $\mathbf{F}$ to Problem* (1) *such that*

$$\|\mathbf{WF}\|_{q,p}^{pq}\le(1+\varepsilon)\,\mathrm{OPT}+\varepsilon.$$

The algorithm runs in time

$$O\left(p^2\cdot\left(\frac{1}{q-1}\right)^{\frac{1}{q-1}}m^{1+o(1)}\cdot k^2\cdot\log\frac{1}{\varepsilon}\right)$$

with high probability.

Table 1 A summary of algorithms for the max concurrent k-commodity flow problem.

Year	References	Time	Directed?
1990	[64]	$O(nm^7/\varepsilon^5)$	Directed
1991	[49]	$\widetilde{O}(mnk/\varepsilon^3)$	Directed
1996	[34]	$\widetilde{O}(mnk/\varepsilon^2)$	Directed
1996	[61]	$\widetilde{O}(mnk/\varepsilon^2)$	Directed
2009	[56]	$\widetilde{O}(k^2m^2/\varepsilon)$	Directed
2010	[51]	$\widetilde{O}((m+k)n/\varepsilon^2)$	Directed
2012	[42]	$\widetilde{O}(m^{4/3}\text{poly}(k, \frac{1}{\varepsilon}))$	Undirected
2014	[41]	$m^{1+o(1)}k^2/\varepsilon^2$	Undirected
2017	[66]	$\widetilde{O}(mk/\varepsilon)$	Undirected
2019	[21]	$(mk)^{\omega+o(1)} \log \frac{1}{\varepsilon}$	Directed

1.2 Related Works

In this section, we discuss some work related to the problem and the techniques we use.

Multicommodity Flow

The multicommodity flow problem is a classic problem in network flow and combinatorial optimization. Multicommodity flow has a wide range of applications in various fields which are addressed in numerous surveys [43, 6, 57, 9, 71]. These problems can be formulated as linear programs and solved using generic linear program solvers which remain the fastest algorithms for computing high-accuracy solutions [44, 39, 62, 22]. On the other hand, linear programs can be reduced to 2-commodity flow efficiently [35, 26]. Specifically, any linear program can be reduced to a maximum throughput 2-commodity flow on sparse graphs with a near-linear overhead [26]. Recently, Brand and Zhang [13] proposed a faster algorithm for 2-commodity flow on non-sparse graphs with running time $\widetilde{O}(\sqrt{m}n^{\omega-1/2})$ that outperforms the existing time complexities $\widetilde{O}(m^\omega)$ [22] and $\widetilde{O}(\sqrt{m}n^2)$ [36]. For the general k-commodity flow, they proposed an algorithm with time complexity $\widetilde{O}(k^{2.5}\sqrt{m}n^{\omega-1/2})$.

Much of the existing works focus on finding $(1+\varepsilon)$-approximate solutions. [64] gave the first FPTAS to the maximum concurrent flow problem with unit capacity (Problem (2)). Subsequently, a series of work [49, 48, 32, 33, 45, 38, 60, 34, 61, 67] based on Lagrangian relaxation and linear program decomposition gave algorithms with improved running times for various versions of the problem with arbitrary edge capacity. These algorithms iteratively update the current flow and make progress by computing a series of either shortest paths [64, 45, 60, 67], or single commodity minimum cost flows [49, 48, 32, 33, 38, 34, 61]. In particular, [61] and [34] showed that finding $(1+\varepsilon)$-approximate solutions can be reduced to $\widetilde{O}(k/\varepsilon^2)$ min-cost flow computations which take $\widetilde{O}(kmn/\varepsilon^2)$-time in total using the fastest min-cost flow algorithm at the time. Combining with the almost-linear time min-cost flow algorithm yields a randomized $m^{1+o(1)}k/\varepsilon^2$-time max concurrent flow algorithm.

Later, a series of works based on multiplicative weight updates (MWU) gave conceptually simpler and faster algorithms at their time [73, 27, 37, 30, 51]. These methods build the solution from scratch without re-routing the current flow. At each step, they augment the current flow via a shortest path computation that favors relatively uncongested paths. [51] used dynamic APSP data structure to speed up these computations and resulted in an

$\widetilde{O}((m+k)n/\varepsilon^2)$-time max concurrent flow algorithm. At the time, the fastest max concurrent flow runs in $\widetilde{O}(m^{1.5}k/\varepsilon^2)$-time due to the $\widetilde{O}(m^{1.5})$-time min-cost flow algorithm by [24]. The result of [51] was a significant improvement in the case when k is large.

Another line of work focused on improving the $1/\varepsilon^2$ term. [11] found an FPTAS that only has $O(\frac{1}{\varepsilon \log 1/\varepsilon})$ dependence. Later, [56] gave an FPTAS with only $O(1/\varepsilon)$ dependence. In particular, the algorithm by Nesterov runs in $\widetilde{O}(k^2m^2/\varepsilon)$-time.

Similar to the situation of single commodity flows, researchers have discovered approximate algorithms with $m^{1.5-O(1)}$ dependence on undirected graphs. [42] gave the first $\widetilde{O}(m^{4/3}\mathrm{poly}(k, 1/\varepsilon))$ algorithm for max concurrent flow. The algorithm implements the method of [49, 48] using *electrical capacity-constrained flows* instead of min-cost flows. Each electrical capacity-constrained flow can be reduced to, using width-reduction MWU [17], $\widetilde{O}(m^{1/3}\mathrm{poly}(k, 1/\varepsilon))$ graph Laplacian systems. In the breakthrough result of [41], they improved the running time to $m^{1+o(1)}k^2/\varepsilon^2$ based on non-Euclidean gradient descent and fast oblivious routing. Specifically, the algorithm computes an oblivious routing of congestion $m^{o(1)}$ and uses it to reduce the number of gradient descent iterations to $m^{o(1)} \cdot k/\varepsilon^2$. Later, [66] introduced the idea of *area convexity* and improved the iteration count to $\widetilde{O}(1/\varepsilon)$. This resulted in the first $\widetilde{O}(mk/\varepsilon)$-time max concurrent flow algorithm.

ℓ_p-Norm Regression

The ℓ_p-norm regression problem seeks to find a vector $\mathbf{x}$ that minimizes $\|\mathbf{Ax} - \mathbf{b}\|_p$, where $\mathbf{A} \in \mathbb{R}^{d\times n}$ and $\mathbf{b} \in \mathbb{R}^d$. Varying in p interpolates between linear regression ($p = 2$) and linear program ($p \in \{1, \infty\}$). ℓ_p-norm regression has gained significant attention in the past decade due to its wide range of applications and its implications for other convex optimization problems [54, 72]. Many works have focused on low-accuracy algorithms for overconstrained matrices, i.e., $d \gg n$ [25, 54, 72, 19, 20, 18]. These results show various ways to find another matrix $\widetilde{\mathbf{A}}$ with fewer rows such that $\|\widetilde{\mathbf{A}}\mathbf{x}\|_p \approx \|\mathbf{Ax}\|_p$ for any $\mathbf{x}$. Then, approximate ℓ_p-norm regression can be reduced to $\widetilde{\mathbf{A}}$ and solved in time $O(\mathsf{nnz}(\mathbf{A}) + \mathrm{poly}(d, 1/\varepsilon))$ when p is a constant.

High-accuracy solutions can be found using interior point methods (IPM) in $\widetilde{O}(\sqrt{n})$ or $\widetilde{O}(\sqrt{d})$ iterations [47]. [14] showed a homotopy method that finds high accuarcy solution in $\widetilde{O}(n^{|1/2-1/p|})$ iterations of linear system solvers. Based on the idea of iterative refinement and width reduction, a series of works [2, 4, 5, 3] obtained improved iteration complexities of $\widetilde{O}_p(n^{(p-2)/(3p-2)})$ for $p \geq 2$. Motivated by the success of sparse linear system solver [59] and linear program in matrix multiplication time [22], [31] gave a high-accuracy algorithm that runs in time $\widetilde{O}_p(n^\theta)$ for some $\theta < \omega - \Omega(1)$, where ω is the matrix multiplication time exponent.

ℓ_p-Norm Flows

The $\ell_{q,p}$-norm formulation can be viewed as a multicommodity extension of the ℓ_p-norm flows, which seeks to find a flow $\mathbf{f} \in \mathbb{R}^E$ that routes the given demand and minimizes $\|\mathrm{diag}(\mathbf{w})\mathbf{f}\|_p$ where $\mathbf{w} \in \mathbb{R}^E_+$. Varying in p interpolates between the transshipment ($p = 1$), the electrical flow ($p = 2$), and the maximum flow problem ($p = \infty$). Combining the result on ℓ_p-norm regressions and Laplacian solvers, [2] gave an $\widetilde{O}_p(m^{1+|p-2|/(2p+|p-2|)})$-time ℓ_p-norm flow algorithm. Opening up the black box of the Spielman-Teng Laplacian solver [68], [46] gave the first $\widetilde{O}_p(m^{1+O(1/\sqrt{p})})$-time high-accuracy ℓ_p-norm flow algorithm for unweighted graphs, i.e., $\mathbf{w} = \mathbf{1}$. The runtime is almost linear for $p = \omega(1)$ and this was the first almost-linear time high-accuracy algorithm for a large family of single commodity flow problems. In the weighted case, [1] gave a $p(m^{1+o(1)} + n^{4/3+o(1)})$-time high-accuracy ℓ_p-norm flow algorithm

by combining [46] with the idea of sparsification. The study of the ℓ_p-norm flow algorithms has been proved useful for single commodity flow problems such as unit-capacity maximum flows, bipartite matchings, and min-cost flows [40, 7].

Continuous Optimization on Graphs

From a technical point of view, our $\ell_{q,p}$-norm multicommodity flow algorithm is inspired by the recent trend of applying continuous optimization techniques to solve graph problems. As one of the earlier results in this direction, [24] combined interior point methods with fast graph Laplacian system solver [68] and gave an $\widetilde{O}(m^{3/2})$-time min-cost flow algorithm. Later, the idea culminated in a decade of works improving max flow and min-cost flow algorithms [17, 52, 65, 41, 53, 58, 23, 7, 12, 40, 70, 8, 10, 15, 29, 69].

Beyond classical flow problems, the idea of combining continuous and combinatorial techniques gives improved algorithms for approximate shortest paths in parallel/distributed setting [50, 74], faster network flow algorithms in distributed setting [28], flow diffusion [16], ℓ_p-norm flows [46, 1], and more.

1.3 Our Approach

For the clarity of the presentation, we focus on the unweighted version of Problem (1), i.e., $w_e = 1$ for each edge e, which is shown as follows:

$$\min_{\mathbf{B}^\top \mathbf{F} = \mathbf{D}} \|\mathbf{F}\|_{q,p}^{pq} = \sum_{e \in E} \left(\sum_{j=1}^{k} |F_{ej}|^q \right)^p .$$

The algorithm follows an overall *iterative refinement* framework. That is, given the current flow $\mathbf{F}$, we want to find an update direction $\mathbf{\Delta}$ so that $\|\mathbf{F} + \mathbf{\Delta}\|_{q,p}^{pq}$ is smaller than $\|\mathbf{F}\|_{q,p}^{pq}$. However, finding the optimal $\mathbf{\Delta}$ is equivalent to the original problem. The idea of iterative refinement is to find a proxy *residual* function $\mathcal{R}(\mathbf{\Delta}; \mathbf{F})$ that approximates the Bregman divergence of $\|\mathbf{F}\|_{q,p}^{pq}$, i.e.,

$$\mathcal{R}(\mathbf{\Delta}; \mathbf{F}) \approx \|\mathbf{F} + \mathbf{\Delta}\|_{q,p}^{pq} - \|\mathbf{F}\|_{q,p}^{pq} - \langle \mathbf{G}, \mathbf{\Delta} \rangle ,$$

where $\mathbf{G} \stackrel{\text{def}}{=} \frac{\mathrm{d}}{\mathrm{d}\mathbf{F}} \|\mathbf{F}\|_{q,p}^{pq} \in \mathbb{R}^{E \times k}$ is the gradient. Then we can compute the direction $\mathbf{\Delta}$ by solving the *residual problem*:

$$\min_{\mathbf{B}^\top \mathbf{\Delta} = \mathbf{0}} \langle \mathbf{G}, \mathbf{\Delta} \rangle + \mathcal{R}(\mathbf{\Delta}; \mathbf{G}) \approx \|\mathbf{F} + \mathbf{\Delta}\|_{q,p}^{pq} - \|\mathbf{F}\|_{q,p}^{pq}.$$

If $\mathcal{R}(\mathbf{\Delta}; \mathbf{F})$ is a good approximation to the Bregman divergence, i.e., has a "condition number" of κ, we would obtain a $(1+\varepsilon)$-approximate solution in $O(\kappa \log(1/\varepsilon))$ iterations. On the other hand, $\mathcal{R}(\mathbf{\Delta}; \mathbf{F})$ should be computationally easier to minimize so that we can implement each iteration efficiently.

For any $p > 1$, [2] shows that $|x + \delta|^p$ can be locally approximated by a linear term plus an error term $\gamma_p(\delta; |x|)$, which behaves quadratically in δ when $|\delta| \leq |x|$ and as $|\delta|^p$ otherwise. Our key lemma (Lemma 8) extends the observation to approximating $\|\mathbf{F} + \mathbf{\Delta}\|_{q,p}^{pq}$ and gives

$$\begin{aligned} \|\mathbf{F} + \mathbf{\Delta}\|_{q,p}^{pq} - \|\mathbf{F}\|_{q,p}^{pq} - \langle \mathbf{G}, \mathbf{\Delta} \rangle \approx{}& O_{p,q}(k) \sum_{e,j} \|\mathbf{f}^e\|_q^{q(p-1)} \gamma_q(\Delta_{ej}; F_{ej}) \\ &+ O_{p,q}(k^p) \sum_{e,j} |\Delta_{ej}|^{pq}. \end{aligned} \tag{3}$$

Intuitively, the Bregman divergence can be approximated by a decomposable function on each coordinate (e, j). The contribution of each coordinate (e, j) behaves differently depending on the absolute value of Δ_{ej}. When $|\Delta_{ej}| \leq |F_{ej}|$, it behaves quadratically in $|\Delta_{ej}|^q$; when $|\Delta_{ej}| > |F_{ej}|$ but smaller than $\|\mathbf{f}^e\|_q^q$, the summation of k flow values to the power of q on edge $e \in E$, it is dominated by the term $|\Delta_{ej}|^{pq}$. The factors k and k^p in Equation (3) come from that given any k-dimensional vector $\mathbf{x} \in \mathbb{R}^k$, we have

$$\|\mathbf{x}\|_p^p \leq \|\mathbf{x}\|_1^p \leq k^{p-1} \|\mathbf{x}\|_p^p .$$

Surprisingly, this approximation has a conditioner number of $O_{q,p}(k)$ and is decomposable for each commodity $j \in [k]$. To obtain a $(1+\varepsilon)$-approximate solution, we only need to solve $O_{q,p}(k \log(1/\varepsilon))$ iterations of residual problems of the form

$$\min_{\mathbf{B}^\top \boldsymbol{\Delta} = \mathbf{0}} \langle \mathbf{G}, \boldsymbol{\Delta} \rangle + O_{p,q}(k) \sum_{e,j} \|\mathbf{f}^e\|_q^{q(p-1)} \gamma_q(\Delta_{ej}; F_{ej}) + O_{p,q}(k^p) \sum_{e,j} |\Delta_{ej}|^{pq}.$$

The decomposability allows us to use the convex flow solver from [15] and solve the residual problem to high accuracy for each commodity in almost-linear time. Thus, each iteration takes $km^{1+o(1)}$-time and the final running time is $O_{q,p}(k^2 m^{1+o(1)} \log(1/\varepsilon))$.

1.4 Paper Organization

In Section 2, we introduce some preliminaries before presenting the technical parts, including the convex flow solver [15] that is the core tool for solving the residual problem and the iterative refinement framework shown in [2, 3]. We formally present the proposed $\ell_{q,p}$-norm k-commodity flow algorithm in Section 3, and then solve the residual problem in Section 4.

2 Preliminaries

2.1 General Notations

We denote vectors (resp. matrices) by boldface lowercase (reps. uppercase) letters. For a vector $\mathbf{x} \in \mathbb{R}^n$, the scalar x_i, $i \in [n]$ represents the i-th entry of $\mathbf{x}$. For two vectors $\mathbf{x}$ and $\mathbf{y}$, the vector $\mathbf{x} \cdot \mathbf{y}$ represents the entrywise product, i.e., $(\mathbf{x} \cdot \mathbf{y})_i = x_i y_i$. Besides, for a vector $\mathbf{x}$, let $|\mathbf{x}|$ and $\mathbf{x}^p$ denote the entrywise absolute value and entrywise power of $\mathbf{x}$ respectively, that is, $|\mathbf{x}|_i = |x_i|$ and $(\mathbf{x}^p)_i = (x_i)^p$. We use $\langle \mathbf{x}, \mathbf{y} \rangle$ to denote the inner product of $\mathbf{x}, \mathbf{y}$, i.e., $\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}^\top \mathbf{y}$. For a vector $\mathbf{x}$, let $\mathrm{diag}(\mathbf{x})$ represent a diagonal matrix whose i-th entry is equal to x_i.

Graphs

In this paper, we consider multi-graph G, with edge set $E(G)$ and vertex set $V(G)$. When the graph is clear from context, we use the short-hand E for $E(G)$ and V for $V(G)$ with $|E| = m$ and $|V| = n$. Assume that each edge $e \in E$ has an implicit direction, used to define its edge-vertex incidence matrix $\mathbf{B} \in \mathbb{R}^{E \times V}$. Abusing notation slightly, we often write $e = (u, v) \in E$, where u and v are the tail and head of e respectively (note that technically multi-graph does not allow for edges to be specified by their endpoints). We say a flow $\mathbf{f} \in \mathbb{R}^E$ routes a demand $\mathbf{d} \in \mathbb{R}^V$ if $\mathbf{B}^\top \mathbf{f} = \mathbf{d}$. Given a k-commodity flow $\mathbf{F} \in \mathbb{R}^{E \times k}$, let $\mathbf{f}^e \in \mathbb{R}^k$ denote $\mathbf{F}$'s e-th row vector, a vector of k flow values through edge $e \in E$, and $\mathbf{f}_j \in \mathbb{R}^E$ denote $\mathbf{F}$'s j-th column vector, the flow vector of commodity $j \in [k]$; let $F_{i,j}$ denote the entry at the i-th row and j-th column of $\mathbf{F}$.

Model of Computation

In this paper, for problem instances encoded with z bits, all algorithms work in fixed-point arithmetic where words have $O(\log^{O(1)} z)$ bits, i.e., we prove that all numbers stored are in the interval $[\exp(-\log^{O(1)} z), \exp(\log^{O(1)} z)]$.

2.2 Convex Flow Solver

In this paper, we utilize the almost-linear time convex flow algorithm from [15]. Given a set of *computationally efficient* convex cost functions on edges $\{c_e(\cdot)\}_e$, the algorithm finds a single commodity flow $\mathbf{f}$ that routes the given demand $\mathbf{d}$ and minimizes $\sum_e c_e(f_e)$ up to a small $\exp(-\log^{O(1)} m)$ additive error.

▶ **Assumption 1** (Definition 10.1 and Assumption 10.2, [15]). *Let $K = \widetilde{O}(1)$ be a parameter fixed throughout. Given a convex cost function $c : \mathbb{R} \to \mathbb{R} \cup \{+\infty\}$, c is* computationally efficient *if there is a barrier function $\psi_c(f, y)$ defined on the domain $\mathcal{D}_c \stackrel{\text{def}}{=} \{(f, y) |\ c(f) \le y\}$ such that*

1. *The cost is quasi-polynomially bounded, i.e., $|c(f)| = O(m^K + |f|^K)$ for all $f \in \mathbb{R}$.*
2. *ψ_c is a ν-*self-concordant barrier *for some $\nu = O(1)$, that is, the following holds*

$$\begin{aligned} \psi_c(\mathbf{x}) &\to \infty, \text{ as } \mathbf{x} \text{ approaches the boundary of } \mathcal{D}_c, \\ \left|\nabla^3 \psi_c(\mathbf{x})[\mathbf{v}, \mathbf{v}, \mathbf{v}]\right| &\le 2\left(\nabla^2 \psi_c(\mathbf{x})[\mathbf{v}, \mathbf{v}]\right)^{3/2}, \forall \mathbf{x} \in \mathcal{D}_c, \mathbf{v} \in \mathbb{R}^2, \\ \langle \nabla \psi_c(\mathbf{x}), \mathbf{v}\rangle^2 &\le \nu \cdot \nabla^2 \psi_c(\mathbf{x})[\mathbf{v}, \mathbf{v}]. \end{aligned}$$

3. *The Hessian is quasi-polynomially bounded as long as the function value is $\widetilde{O}(1)$ bounded, i.e., for all points $|f|, |y| \le m^K$ with $\psi_c(f, y) \le \widetilde{O}(1)$, $\nabla^2 \psi_c(f, y) \preceq \exp(\log^{O(1)} m)\mathbf{I}$.*
4. *Both $\nabla \psi_c$ and $\nabla^2 \psi_c$ can be computed and accessed in $\widetilde{O}(1)$-time.*

▶ **Theorem 2** (Theorem 10.13, [15]). *Let G be a graph, and $\mathbf{d} \in \mathbb{R}^V$ be a demand vector. Given a collection of computationally efficient cost functions on edges $\{c_e(\cdot)\}_e$ and their barriers $\{\psi_e(\cdot)\}_e$ (Assumption 1), there is an algorithm that runs in $m^{1+o(1)}$ time and outputs a flow $\mathbf{f} \in \mathbb{R}^E$ that routes $\mathbf{d}$ and for any fixed constant $C > 0$,*

$$c(\mathbf{f}) \le \min_{\mathbf{B}^\top \mathbf{f}^* = \mathbf{d}} c(\mathbf{f}^*) + \exp(-\log^C m),$$

where $c(\mathbf{f}) \stackrel{\text{def}}{=} \sum_{e \in E} c_e(f_e)$.

2.3 Iterative Refinement

At a high level, the iterative refinement framework introduced by [2] approximates the Bregman Divergence of the function $|x|^p$ with something simpler.

▶ **Definition 3** (Bregman divergence). *Given a differentiable convex function $g(\cdot)$ and any two points $\mathbf{x}, \mathbf{y}$ in its domain, we define its* Bregman divergence *as*

$$D_g(\mathbf{y}, \mathbf{x}) \stackrel{\text{def}}{=} g(\mathbf{y}) - g(\mathbf{x}) - \langle \nabla g(\mathbf{x}), \boldsymbol{\Delta}\rangle = \int_0^1 \int_0^t \langle \boldsymbol{\Delta}, \nabla^2 g(\mathbf{x} + u\boldsymbol{\Delta})\boldsymbol{\Delta}\rangle \mathrm{d}u\mathrm{d}t,$$

where $\boldsymbol{\Delta} \stackrel{\text{def}}{=} \mathbf{y} - \mathbf{x}$.

For any $p > 1$, [2] shows that the Bregman Divergence of $|x + \delta|^p$ can be locally approximated by an error term $\gamma_p(\delta; |x|)$, which behaves quadratically in δ when $|\delta| \le |x|$ and as $|\delta|^p$ otherwise.

▶ **Definition 4** ([14]). *For $x \in \mathbb{R}$, $f > 0$, and $p > 1$, we define*

$$\gamma_p(x, f) \stackrel{\text{def}}{=} \begin{cases} \frac{p}{2} f^{p-2} x^2 & |x| \le f, \\ |x|^p - \left(1 - \frac{p}{2}\right) f^p & |x| > f. \end{cases}$$

For $1 < q \le 2$, [3] approximates the Bregman divergence of $|x|^q$ using the γ_q function we just defined.

▶ **Lemma 5** (Lemma 2.14, [3]). *For $q \in (1, 2]$ and $f, x \in \mathbb{R}$, it holds that*

$$\frac{q-1}{q 2^q} \cdot \gamma_q(x, |f|) \le |f + x|^q - |f|^q - q|f|^{q-2} fx \le 2^q \cdot \gamma_q(x, |f|).$$

For $p \ge 2$, the Bregman divergence of $|x|^p$ can be approximated by an x^2 and an $|x|^p$ term.

▶ **Lemma 6** (Lemma 2.5, [3]). *For $p \ge 2$ and $f, x \in \mathbb{R}$, it holds that*

$$\frac{p}{8}|f|^{p-2}x^2 + \frac{1}{2^{p+1}}|x|^p \le |f + x|^p - |f|^p - p|f|^{p-2} fx \le 2p^2|f|^{p-2}x^2 + p^p|x|^p.$$

Here we present some facts about γ_q functions that are helpful.

▶ **Lemma 7** (Lemma 3.3, [2]). *For $q \in (1, 2]$, $f, x \in \mathbb{R}$, and $t \ge 1$, it holds that*

$$t^q \cdot \gamma_q(x, |f|) \le \gamma_q(tx, |f|) \le t^2 \cdot \gamma_q(x, |f|).$$

3 Iterative Refinement Algorithm

In this section, we present the $\ell_{q,p}$-norm k-commodity flow algorithm based on iterative refinement and prove Theorem 1.

▶ **Theorem 1.** *Given any $1 < q \le 2 \le p$ with $p = \widetilde{O}(1)$ and $\frac{1}{q-1} = O(1)$ and an error parameter $\varepsilon > \exp(-\widetilde{O}(1))$, let* OPT *be the optimal value to Problem* (1)*. There is a randomized algorithm that computes a feasible flow $\mathbf{F}$ to Problem* (1) *such that*

$$\|\mathbf{WF}\|_{q,p}^{pq} \le (1 + \varepsilon)\,\mathrm{OPT} + \varepsilon.$$

The algorithm runs in time

$$O\left(p^2 \cdot \left(\frac{1}{q-1}\right)^{\frac{1}{q-1}} m^{1+o(1)} \cdot k^2 \cdot \log \frac{1}{\varepsilon}\right)$$

with high probability.

In the rest of the paper, we use $\mathcal{E}(\mathbf{F})$ to denote the objective of Problem (1), that is, $\mathcal{E}(\mathbf{F}) = \|\mathbf{WF}\|_{q,p}^{pq}$.

Our iterative refinement algorithm is based on an approximation to the Bregman divergence of the objective $\|\mathbf{WF}\|_{q,p}^{pq}$. Since the objective can be decomposed into a summation of m seperate terms, i.e., $\sum_{e \in E} w_e^{pq} \|\mathbf{f}^e\|_q^{pq}$, we approximate the Bregman divergence for each edge separately. Consequently, we present the following lemma that approximates $\|\mathbf{f}^e + \mathbf{x}^e\|_q^{pq}$ for each edge e. This is the key technical lemma of this paper and its proof can be found in the full version.

▶ **Lemma 8.** *[$\ell_{q,p}$-Norm Iterative Refinement] Given $1 < q \le 2 \le p$, and any $\mathbf{f}, \mathbf{x} \in \mathbb{R}^k$, we have*

$$\|\mathbf{f}+\mathbf{x}\|_q^{pq} - \|\mathbf{f}\|_q^{pq} - pq\|\mathbf{f}\|_q^{q(p-1)}\langle|\mathbf{f}|^{q-2}\cdot\mathbf{f},\mathbf{x}\rangle \ge \frac{p(q-1)}{16}\|\mathbf{f}\|_q^{q(p-1)}\cdot\gamma_q(\mathbf{x},|\mathbf{f}|) + \frac{q-1}{pq-1}\frac{1}{2^{pq+2}}\|\mathbf{x}\|_{pq}^{pq}, \tag{4}$$

$$\|\mathbf{f}+\mathbf{x}\|_q^{pq} - \|\mathbf{f}\|_q^{pq} - pq\|\mathbf{f}\|_q^{q(p-1)}\langle|\mathbf{f}|^{q-2}\cdot\mathbf{f},\mathbf{x}\rangle \le \frac{7}{k}\|\mathbf{f}\|_q^{q(p-1)}\cdot\gamma_q(6kp\mathbf{x},|\mathbf{f}|) + \frac{3(6pk)^{pq}}{k}\|\mathbf{x}\|_{pq}^{pq}, \tag{5}$$

where we write $\gamma_q(\mathbf{x},\mathbf{y}) \stackrel{\text{def}}{=} \sum_j \gamma_q(x_j,y_j)$ for vectors $\mathbf{x},\mathbf{y} \in \mathbb{R}^k$.

Using Lemma 8, we can define the *residual function* $\mathcal{R}(\mathbf{x};\mathbf{f})$ that upper bounds the difference in the objective, i.e., $\mathcal{R}(\mathbf{x};\mathbf{f}) \ge \|\mathbf{f}+\mathbf{x}\|_q^{pq} - \|\mathbf{f}\|_q^{pq}$.

▶ **Definition 9** (Residual Problem)**.** *Given two vectors $\mathbf{f},\mathbf{x} \in \mathbb{R}^k$, we define $\mathcal{R}(\mathbf{x};\mathbf{f})$ as*

$$\mathcal{R}(\mathbf{x};\mathbf{f}) \stackrel{\text{def}}{=} pq\|\mathbf{f}\|_q^{q(p-1)}\langle|\mathbf{f}|^{q-2}\cdot\mathbf{f},\mathbf{x}\rangle + \frac{7}{k}\|\mathbf{f}\|_q^{q(p-1)}\cdot\gamma_q(6kp\mathbf{x},|\mathbf{f}|) + \frac{3(6pk)^{pq}}{k}\|\mathbf{x}\|_{pq}^{pq}.$$

In the setting of Problem 1, given a feasible flow $\mathbf{F} \in \mathbb{R}^{E\times k}$, we define the residual problem *w.r.t. $\mathbf{F}$ as follows*

$$\min_{\mathbf{X}\in\mathbb{R}^{E\times k}} \mathcal{R}(\mathbf{X};\mathbf{F}) \stackrel{\text{def}}{=} \sum_{e\in E} w_e^{pq}\mathcal{R}(\mathbf{x}^e;\mathbf{f}^e) \quad \text{s.t. } \mathbf{B}^\top\mathbf{X} = \mathbf{0}. \tag{6}$$

Note that in the residual problem, the objective is decomposable for each coordinate (e,j). The decomposability allows us to use the almost-linear time convex flow solver (Theorem 2) to solve the residual problem to high accuracy. The following lemma summarizes the algorithm and the proof is deferred to Section 4.

▶ **Lemma 10.** *Given any feasible flow $\mathbf{F} \in \mathbb{R}^{E\times k}$ to Problem (1), there is a randomized algorithm that runs in time $km^{1+o(1)}$ and outputs, for any $C > 0$, a k-commodity circulation $\mathbf{X}$ such that*

$$\mathcal{R}(\mathbf{X};\mathbf{F}) \le \min_{\mathbf{B}^\top\mathbf{X}^*=\mathbf{0}} \mathcal{R}(\mathbf{X}^*;\mathbf{F}) + \exp\left(-\log^C m\right).$$

Now, we are ready to prove Theorem 1 with the following Algorithm 1. The algorithm starts with some initial flow and runs in T iterations. Each iteration, the algorithm updates the flow $\mathbf{F}^{(t+1)} \leftarrow \mathbf{F}^{(t)} + \mathbf{X}^{(t)}$ with a near-optimal solution to the residual problem. In other words, given a current flow $\mathbf{F}$, the algorithm updates the flow by solving a simpler residual problem which is an upper bound to the change in objective value.

To analyze Algorithm 1 and prove Theorem 1, we first relate the value of $\mathcal{R}(\mathbf{X};\mathbf{F})$ to the change in objective value when updating $\mathbf{F}$ with $\mathbf{X}$.

▶ **Lemma 11.** *For any $\mathbf{F},\mathbf{X} \in \mathbb{R}^{E\times k}$, we have*

$$\mathcal{E}(\mathbf{F}+\mathbf{X}) - \mathcal{E}(\mathbf{F}) \le \mathcal{R}(\mathbf{X};\mathbf{F}) \text{ and} \tag{7}$$

$$\mathcal{E}(\mathbf{F}+\lambda\mathbf{X}) - \mathcal{E}(\mathbf{F}) \ge \lambda\mathcal{R}(\mathbf{X};\mathbf{F}), \text{ where } \lambda \stackrel{\text{def}}{=} O\left(kp\left(\frac{4032}{q-1}\right)^{\frac{1}{q-1}}\right). \tag{8}$$

Algorithm 1 High-Accuracy $\ell_{q,p}$-norm Multicommodity Flow.

procedure *LqpNormFlow*$(G, \mathbf{w}, \mathbf{D}, q, p)$
  Initialize the flow $\mathbf{F}^{(0)} \in \mathbb{R}^{E\times k}$ such that $\mathbf{B}^\top \mathbf{F}^{(0)} = \mathbf{D}$ via Lemma 13.
  $T \leftarrow O(p\lambda \log(m/\varepsilon))$, where λ comes from Lemma 11
  for $t = 0$ **to** $T-1$ **do**
    Solve the residual problem (6) w.r.t. $\mathbf{F}^{(t)}$ via Lemma 10 and obtain the solution $\mathbf{X}^{(t)}$.
    $\mathbf{F}^{(t+1)} \leftarrow \mathbf{F}^{(t)} + \mathbf{X}^{(t)}$
  end
  return $\mathbf{F}^{(T)}$

Proof. By the definition of $\mathcal{R}(\mathbf{x})$ and Lemma 8, the Inequality (7) holds trivially. We now focus on proving Inequality (8). In particular, we show that for any vector $\mathbf{f}, \mathbf{x} \in \mathbb{R}^k$, the following

$$\|\mathbf{f}+\lambda\mathbf{x}\|_q^{pq} - \|\mathbf{f}\|_q^{pq} \geq \lambda\mathcal{R}(\mathbf{f};\mathbf{x}). \tag{9}$$

Inequality (9) implies Inequality (8) by taking the summation over all edges.

Lemma 8 gives

$$\begin{aligned}&\|\mathbf{f}+\lambda\mathbf{x}\|_q^{pq} - \|\mathbf{f}\|_q^{pq}\\ &\geq pq\|\mathbf{f}\|_q^{(p-1)q}\langle|\mathbf{f}|^{q-2}\cdot\mathbf{f}, \lambda\mathbf{x}\rangle + \frac{p(q-1)}{16}\|\mathbf{f}\|_q^{(p-1)q}\cdot\gamma_q(\lambda\mathbf{x}, |\mathbf{f}|) + \frac{q-1}{pq-1}\frac{1}{2^{pq+2}}\|\lambda\mathbf{x}\|_{pq}^{pq}.\end{aligned}$$

In addition, we have

$$\lambda\mathcal{R}(\mathbf{x}) = \lambda pq\|\mathbf{f}\|_q^{(p-1)q}\langle|\mathbf{f}|^{q-2}\cdot\mathbf{f}, \mathbf{x}\rangle + \frac{7\lambda}{k}\|\mathbf{f}\|_q^{(p-1)q}\cdot\gamma_q(6kp\mathbf{x}, |\mathbf{f}|) + \frac{3\lambda(6pk)^{pq}}{k}\|\mathbf{x}\|_{pq}^{pq}.$$

In order to prove (9), it suffices to ensure that

$$\begin{aligned}\frac{p(q-1)}{16}\|\mathbf{f}\|_q^{(p-1)q}\cdot\gamma_q(\lambda\mathbf{x}, |\mathbf{f}|) &\geq \frac{7\lambda}{k}\|\mathbf{f}\|_q^{(p-1)q}\cdot\gamma_q(6kp\mathbf{x}, |\mathbf{f}|),\\ \frac{q-1}{pq-1}\frac{1}{2^{pq+2}}\|\lambda\mathbf{x}\|_{pq}^{pq} &\geq \frac{3\lambda(6pk)^{pq}}{k}\|\mathbf{x}\|_{pq}^{pq}.\end{aligned}$$

We can consider each entry separately and (9) follows if for any $f, x \in \mathbb{R}$, we have

$$\frac{p(q-1)}{16}\cdot\gamma_q(\lambda x, |f|) \geq \frac{7\lambda}{k}\cdot\gamma_q(6kpx, |f|), \tag{10}$$

$$\frac{q-1}{pq-1}\frac{1}{2^{pq+2}}|\lambda x|^{pq} \geq \frac{3\lambda(6pk)^{pq}}{k}|x|^{pq}. \tag{11}$$

Inequality (10) follows from Lemma 7

$$\frac{p(q-1)}{16}\cdot\gamma_q(\lambda x, |f|) \geq \frac{p(q-1)}{16}\left(\frac{\lambda}{6kp}\right)^q\cdot\gamma_q(6kpx, |f|) \geq \frac{7\lambda}{k}\cdot\gamma_q(6kpx, |f|),$$

if we set

$$\lambda \geq kp\left(\frac{4032}{q-1}\right)^{\frac{1}{q-1}} \geq 6kp. \tag{12}$$

For Inequality (11) to hold, we need to set

$$\lambda \geq 1728k \left(\frac{pq-1}{q-1}\right)^{\frac{1}{pq-1}} p^{\frac{pq}{pq-1}}. \tag{13}$$

Observe that

$$\left(\frac{pq-1}{q-1}\right)^{\frac{1}{pq-1}} = (pq-1)^{\frac{1}{pq-1}} \left(\frac{1}{q-1}\right)^{\frac{1}{pq-1}} \leq e \left(\frac{1}{q-1}\right)^{\frac{1}{q-1}}, \text{ and}$$
$$p^{\frac{pq}{pq-1}} = p^{1+\frac{1}{pq-1}} \leq p^{1+\frac{1}{p-1}} \leq 2p.$$

Combining the observation along with (12) and (13), Inequality (9) follows if we set

$$\lambda = O\left(kp\left(\frac{4032}{q-1}\right)^{\frac{1}{q-1}}\right).$$

This completes the proof. ◀

Using Lemma 11, we now show that each iteration decreases the objective exponentially. This is summarized by the following lemma

▶ **Lemma 12** (Convergence Rate). *Let $\mathbf{F}^*$ be the optimal solution to Problem (1). At any iteration t of Algorithm 1, we have*

$$\mathcal{E}(\mathbf{F}^{(t+1)}) - \mathcal{E}(\mathbf{F}^*) \leq \left(1 - \frac{1}{\lambda}\right)\left(\mathcal{E}(\mathbf{F}^{(t)}) - \mathcal{E}(\mathbf{F}^*)\right) + \exp(-\log^C m).$$

Proof. Recall that $\mathbf{F}^{(t+1)} = \mathbf{F}^{(t)} + \mathbf{X}^{(t)}$ and $\mathbf{X}^{(t)}$ is a high-accuracy solution to the residual problem w.r.t. $\mathbf{F}^{(t)}$. Lemma 11 and the optimality of $\mathbf{X}^{(t)}$ yields

$$\begin{aligned}\mathcal{E}(\mathbf{F}^{(t+1)}) - \mathcal{E}(\mathbf{F}^{(t)}) &\leq \mathcal{R}(\mathbf{X}^{(t)}; \mathbf{F}^{(t)}) \\ &\leq \mathcal{R}\left(\lambda^{-1}\left(\mathbf{F}^* - \mathbf{F}^{(t)}\right); \mathbf{F}^{(t)}\right) + \exp(-\log^C m) \\ &\leq \frac{1}{\lambda}\left(\mathcal{E}(\mathbf{F}^*) - \mathcal{E}(\mathbf{F}^{(t)})\right) + \exp(-\log^C m).\end{aligned} \tag{14}$$

We can conclude the lemma as follows:

$$\begin{aligned}\mathcal{E}(\mathbf{F}^{(t+1)}) - \mathcal{E}(\mathbf{F}^*) &= \mathcal{E}(\mathbf{F}^{(t+1)}) - \mathcal{E}(\mathbf{F}^{(t)}) + \mathcal{E}(\mathbf{F}^{(t)}) - \mathcal{E}(\mathbf{F}^*) \\ &\leq \frac{1}{\lambda}\left(\mathcal{E}(\mathbf{F}^*) - \mathcal{E}(\mathbf{F}^{(t)})\right) + \exp(-\log^C m) + \mathcal{E}(\mathbf{F}^{(t)}) - \mathcal{E}(\mathbf{F}^*) \\ &= \left(1 - \frac{1}{\lambda}\right)\left(\mathcal{E}(\mathbf{F}^{(t)}) - \mathcal{E}(\mathbf{F}^*)\right) + \exp(-\log^C m).\end{aligned}$$ ◀

Then, we show how to find an initial solution efficiently.

▶ **Lemma 13** (Initial Flow). *Given an instance of Problem (1), there is an algorithm that runs in $\widetilde{O}(pmk)$ time and finds a feasible flow $\mathbf{F}^{(0)} \in \mathbb{R}^{E\times k}$ such that $\mathcal{E}(\mathbf{F}^{(0)}) \leq 4m^{p+1}\mathcal{E}(\mathbf{F}^*)$, where $\mathbf{F}^*$ is the optimal solution.*

Proof. Let flow $\mathbf{F}^{(0)}$ be a $(1+1/pq)$-approximate maximum concurrent flow on $(G, \mathbf{w})$ that routes the demand $\mathbf{D}$. That is, $\mathbf{F}^{(0)}$ is a $(1+1/pq)$-approximate solution to the following problem

$$\min_{\mathbf{B}^\top \mathbf{F} = \mathbf{D}} \max_{e\in E} w_e \sum_{j=1}^{k} |F_{ej}|.$$

Moreover, $\mathbf{F}^{(0)}$ can be computed in $\widetilde{O}(pmk)$ time using the algorithm from [66]. Set $\mathbf{F} = \mathbf{F}^{(0)}$ and let $\mathbf{G} = \mathbf{F}^*$ be the optimal solution to Problem (1). We can view $\mathbf{G}$ as a collection of k single commodity flows and the approximation guarantee of $\mathbf{F}$ yields

$$\max_{e\in E} w_e \left\|\mathbf{f}^e\right\|_1 \leq \left(1+\frac{1}{pq}\right) \max_{e\in E} w_e \left\|\mathbf{g}^e\right\|_1 . \tag{15}$$

We now analyze the approximation ratio of $\mathbf{F}$. Recall the fact that for any vector $\mathbf{x} \in \mathbb{R}^k$, we have $\|\mathbf{x}\|_1 \leq m^{1-1/q} \|\mathbf{x}\|_q$. Combining the observation with Inequality (15), we have

$$\begin{aligned}
\mathcal{E}(\mathbf{G}) \leq \mathcal{E}(\mathbf{F}) &= \sum_{e\in E} w_e^{pq} \left\|\mathbf{f}^e\right\|_q^{pq} \\
&\leq \sum_{e\in E} w_e^{pq} \left\|\mathbf{f}^e\right\|_1^{pq} \leq m \left(\max_{e\in E} w_e \left\|\mathbf{f}^e\right\|_1\right)^{pq} \\
&\leq m\left(1+\frac{1}{pq}\right)^{pq} \left(\max_{e\in E} w_e \left\|\mathbf{g}^e\right\|_1\right)^{pq} \\
&\leq 4m \sum_{e\in E} w_e^{pq} \left\|\mathbf{g}^e\right\|_1^{pq} \leq 4m^{1+pq-p} \sum_{e\in E} w_e^{pq} \left\|\mathbf{g}^e\right\|_q^{pq} \\
&\leq 4m^{p+1}\mathcal{E}(\mathbf{G}).
\end{aligned}$$

◀

We use Lemma 12 to analyze the correctness of the algorithm and prove Theorem 1.

Proof of Theorem 1. After $T = O(p\lambda \log(m/\varepsilon))$ iterations, we use Lemma 12 to bound the objective of the final flow $\mathbf{F}^{(T)}$ output by Algorithm 1 as follows:

$$\begin{aligned}
\mathcal{E}(\mathbf{F}^{(T)}) - \mathcal{E}(\mathbf{F}^*) &\leq \left(1-\frac{1}{\lambda}\right)^T \left(\mathcal{E}(\mathbf{F}^{(0)}) - \mathcal{E}(\mathbf{F}^*)\right) + \exp(-\log^C m) \cdot \sum_{t=0}^{T-1} \left(1-\frac{1}{\lambda}\right)^t \\
&\leq \exp\left(-\frac{T}{\lambda}\right) \left(\mathcal{E}(\mathbf{F}^{(0)}) - \mathcal{E}(\mathbf{F}^*)\right) + \lambda \cdot \exp(-\log^C m) \\
&\leq \varepsilon \mathcal{E}(\mathbf{F}^*) + \varepsilon,
\end{aligned}$$

where the last inequality comes from $\mathcal{E}(\mathbf{F}^{(0)}) - \mathcal{E}(\mathbf{F}^*) \leq 4m^{p+1}\mathcal{E}(\mathbf{F}^*)$ and the sufficiently small constant C in Lemma 10. Rearrangement yields that $\mathcal{E}(\mathbf{F}^{(T)})$ is at most $(1+\varepsilon)\mathcal{E}(\mathbf{F}^*) + \varepsilon$.

We now analyze the runtime. Initialization of $\mathbf{F}^{(0)}$ takes $\widetilde{O}(pmk)$ time by Lemma 13. Each iteration takes $km^{1+o(1)}$ time due to Lemma 10 and there are $T = \widetilde{O}(p\lambda \log(1/\varepsilon))$ iterations. Then the runtime bound follows. ◀

4 Solving the Residual Problem

In this section, we prove Lemma 10.

▶ **Lemma 10.** *Given any feasible flow $\mathbf{F} \in \mathbb{R}^{E\times k}$ to Problem (1), there is a randomized algorithm that runs in time $km^{1+o(1)}$ and outputs, for any $C > 0$, a k-commodity circulation $\mathbf{X}$ such that*

$$\mathcal{R}(\mathbf{X};\mathbf{F}) \leq \min_{\mathbf{B}^\top \mathbf{X}^* = \mathbf{0}} \mathcal{R}(\mathbf{X}^*;\mathbf{F}) + \exp\left(-\log^C m\right).$$

At a high level, we will show that solving the residual problem is equivalent to solving k instances of single commodity convex flow problems. Each of them can be solved in $m^{1+o(1)}$ time via Theorem 2. In particular, we need to construct computationally efficient barriers for the edge cost functions.

Proof of Lemma 10. Recall the residual problem

$$\begin{aligned} &\min_{\mathbf{X}\in\mathbb{R}^{E\times k}} \mathcal{R}(\mathbf{X};\mathbf{F}) \\ &\quad \text{s.t. } \mathbf{B}^\top\mathbf{X} = \mathbf{0}. \end{aligned} \tag{16}$$

From Definition 9, we know

$$\mathcal{R}(\mathbf{X};\mathbf{F}) = \sum_{e\in E}\sum_{j=1}^{k} w_e^{pq}\cdot\left(pq\|\mathbf{f}^e\|_q^{q(p-1)}|F_{ej}|^{q-2}F_{ej}X_{ej} + \frac{7}{k}\|\mathbf{f}^e\|_q^{q(p-1)}\gamma_q(6kpX_{ej},|F_{ej}|) + \frac{3(6pk)^{pq}}{k}|X_{ej}|^{pq}\right).$$

For each commodity $j\in[k]$, we write $\mathbf{x}_j$ and $\mathbf{f}_j$ to be the j-th column of $\mathbf{X}$ and $\mathbf{F}$ respectively, i.e., the flow that routes the commodity j. The constraint $\mathbf{B}^\top\mathbf{X}=\mathbf{0}$ is equivalent to $\mathbf{B}^\top\mathbf{x}_j=\mathbf{0}$ for each j. Thus, (16) is equivalent to solving, for each commodity j, the following single commodity flow problem:

$$\begin{aligned} &\min_{\mathbf{x}\in\mathbb{R}^E} \sum_{e\in E} w_e^{pq}\left(pq\|\mathbf{f}^e\|_q^{q(p-1)}|F_{ej}|^{q-2}F_{ej}x_e + \frac{7}{k}\|\mathbf{f}^e\|_q^{q(p-1)}\gamma_q(6kpx_e,|F_{ej}|) + \frac{3(6pk)^{pq}}{k}|x_e|^{pq}\right) \\ &\text{s.t. } \mathbf{B}^\top\mathbf{x}=\mathbf{0}. \end{aligned} \tag{17}$$

To use Theorem 2 to solve (17), we need to show that the objective is a sum of convex edge costs with efficient barriers. For each edge e and commodity j, we define the cost $c_{ej}(x)$ to be

$$\begin{aligned} &c_{ej}(x) = A_{ej}x + B_{ej}\gamma_q(6kpx,|F_{ej}|) + C_{ej}|x|^{pq}, \text{ where} \\ &\quad A_{ej} \stackrel{\text{def}}{=} w_e^{pq}\cdot pq\|\mathbf{f}^e\|_q^{q(p-1)}|F_{ej}|^{q-2}F_{ej}, \\ &\quad B_{ej} \stackrel{\text{def}}{=} w_e^{pq}\cdot\frac{7}{k}\|\mathbf{f}^e\|_q^{q(p-1)}, \\ &\quad C_{ej} \stackrel{\text{def}}{=} w_e^{pq}\cdot\frac{3(6pk)^{pq}}{k}. \end{aligned}$$

Clearly $c_{ej}(x)$ is a convex function and the objective of (17) is exactly $\sum_e c_{ej}(x_e)$.

We now present computationally efficient barriers for each $c_{ej}(x)$ and show that $c_{ej}(x)$ satisfies Assumption 1. For clarity, we ignore both subscripts e and j and use f to denote $|F_{ej}|$. That is, we will show that the following function satisfies Assumption 1 for any positive $A,B,C,f>0$,

$$c(x) = A\cdot x + B\cdot\gamma_q(6kpx;f) + C\cdot|x|^{pq}.$$

Item 1 holds because $p=\widetilde{O}(1)$.

We now show Item 2 using Theorem 9.1.1 from [55]. It says that the function $\psi_c(x,y) = -\ln(y-c(x))$ is a 1-self-concordance barrier for the epigraph $\mathcal{D}_c \stackrel{\text{def}}{=} \{(x,y)|c(x)\le y\}$ if there is some $\beta>0$ such that $|c'''(x)|\le 3\beta|c''(x)/x|$ holds for all $x\in\mathbb{R}$. The derivatives of $c(x)$ have different forms depending on the value of x due to the γ_q function.

- If $|6kpx|\le f$, we know $\gamma_q(6kpx;f) = \frac{q}{2}f^{q-2}(6kp)^2x^2$ and

$$\begin{aligned} c'(x) &= A + Bqf^{q-2}(6kp)^2x + Cpq|x|^{pq-1}\mathrm{sgn}(x), \\ c''(x) &= Bqf^{q-2}(6kp)^2 + Cpq(pq-1)|x|^{pq-2}, \\ c'''(x) &= Cpq(pq-1)(pq-2)|x|^{pq-3}\mathrm{sgn}(x). \end{aligned}$$

In this case, we have

$$\left|\frac{c''(x)}{x}\right| = \frac{Bqf^{q-2}(6kp)^2}{|x|} + Cpq(pq-1)|x|^{pq-3} \geq \frac{1}{pq-2}|c'''(x)|.$$

It suffices to set $\beta \geq (pq-2)/3$.

- Otherwise, $|6kpx| > f$. We know $\gamma_q(6kpx; f) = |6kpx|^q - (1-\frac{q}{2})f^q$, and

$$\begin{aligned}
c'(x) &= A + B(6kp)^q q|x|^{q-1}\mathrm{sgn}(x) + Cpq|x|^{pq-1}\mathrm{sgn}(x),\\
c''(x) &= B(6kp)^q q(q-1)|x|^{q-2} + Cpq(pq-1)|x|^{pq-2},\\
c'''(x) &= B(6kp)^q q(q-1)(q-2)|x|^{q-3}\mathrm{sgn}(x) + Cpq(pq-1)(pq-2)|x|^{pq-3}\mathrm{sgn}(x).
\end{aligned}$$

In this case, notice that $q-2<0$ and we have

$$\begin{aligned}
\left|\frac{c''(x)}{x}\right| &= B(6kp)^q q(q-1)|x|^{q-3} + Cpq(pq-1)|x|^{pq-3}, \text{ and}\\
|c'''(x)| &= \left|B(6kp)^q q(q-1)(q-2)|x|^{q-3} + Cpq(pq-1)(pq-2)|x|^{pq-3}\right|\\
&\leq B(6kp)^q q(q-1)(2-q)|x|^{q-3} + Cpq(pq-1)(pq-2)|x|^{pq-3}.
\end{aligned}$$

Setting $\beta \geq \max\{2-q, pq-2\}/3$ yields that

$$3\beta\left|\frac{c''(x)}{x}\right| \geq B(6kp)^q q(q-1)(2-q)|x|^{q-3} + Cpq(pq-1)(pq-2)|x|^{pq-3} \geq |c'''(x)|.$$

We conclude Item 2 with the barrier function $\psi_c(y,x) = -\ln(y - c(x))$. Item 3 follows directly from Item 1, and Item 4 follows using the explicit formula for $c(x), c'(x)$, and $c''(x)$.

Now we can apply Theorem 2 to compute, for each j, a flow $\mathbf{x}_j$ in time $m^{1+o(1)}$ such that

$$c(\mathbf{x}_j) \leq \min_{\mathbf{B}^\top \mathbf{x}^* = \mathbf{0}} c(\mathbf{x}^*) + \exp(-\log^C m).$$

Let $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_k]$, then we have $\mathbf{B}^\top \mathbf{X} = \mathbf{0}$ and $\mathbf{X}$ is optimal to (16) up to a $k\exp(-\log^C m)$-additive error, i.e.,

$$\mathcal{R}(\mathbf{X}; \mathbf{F}) \leq \min_{\mathbf{B}^\top \mathbf{X}^* = \mathbf{0}} \mathcal{R}(\mathbf{X}^*; \mathbf{F}) + k\exp(-\log^C m).$$

The total runtime is $k \cdot m^{1+o(1)}$ since we compute $\mathbf{x}_j$ for each $j \in [k]$. ◀

References

1. Deeksha Adil, Brian Bullins, Rasmus Kyng, and Sushant Sachdeva. Almost-linear-time weighted ℓ_p-norm solvers in slightly dense graphs via sparsification. *arXiv preprint*, 2021. arXiv:2102.06977.
2. Deeksha Adil, Rasmus Kyng, Richard Peng, and Sushant Sachdeva. Iterative refinement for ℓ_p-norm regression. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1405–1424, 2019.
3. Deeksha Adil, Rasmus Kyng, Richard Peng, and Sushant Sachdeva. Fast algorithms for ℓ_p-regression. *arXiv preprint*, 2022. arXiv:2211.03963.
4. Deeksha Adil, Richard Peng, and Sushant Sachdeva. Fast, provably convergent IRLS algorithm for p-norm linear regression. *Advances in Neural Information Processing Systems (NeurIPS)*, 32, 2019.

5 Deeksha Adil and Sushant Sachdeva. Faster p-norm minimizing flows, via smoothed q-norm problems. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 892–910, 2020.

6 Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network flows: theory, algorithms, and applications*. Prentice-Hall, Inc., USA, 1993.

7 Kyriakos Axiotis, Aleksander Mądry, and Adrian Vladu. Circulation control for faster minimum cost flow in unit-capacity graphs. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 93–104, 2020.

8 Kyriakos Axiotis, Aleksander Mądry, and Adrian Vladu. Faster sparse minimum cost flow by electrical flow localization. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 528–539, 2022.

9 Cynthia Barnhart, Niranjan Krishnan, and Pamela H Vance. Multicommodity flow problems. *Encyclopedia of Optimization*, 14:2354–2362, 2009.

10 Aaron Bernstein, Maximilian Probst Gutenberg, and Thatchaphol Saranurak. Deterministic decremental SSSP and approximate min-cost flow in almost-linear time. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1000–1008, 2022.

11 Daniel Bienstock and Garud Iyengar. Solving fractional packing problems in $\widetilde{O}(1/\varepsilon)$ iterations. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing (STOC)*, pages 146–155, 2004.

12 Jan van den Brand, Yin-Tat Lee, Danupon Nanongkai, Richard Peng, Thatchaphol Saranurak, Aaron Sidford, Zhao Song, and Di Wang. Bipartite matching in nearly-linear time on moderately dense graphs. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 919–930, 2020.

13 Jan van den Brand and Daniel Zhang. Faster high accuracy multi-commodity flow from single-commodity techniques. *arXiv preprint*, 2023. `arXiv:2304.12992`.

14 Sébastien Bubeck, Michael B Cohen, Yin Tat Lee, and Yuanzhi Li. An homotopy method for lp regression provably beyond self-concordance and in input-sparsity time. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 1130–1137, 2018.

15 Li Chen, Rasmus Kyng, Yang P Liu, Richard Peng, Maximilian Probst Gutenberg, and Sushant Sachdeva. Maximum flow and minimum-cost flow in almost-linear time. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 612–623, 2022.

16 Li Chen, Richard Peng, and Di Wang. 2-norm flow diffusion in near-linear time. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 540–549, 2022.

17 Paul Christiano, Jonathan A Kelner, Aleksander Madry, Daniel A Spielman, and Shang-Hua Teng. Electrical flows, laplacian systems, and faster approximation of maximum flow in undirected graphs. In *Proceedings of the forty-third annual ACM symposium on Theory of computing (STOC)*, pages 273–282, 2011.

18 Kenneth Clarkson, Ruosong Wang, and David Woodruff. Dimensionality reduction for tukey regression. In *International Conference on Machine Learning (ICML)*, pages 1262–1271, 2019.

19 Kenneth L Clarkson, Petros Drineas, Malik Magdon-Ismail, Michael W Mahoney, Xiangrui Meng, and David P Woodruff. The fast cauchy transform and faster robust linear regression. *SIAM Journal on Computing*, 45(3):763–810, 2016.

20 Kenneth L Clarkson and David P Woodruff. Low-rank approximation and regression in input sparsity time. *Journal of the ACM (JACM)*, 63(6):1–45, 2017.

21 Michael B Cohen, Yin Tat Lee, and Zhao Song. Solving linear programs in the current matrix multiplication time. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 938–942, 2019.

22 Michael B Cohen, Yin Tat Lee, and Zhao Song. Solving linear programs in the current matrix multiplication time. *Journal of the ACM (JACM)*, 68(1):1–39, 2021.

23 Michael B Cohen, Aleksander Mądry, Piotr Sankowski, and Adrian Vladu. Negative-weight shortest paths and unit capacity minimum cost flow in $\widetilde{O}(m^{10/7} \log w)$ time. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 752–771, 2017.

24 Samuel I Daitch and Daniel A Spielman. Faster approximate lossy generalized flow via interior point algorithms. In *Proceedings of the fortieth annual ACM symposium on Theory of computing (STOC)*, pages 451–460, 2008.

25 Anirban Dasgupta, Petros Drineas, Boulos Harb, Ravi Kumar, and Michael W Mahoney. Sampling algorithms and coresets for ℓ_p regression. *SIAM Journal on Computing*, 38(5):2060–2078, 2009.

26 Ming Ding, Rasmus Kyng, and Peng Zhang. Two-commodity flow is equivalent to linear programming under nearly-linear time reductions. In *49th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 54:1–54:19, 2022.

27 Lisa K Fleischer. Approximating fractional multicommodity flow independent of the number of commodities. *SIAM Journal on Discrete Mathematics*, 13(4):505–520, 2000.

28 Sebastian Forster, Gramoz Goranci, Yang P Liu, Richard Peng, Xiaorui Sun, and Mingquan Ye. Minor sparsifiers and the distributed laplacian paradigm. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 989–999, 2022.

29 Yu Gao, Yang P Liu, and Richard Peng. Fully dynamic electrical flows: Sparse maxflow faster than goldberg-rao. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 516–527, 2022.

30 Naveen Garg and Jochen Könemann. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. *SIAM Journal on Computing*, 37(2):630–652, 2007.

31 Mehrdad Ghadiri, Richard Peng, and Santosh S Vempala. Faster p-norm regression using sparsity. *arXiv preprint arXiv:2109.11537*, 2021.

32 Andrew V Goldberg. A natural randomization strategy for multicommodity flow and related algorithms. *Information Processing Letters*, 42(5):249–256, 1992.

33 Michael D Grigoriadis and Leonid G Khachiyan. Fast approximation schemes for convex programs with many blocks and coupling constraints. *SIAM Journal on Optimization*, 4(1):86–107, 1994.

34 Michael D Grigoriadis and Leonid G Khachiyan. Approximate minimum-cost multicommodity flows in $\widetilde{O}(\varepsilon^{-2}knm)$ time. *Mathematical Programming*, 75(3):477–482, 1996.

35 Alon Itai. Two-commodity flow. *Journal of the ACM (JACM)*, 25(4):596–611, 1978.

36 Sanjiv Kapoor and Pravin M Vaidya. Speeding up karmarkar's algorithm for multicommodity flows. *Mathematical programming*, 73:111–127, 1996.

37 George Karakostas. Faster approximation schemes for fractional multicommodity flow problems. In *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 166–173, 2002.

38 David Karger and Serge Plotkin. Adding multiple cost constraints to combinatorial optimization problems, with applications to multicommodity flows. In *Proceedings of the twenty-seventh annual ACM symposium on Theory of computing (STOC)*, pages 18–25, 1995.

39 Narendra Karmarkar. A new polynomial-time algorithm for linear programming. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing (STOC)*, pages 302–311, 1984.

40 Tarun Kathuria, Yang P. Liu, and Aaron Sidford. Unit capacity maxflow in almost $O(m^{4/3})$ time. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 119–130, 2020.

41 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 *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms (SODA)*, pages 217–226, 2014.

42 Jonathan A Kelner, Gary L Miller, and Richard Peng. Faster approximate multicommodity flow using quadratically coupled flows. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing (STOC)*, pages 1–18, 2012.

43 Jeff L Kennington. A survey of linear cost multicommodity network flows. *Operations Research*, 26(2):209–236, 1978.

44 Leonid G Khachiyan. Polynomial algorithms in linear programming. *USSR Computational Mathematics and Mathematical Physics*, 20(1):53–72, 1980.

45 Philip Klein, Serge Plotkin, Clifford Stein, and Eva Tardos. Faster approximation algorithms for the unit capacity concurrent flow problem with applications to routing and finding sparse cuts. *SIAM Journal on Computing*, 23(3):466–487, 1994.

46 Rasmus Kyng, Richard Peng, Sushant Sachdeva, and Di Wang. Flows in almost linear time via adaptive preconditioning. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 902–913, 2019.

47 Yin Tat Lee and Aaron Sidford. Solving linear programs with sqrt (rank) linear system solves. *arXiv preprint*, 2019. `arXiv:1910.08033`.

48 T. Leighton, F. Makedon, S. Plotkin, C. Stein, E. Tardos, and S. Tragoudas. Fast approximation algorithms for multicommodity flow problems. *Journal of Computer and System Sciences*, 50(2):228–243, 1995.

49 Tom Leighton, Clifford Stein, Fillia Makedon, Éva Tardos, Serge Plotkin, and Spyros Tragoudas. Fast approximation algorithms for multicommodity flow problems. In *Proceedings of the twenty-third annual ACM symposium on Theory of Computing (STOC)*, pages 101–111, 1991.

50 Jason Li. Faster parallel algorithm for approximate shortest path. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 308–321, 2020.

51 Aleksander Madry. Faster approximation schemes for fractional multicommodity flow problems via dynamic graph algorithms. In *Proceedings of the forty-second ACM symposium on Theory of computing (STOC)*, pages 121–130, 2010.

52 Aleksander Madry. Navigating central path with electrical flows: From flows to matchings, and back. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 253–262, 2013.

53 Aleksander Madry. Computing maximum flow with augmenting electrical flows. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 593–602, 2016.

54 Xiangrui Meng and Michael Mahoney. Robust regression on mapreduce. In *International Conference on Machine Learning (ICML)*, pages 888–896, 2013.

55 Arkadi Nemirovski. Interior point polynomial time methods in convex programming. *Lecture notes*, 42(16):3215–3224, 2004.

56 Yurii Nesterov. Fast gradient methods for network flow problems. In *Proceeding of the 20th International Symposium of Mathematical Programming (ISMP)*, 2009.

57 Adamou Ouorou, Philippe Mahey, and J-Ph Vial. A survey of algorithms for convex multicommodity flow problems. *Management science*, 46(1):126–147, 2000.

58 Richard Peng. Approximate undirected maximum flows in $O(m\text{poly}\log(n))$ time. In *Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms (SODA)*, pages 1862–1867, 2016.

59 Richard Peng and Santosh Vempala. Solving sparse linear systems faster than matrix multiplication. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 504–521, 2021.

60 Serge A Plotkin, David B Shmoys, and Éva Tardos. Fast approximation algorithms for fractional packing and covering problems. *Mathematics of Operations Research*, 20(2):257–301, 1995.

61 Tomasz Radzik. Fast deterministic approximation for the multicommodity flow problem. *Mathematical Programming*, 78:43–58, 1996.

62 James Renegar. A polynomial-time algorithm, based on newton's method, for linear programming. *Mathematical programming*, 40(1-3):59–93, 1988.

63 Bruce Rothschild and Andrew Whinston. Feasibility of two commodity network flows. *Operations Research*, 14(6):1121–1129, 1966.

64 Farhad Shahrokhi and David W Matula. The maximum concurrent flow problem. *Journal of the ACM (JACM)*, 37(2):318–334, 1990.

65 Jonah Sherman. Nearly maximum flows in nearly linear time. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 263–269, 2013.

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

67 David B Shmoys. Cut problems and their application to divide-and-conquer. *Approximation algorithms for NP-hard problems*, pages 192–235, 1997.

68 D. Spielman and S. Teng. Nearly linear time algorithms for preconditioning and solving symmetric, diagonally dominant linear systems. *SIAM Journal on Matrix Analysis and Applications*, 35(3):835–885, 2014. Available at http://arxiv.org/abs/cs/0607105.

69 Jan van den Brand, Yu Gao, Arun Jambulapati, Yin Tat Lee, Yang P Liu, Richard Peng, and Aaron Sidford. Faster maxflow via improved dynamic spectral vertex sparsifiers. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 543–556, 2022.

70 Jan van den Brand, Yin Tat Lee, Yang P Liu, Thatchaphol Saranurak, Aaron Sidford, Zhao Song, and Di Wang. Minimum cost flows, MDPs, and ℓ_1-regression in nearly linear time for dense instances. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 859–869, 2021.

71 I-Lin Wang. Multicommodity network flows: A survey, part i: Applications and formulations. *International Journal of Operations Research*, 15(4):145–153, 2018.

72 David Woodruff and Qin Zhang. Subspace embeddings and ℓ_p-regression using exponential random variables. In *Conference on Learning Theory (COLT)*, pages 546–567, 2013.

73 Neal E. Young. Randomized rounding without solving the linear program. In *Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 170–178, 1995.

74 Goran Zuzic, Gramoz Goranci, Mingquan Ye, Bernhard Haeupler, and Xiaorui Sun. Universally-optimal distributed shortest paths and transshipment via graph-based ℓ_1-oblivious routing. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2549–2579, 2022.