Improved Interior Point Algorithms for Exact and Approximate solution of Multicommodity Flow Problems

Anil Kamath[*] ***Omri Palmon***[†]

Abstract

In this paper, we present a new interior-point based polynomial algorithm for the multicommodity flow problem and its variants. Unlike all previously known interior point algorithms for multicommodity flow that have the same complexity for approximate and exact solutions, our algorithm improves running time in the approximate case by a polynomial factor. For many cases, the exact bounds are better as well.

Instead of using the conventional linear programming formulation for the multicommodity flow problem, we model it as a quadratic optimization problem which is solved using interior-point techniques. This formulation allows us to exploit the underlying structure of the problem and to solve it efficiently.

The algorithm is also shown to have improved stability properties. The improved complexity results extend to minimum cost multicommodity flow, concurrent flow and generalized flow problems.

1 Introduction

The multicommodity flow problem is the problem of finding several network flows, each satisfying a demand between given source and sink, such that the total flow on each edge obeys capacity. The multicommodity flow problems arises naturally in many contexts, including virtual circuit routing in communication networks, VLSI layout, scheduling, and transportation, and hence has been extensively studied [4, 6, 16, 9, 14, 15]. Though a natural extension of the single commodity flow problem, all known combinatorial algorithms developed for the single commodity case do not readily extend to the multicommodity flow problem. This is because many of the properties like unimodularity of the constraint matrix in the linear programming formulation and the max-flow min-cut relation do not apply to the multicommodity flow case.

In this paper, we present a new approach to the multicommodity flow problem that is based on the interior point method. Traditional interior-point based algorithms for this problem solve a linear programming (LP) formulation of the problem. The LP formulation has two types of constraints, one to enforce conservation and the second for capacity. The conservation constraints are independent over commodities and the capacity constraints over edges. In this paper, we try to use the separability of the constraints to design an efficient algorithm for the problem. We retain the relatively simpler capacity constraints in the linear form, but define a quadratic objective function to enforce flow conservation. The resulting quadratic programming problem (QP) can then be solved efficiently using interior-point techniques. For a multicommodity flow problem that has n nodes, m edges and k commodities, the new approach has an improved complexity when $m < n^{0.9}k^{0.9}$. Moreover since in our formulation the region of optimization is defined by simple separable linear conditions, we can find a good starting point to the

[*]Department of Computer Science, Stanford University. Research supported by U.S. Army Research Office Grant DAAL-03-91-G-0102, by NSF grant number CCR-9304971 and by Terman Fellowship of Prof. Plotkin.

[†]Department of Computer Science, Stanford University. Research supported by a grant from Stanford Office of Technology Licensing, by NSF grant number CCR-9304971 and by Terman Fellowship of Prof. Plotkin.

paper	exact solution[a]	ϵ approximation[a]
Vaidya [16] with FMM[b]	$k^{2.5}m^{0.5}n^2L^c$	same
Vaidya [16] without FMM	$k^3m^{0.5}n^{2.5}L^c$	same
Murray [11]	$k^{2.5}m^{1.5}nL^c$	same
Leighton et. al. [8], Radzik [13] [d]		$kmn\epsilon^{-2}$
Plotkin et. al. [12] (randomized)		$km^2\epsilon^{-2}$
Plotkin et. al. [12]		$k^2m^2\epsilon^{-2}$
Kamath et. al. [5]		$kmn^2\epsilon^{-6}$
Current paper		
Variant 1	$k^{2.5}m^{1.5}nL^e$	$k^{2.5}m^{1.5}n\log\epsilon^{-1}$
Variant 2 with FMM	$(k^{0.5}m^{2.7}+km^{1.5}n^{1.2}+km^{2.5})L^e$	$(k^{0.5}m^{2.7}+km^{1.5}n^{1.2}+km^{2.5})\log\epsilon^{-1}$
Variant 2 without FMM	$(k^{0.5}m^3+km^{1.5}n^{1.5})L^e$	$(k^{0.5}m^3+km^{1.5}n^{1.5})\log\epsilon^{-1}$

[a]These are asymptotic complexities without log factors
[b]Fast Matrix Multiplication
[c]L not specified.
[d]without costs.
[e]$L = m\log m + \log DU$.

Figure 1: Comparison of Complexities.

algorithm and hence improve the running time by a polynomial factor whenever we need an approximate solution to the problem.

We present two variants of our algorithm. Both variants compute an exact solution in $O(k^{0.5}m^{0.5}L)$ interior point iterations, where $L = m\log m + \log(DU)$, D is the biggest demand and U the biggest capacity . The main step in each interior-point iteration involves solving a linear system of equations. The first variant computes a single iteration in $O(k^2mn)$ time and the second requires $O(m^{2.5}+k^{0.5}n^{1.5}m)$ time. Using fast matrix multiplication the complexity of computing a single iteration in the second variant can be improved to $O(m^{2.2}+k^{0.5}m^2+k^{0.5}mn^{1.2}+kn^2)$. The fastest previously known multicommodity flow algorithm was due to Vaidya [6, 16], and achieves a complexity bound of $O(k^{2.5}m^{0.5}n^2L\log(nDU))$, with fast matrix multiplication techniques and $O(k^3m^{0.5}n^3L\log(nDU))$ without fast matrix multiplication. The complexities per iteration are $O(n^2k^2)$ when using fast matrix multiplication, and $O(n^{2.5}k^{2.5})$ when not. Even though fast matrix multiplication has better asymptotic bounds it is considered to be impractical for any reasonable sized problems. Hence we report the complexities with and without fast matrix multiplications. Our algorithm improves on the previously best known complexity bound when $m < n^{0.9}k^{0.9}$. For example, when the multicommodity flow problem has $k = \Theta(n)$ commodities and $m = \Theta(n)$ edges our algorithm is faster than the best previously known algorithm by a factor of $\Theta(n)$. The same results apply to the minimum cost multicommodity flow and the concurrent flow problems.

Most practical applications of the multicommodity flow problem do not need the exact solution but only need to solve it up to a finite precision. When this precision is small, relative to the problem size, our algorithm is better than previous algorithms by a polynomial factor, in all cases. The improvement in running time is possible because we can compute a good starting point for our algorithm. In previous formulations of multicommodity flow as an LP problem the polytope is defined by both capacity and conservation constraints. Because of the way these constraints interleave it does not seem possible to compute a good starting point for these algorithms. Thus their asymptotic complexities for computing both approximate and exact solutions are the same. On the other hand, the region of optimization in our formulation is defined by only simple separable capacity constraints. Hence it is easier to compute a good starting point from which we can converge faster to an approximate solution. To get the complexities of our algorithms for computing ϵ-approximate solutions, one can replace L (which is at least nk) by $\log nDU\epsilon^{-1}$. For example, in the case described above ($k = \theta(n)$ and $m = \theta(n)$), our complexity is an improvement by a factor of $\theta(n^3/\log\epsilon^{-1})$ over previously known results. Moreover, previously known interior point algorithms for multicommodity flow [6] required to work with $O(L)$ bits. If b bits of precision are needed, then our algorithm needs to work with $O(b + \log(nDU))$ bits of precision.

Recently, several combinatorial methods have been suggested to compute approximate solutions to the multicommodity flow problem [8, 1, 13, 12, 5]. Leighton et. al. [8] provided an algorithm that runs in $O^*(k^2mn\epsilon^{-2})$ deterministic time[1] and $O^*(kmn\epsilon^{-2})$ randomized time. Radzik [13] showed how to derandomize this algorithm without increasing the complexity . Best running times for the minimum cost multicommodity flow problem are $O^*(k^2m^2\epsilon^{-2})$ ([12]) and $O^*(kmn^2\epsilon^{-5})$ ([5]). Though for constant ϵ, these algorithms have better running times, their complexity increases exponentially with the desired precision.

The algorithm extends to the generalized flow problem and the generalized multicommodity flow problem. Our complexity bound for the generalized multicommodity flow is the same (up to log factors) as for the multicommodity flow problem. For the single commodity generalized flow problem, our running time is $O\left(m^{1.5}n^2\log(nUD\gamma)\right)$ (where γ is the biggest ratio between two generalized flow coefficients) which matches the best previously known result in [16] and [11].

Section 2 will describe the formulation of the multicommodity flow as a QP problem. Section 3 will describe the algorithm. Section 4 will describe methods to efficiently solve the linear system that arises in the formulation of the problem. Section 5 will describe how to obtain the starting point to the algorithm. Section 6 will prove the numerical stability of the algorithm. Section 7 will show how an exact solution is obtained. Section 8 will describe how to extend the algorithm to solve the minimum cost, concurrent flow and generalized multicommodity flow problems.

2 Multicommodity flow as QP

An instance of the *multicommodity flow problem* consists of a directed graph $G = (V, E)$, a non-negative capacity $u(e)$ for every edge $e \in E$, and a specification of k commodities, numbered 1 through k, where the specification for commodity i consists of a source-sink pair $s_i, t_i \in V$ and a non-negative demand d_i. We will denote the number of nodes by n, and the number of edges by m. We assume that $m \geq n$, and that the graph G is connected and has no parallel edges. Let U be the biggest capacity and D the biggest demand (all capacities and demands are greater than 1).

In this section, we shall formulate multicommodity flow as a QP problem. The problem will have $m(k+1)$ variables. Define the index set $K \overset{def}{=} \{1, \ldots, k, s\}$. Let x^l_e be the flow of commodity $l = 1, \ldots, k$ on edge e and let x^s_e be the unused capacity (slack variable) for edge e. A feasible multi-commodity flow solution has to obey conservation and capacity constraints.

The capacity constraints are written as linear equations. Define A as an $m(k+1) \times m$ matrix constructed from $k+1$ blocks of identity matrices of dimension $m \times m$:

$$A \quad \overset{def}{=} \quad (I_{m\times m} \;\; I_{m\times m} \quad \cdots I_{m\times m}).$$

Let $\bar{u} \in \mathbb{R}^E$ be the vector of capacities. The capacity constraints can then be written as

$$Ax = \bar{u} \qquad x \geq 0.$$

The conservation constraints are enforced through the objective function. We will define a convex quadratic objective function (positive semi-definite form) which has value 0 if and only if the flow variables obey conservation. Denote the vector of flow variables for the lth commodity by $x^l \in \mathbb{R}^E$. For

[1]We say that $f(n) = O^*(g(n))$ if $f(n) = O(g(n)\log^k n)$ for constant k.

every $v \in V$ define the excess of the lth commodity at vertex v to be

$$Ex_v(x^l) = \sum_{w:wv \in E} x^l_{wv} - \sum_{w:vw \in E} x^l_{vw}.$$

The conservation constraints are enforced by computing the minimum of the quadratic function

$$E(x) \overset{def}{=} \sum_{v \in V, l \leq k} \left(Ex_v(x^l) - h^l(v)\right)^2,$$

where $h^l(s_l) = d_l$, $h^l(t_l) = -d_l$ and $h^l(v) = 0$ otherwise. To write the matrix formalization, we will define B, an $n \times m$ matrix that calculates the excesses of the flow as

$$B_{u,e=vw} = \begin{cases} -1 & u = v \\ 1 & u = w \\ 0 & \text{otherwise.} \end{cases}$$

Now define Q as an $m(k+1) \times m(k+1)$ matrix composed of $(k+1) \times (k+1)$ blocks of size $m \times m$ each:

$$Q \overset{def}{=} \begin{pmatrix} B^TB & 0 & \cdots & 0 & 0 \\ 0 & B^TB & & \vdots & \vdots \\ \vdots & & \ddots & 0 & \vdots \\ 0 & \cdots & 0 & B^TB & 0 \\ 0 & \cdots & \cdots & 0 & 0 \end{pmatrix}$$

Define $c^l_{vw} = h_l(v) - h_l(w)$ for $l \neq s$ and $c^s_{vw} = 0$, then the corresponding QP problem can be written as

$$\min \frac{1}{2}E(x) = c^Tx + \frac{1}{2}x^TQx \qquad s.t.\ Ax = \bar{u} \ \ x \geq 0.$$

A solution to the QP problem that minimizes $E(x)$ subject to the capacity constraints gives a solution to the multi-commodity flow problem. We shall use a primal-dual algorithm to solve this QP problem. The algorithm introduces dual variables, y_e corresponding to the capacity constraints for every edge e and the dual slack variables z^l_e for every edge e and $l \in K$. The primal-dual feasible polytope W is defined as

$$W = \left\{ (x,y,z) \;\middle|\; \begin{array}{c} x \geq 0\ ,\ \ z \geq 0\ ,\ \ Ax = \bar{u}\ , \\ -Qx + A^Ty + z - c = 0 \end{array} \right\}.$$

3 The Algorithm

In this section, we show how to solve the resulting QP problem using interior point techniques. The interior point method, is an iterative method which computes a sequence of points in the interior of the polytope defined by the constraints. The value of the objective function will be improved in each iteration. The algorithm described in this paper is the same as in [10] but has been adapted to exploit the underlying structure of the multicommodity flow problem.

In the feasible set W, we will define the central path $\rho(\mu)$ for every $\mu > 0$ as the point that minimizes

$$E(x) - \mu \sum_{l \in K e \in E} \log x^l_e$$

subject to the capacity constraints. As μ tends to zero, $\rho(\mu)$ converges to the optimal solution. The point $\rho(\mu)$ is completely characterized by the following Karush-Kuhn-Tucker conditions:

$$\begin{array}{ll} \forall e \in E \ \ \forall l \in K & x^l_e > 0 \ \ z^l_e > 0 \\ \forall e \in E \ \ \forall l \in K & x^l_e z^l_e = \mu \\ & Ax = \bar{u} \\ & -Qx + A^Ty + z - c = 0. \end{array}$$

The algorithm constructs a series of approximations to the central path until the current point is sufficiently close to the optimal solution. At each step, given an approximation to a point on the central path $\rho(\mu)$ we will compute an approximation to $\rho(\hat{\mu})$ where $\hat{\mu} = (1 - \delta/\sqrt{mk})\mu$ (δ is a constant). The approximation will be calculated using the Newton direction associated with the Karush-Kuhn-Tucker conditions. The Newton direction $(\Delta x, \Delta y, \Delta z)$ will be approximately computed from the following equations

$$\begin{array}{llrcl} (3.1) & \forall e\ \forall l & \tilde{z}^l_e \Delta x^l_e + \tilde{x}^l_e \Delta z^l_e & = & x^l_e z^l_e - \hat{\mu} \\ & & A\Delta x & = & 0 \\ & & -Q\Delta x + A^T \Delta y + \Delta z & = & 0 \end{array}$$

where $\tilde{x}$ and $\tilde{z}$ are approximations to x and z respectively.

Now we can state the algorithm (compare to algorithm 3.1 in [10]):

Monteiro-Adler Algorithm for QP

Initialize Set $j = 0$. Compute a starting point (x^0, y^0, z^0) and the corresponding parameter μ^0.

Compute while $\mu^j > 0.9\epsilon^2(m(k+1)n)^{-1}$ do

1. Set $\mu^{j+1} = (1 - \delta/\sqrt{mk})\mu^j$.
2. Compute $(\Delta x, \Delta y, \Delta z)$ solving equations (3.1).
3. Set

   $$(x^{j+1}, y^{j+1}, z^{j+1}) = (x^j, y^j, z^j) - (\Delta x, \Delta y, \Delta z)$$

   and set $j = j + 1$.

Round If $E(x) \leq 0.9\epsilon^2(m(k+1)n)^{-1}$ round the solution, otherwise the problem has no feasible solution.

A good starting point is important for proving good complexity bounds in the approximate case and one such point is presented in section 5. The Newton direction is efficiently computed by solving a system of linear equations using one of the procedures described in Section 4. The rounding procedure is outlined in theorem 3.2.

In order to state the theorems that prove the convergence result for this algorithm, we need the following definitions:

Definition 3.1. *A point $w = (x, y, z) \in W$ will be called a θ approximation to the central path $\rho(\cdot)$ at μ if*

$$\sum_{e \in E, l \in K} (x^l_e z^l_e - \mu)^2 \leq \theta^2 \mu^2.$$

Note that the definition of the central path implies that $\rho(\mu)$ is the only point that satisfies $x^l_e z^l_e = \mu$ for all e and l.

Definition 3.2. *A vector $\tilde{v}$ will be called a γ approximation to a vector v if for each i we have*

$$\frac{|\tilde{v}_i - v_i|}{\tilde{v}_i} \leq \gamma.$$

We will also denote by $\|v\|_u$ the Euclidean norm of the vector v normalized by the vector u, i.e.

$$\|v\|_u \overset{def}{=} \left(\sum_i \left(\frac{v_i}{u_i}\right)^2\right)^{1/2}.$$

Using this notation, we can now state the following main theorem that proves that our algorithm can compute approximations to the central path $\rho(\mu)$ for decreasing values of μ.

Theorem 3.1. *Let $\theta = \delta = \gamma = 0.1$. Let $w = (x, y, z) \in W$ and $\mu > 0$ such that w is a θ approximation to the central path at μ. Assume that $\tilde{x}$ and $\tilde{z}$ are γ approximations to x and z. Let $\hat{\mu} = \mu(1 - \delta/\sqrt{mk})$. Consider the point $\hat{w} = w - \Delta w$ where $\Delta w = (\Delta x, \Delta y, \Delta z)$ solves equations (3.1). Then we have*

1. *$\hat{w} \in W$*
2. *$\hat{w}$ is a θ approximation to the central path at $\hat{\mu}$*
3. *$\|\hat{x} - x\|_x \leq 0.28$ and $\|\hat{z} - z\|_z \leq 0.28$.*

Proof. Theorem 3.1 is exactly theorem 4.1 in [10].□

The above theorem implies the convergence of our quadratic optimization method. The next theorem will provide us with a technique for rounding the solution of the quadratic optimization problem to compute an ϵ-approximate solution of the multicommodity flow problem.

Theorem 3.2. *Assume that there is a solution to the multicommodity flow problem. Let $w = (x, y, z) \in W$ be a θ approximation to the central path at $\mu \leq (1 + \theta)^{-1}\epsilon^2(m(k + 1)n)^{-1}$. Then we can obtain in $O(kmn)$ time a solution to the multicommodity flow problem that obeys capacity constraints and for each commodity l, satisfies at least $(1 - \epsilon)$ fraction of the total demand d_l.*

Proof. Since there is a solution to the multicommodity flow problem the optimal value of $E(\cdot)$ is 0. Hence current dual value is non-positive. Thus (see section 2 in [10] for theoretical background)

$$E(x) = (\text{primal value}) \leq (\text{primal value}) - (\text{dual value}) = (\text{primal dual gap}) =$$

$$\sum_{e,l} x_e^l z_e^l \leq m(k + 1)(1 + \theta)\mu \leq \epsilon^2 n^{-1}.$$

Hence for each commodity l, we have

$$\sum_{v \in V} \left(Ex_v(x^l) - h^l(v)\right)^2 \leq \epsilon^2 n^{-1}$$

which implies

$$\left|Ex_{s_l}(x^l) + d_l\right| + \left|Ex_{t_l}(x^l) - d_l\right| + \sum_{v \in V \setminus \{s_l, t_l\}} \left|Ex_v(x^l)\right| \leq \epsilon \leq \epsilon d_l$$

Now solving a single commodity max flow with current flow values as capacities will result in a feasible flow that satisfies at least $(1 - \epsilon)$ fraction of the demand for each commodity.□

We shall now state the main complexity result for our algorithm. The result follows from various theorems that we prove in the following sections.

Theorem 3.3. *We can find an ϵ approximation to the multicommodity flow problem (or determine that no solution exists) in $O(m^{0.5}k^{0.5}\log(n\epsilon^{-1}DU))$ iterations. Each iteration can be done in either $O\left(k^2mn\right)$ or $O\left(m^{2.5} + k^{0.5}mn^{1.5}\right)$ time. The complexity of each iteration can be improved to $O\left(m^{2.2} + k^{0.5}m^1n^{1.2} + k^{0.5}m^2 + kn^2\right)$ using fast matrix multiplication.*

Proof. Using theorem 3.2 and theorem 5.1 that is proved in section 5, we can bound the number of iterations by $O(m^{0.5}k^{0.5}\log(n\epsilon^{-1}DU))$. From theorems 4.1 and 4.2, that are proved in the next section, the overall complexity of the algorithm can be estimated. □

4 Analysis of number of operations per iterations

The dominant factor in the complexity of interior-point algorithms comes from the computation of the Newton step which involves solving the linear system defined in (3.1). In this section, we will describe two ways for using the underlying structure of the problem to efficiently compute the Newton step. In subsection 4.1, we will outline a procedure to solve the equations using conjugate gradient [3]. In subsection 4.2, we will outline a procedure to solve the equations using matrix inversion and rank one updates.

Denote by $X, Z, \tilde{X}, \tilde{Z}$ the diagonal matrices with $x, z, \tilde{x}, \tilde{z}$ on the diagonal respectively. Denote by $\bar{1}$ the vector of ones. A symbol with superscript l will denote the part associated with commodity l (or slack). Using the temporary variables $v^l \in \mathbb{R}^E$ and D^l a diagonal $m \times m$ matrix for every $l \in K$, a generic procedure that uses the matrix structure for solving the linear system (3.1) is outlined below. (compare to the equations on the bottom of page 47 in [10])

Generic Procedure:

1. Compute the temporary variables v^l and D^l for every $l \in K$

$$D^l = \tilde{Z}^l \tilde{X}^{l^{-1}}$$

$$v^l = (X^l Z^l \bar{1} - \hat{\mu} I_{m(k+1)\times m(k+1)})\tilde{X}^{l^{-1}}\bar{1}.$$

2. Using D^l and v^l we compute Δy such that

$$\left((D^s)^{-1} + \sum_{l\in K\setminus\{s\}} \left(D^l + B^T B\right)^{-1}\right)\Delta y \quad = \quad -\left((D^s)^{-1} v^s + \sum_{l\in K\setminus\{s\}} \left(D^l + B^T B\right)^{-1} v^l\right).$$

3. From Δy, v^l and D^l we compute Δx and Δz

$$\forall l \in K \setminus \{s\} \;\; \Delta x^l = \left(D^l + B^T B\right)^{-1}(\Delta y + v^l)$$

$$\Delta x^s = (D^s)^{-1}(\Delta y + v^s)$$

$$\forall l \in K \quad \Delta z^l = v^l - D^l x^l.$$

The following claim will be used extensively in the rest of the section

Claim 4.1. *Given vectors $x \in \mathbb{R}^E$ and $y \in \mathbb{R}^V$ we can calculate Bx and $B^T y$ in $O(m)$ operations.*

Proof. The proof follows immediately from the sparseness of B.□

4.1 Computing the Newton step using conjugate gradient. The dominant factor in the complexity of GENERIC PROCEDURE is in computing Δy in Step 2. It involves solving the linear system,

$$\left(\sum_{l \in K \setminus \{s\}} \left(D^l + B^T B\right)^{-1} + \left(D^s\right)^{-1} \right) \Delta y = b$$

where

$$b = -\left(\left(D^s\right)^{-1} v^s + \sum_{l \in K \setminus \{s\}} \left(D^l + B^T B\right)^{-1} v^l \right).$$

The problem is equivalent to finding $a^l, l \in K$ such that

$$a^1 + \cdots + a^k + a^s = b$$

$$\forall l \in K \setminus \{s\} \quad \left(D^l + B^T B\right) a^l = D^s a^s = \Delta y.$$

We will find these a^l's using conjugate gradient method. For a description of the conjugate gradient method see pages 516–526 in [3].

The procedure for solving the equation is the following:

VARIANT-1:

1. Compute v^l and D^l for all $l \in K$ in the way they are defined. Compute b, using conjugate gradient to compute each of the terms $\left(D^l + B^T B\right)^{-1} v^l$.

2. Define
$$\tilde{D} = \begin{pmatrix} I & I & \cdots & I \\ -D^s & D^1 & & \\ \vdots & & \ddots & \\ -D^s & & & D^k \end{pmatrix}$$
$$\tilde{B} = \begin{pmatrix} 0 & \cdots & \cdots & 0 \\ \vdots & B^T B & & \\ \vdots & & \ddots & \\ 0 & & & B^T B \end{pmatrix}$$
Compute $b' \in \mathbb{R}^{m(k+1)}$ as
$$b' = \left(I + \tilde{D}^{-1}\tilde{B}\right)^T \tilde{D}^{-1} \left(\begin{matrix} b^T & 0 \cdots 0 \end{matrix} \right)^T.$$
where I is $I_{m(k+1) \times m(k+1)}$.

3. Compute $\bar{a} = (a^s, a^1, \ldots, a^k)$ that solves the following equation using conjugate gradient
$$\left(I + \tilde{D}^{-1}\tilde{B}\right)^T \left(I + \tilde{D}^{-1}\tilde{B}\right) \bar{a} = b'.$$
where I is $I_{m(k+1) \times m(k+1)}$.

4. Compute $\Delta y = D^s a^s$.

5. Compute Δx^l and Δz^l, using conjugate gradient to compute Δx^l.

To estimate the complexity of the procedure we need the following claims.

Claim 4.2. *Given $y \in \mathbb{R}^E$ and a diagonal $m \times m$ matrix D, we can compute $\left(D + B^T B\right)^{-1} y$ in $O(mn)$ operations.*

Proof. The solution x should satisfy

$$\left(I_{m\times m} + D^{-\frac{1}{2}} B^T B D^{-\frac{1}{2}}\right)\left(D^{\frac{1}{2}} x\right) = D^{-\frac{1}{2}} y.$$

The matrix $I_{m\times m} + D^{-\frac{1}{2}} B^T B D^{-\frac{1}{2}}$ is self adjoint and positive definite, hence we can use the conjugate gradient method. Since $rank\, B = n$, the conjugate gradient method will converge in n iterations. It follows from claim 4.1 that each iteration (i.e. multiplying a vector by the matrix) can be done in $O(m)$ operations.□

Claim 4.3. *Given $x \in \mathbb{R}^{m(k+1)}$ we can compute $\tilde{D}^{-1}x$ and $\tilde{B}x$ in $O(km)$ operations.*

Proof. We can look at the matrix $\tilde{D}$ as a $(k+1)\times(k+1)$ matrix where each entry is a $m \times m$ diagonal matrix. As a $(k+1)\times(k+1)$ matrix, computing $\tilde{D}^{-1}x$ takes only $O(k)$ "operations" (this is a diagonal matrix with two rank one updates) where each "operation" involves multiplying an m vector by a diagonal $m \times m$ matrix. Hence computing $\tilde{D}^{-1}x$ takes $O(km)$ time.

The complexity of computing $\tilde{B}x$ follows directly from claim 4.1.□

Claim 4.4. *Given $y \in \mathbb{R}^{m(k+1)}$ using conjugate gradient to find the x such that*

$$\left(I_{m(k+1)\times m(k+1)} + \tilde{D}^{-1}\tilde{B}\right)^T \left(I_{m(k+1)\times m(k+1)} + \tilde{D}^{-1}\tilde{B}\right) x = y$$

will converge in $O(kn)$ iterations.

Proof. Since the matrix is self adjoint and positive definite the conjugate gradient method will find a solution. To estimate the number of iterations observe that the matrix can be written as the identity matrix plus three rank n matrices. Hence the claim follows.□

Theorem 4.1. *The complexity of the procedure* Variant-1 *is $O(k^2 mn)$.*

Proof. The bottleneck for computing one iteration is step 3 which can be computed in $O(k^2mn)$ time, because it is involves $O(nk)$ iterations of conjugate gradient, each with complexity of $O(mk)$.

Step 1 can be done in $O(kmn)$ time by claim 4.2. Step 2 can be done in $O(km)$ time by claim 4.3. Step 5 can be done in $O(kmn)$ time by claim 4.2. □

4.2 Computing the Newton step using matrix inversion and rank one updates. In the following procedure, we will show how to efficiently solve the equations using matrix inversions. We will use a standard technique first proposed in [7] to maintain approximate inverses of the matrices involved. The approximation to the inverse matrix is updated using rank one update, when an entry in x or z changes significantly. The convergence of the algorithm is still assured since in theorem 3.1, $\tilde{x}$ and $\tilde{z}$ need only be approximations to x and z.

We will use C^l and C to denote

$$\forall l \in K \setminus \{s\} \quad C^l \overset{def}{=} \left(I_{n\times n} + BD^l B^T\right)^{-1}$$

$$C \overset{def}{=} \left((D^s)^{-1} + \sum_{l \in K \setminus \{s\}} \left(D^l + B^T B\right)^{-1} \right)^{-1}.$$

The matrices C^l, C and D^l will be updated after each update in $\tilde{x}$ and $\tilde{z}$ rather than being computed in each iteration (compare step 5 to section 5 in [10]). Let r denote a constant integer that will be set later to minimize the complexity, i be the iteration counter and let $\gamma = 0.1$. The procedure is the following:

VARIANT-2:

1. If $i \bmod r = 1$ do the following:
   (a) Set $\tilde{x} = x$ and $\tilde{z} = z$. Compute D^l.
   (b) Calculate explicitly for every $l \in K \setminus \{s\}$

$$C^l = \left(I_{n \times n} + B D^{l^{-1}} B^T\right)^{-1}$$

$$\left(D^l + B^T B\right)^{-1} = D^{l^{-1}} - D^{l^{-1}} B^T C^l B D^{l^{-1}}$$

   where C^l is computed using standard matrix inversion techniques and $\left(D^l + B^T B\right)^{-1}$ is computed using the Sherman-Morrison-Woodbury formula.
   (c) Calculate explicitly using results from previous step and standard matrix inversion techniques

$$C \overset{def}{=} \left((D^s)^{-1} + \sum_{l \in K \setminus \{s\}} \left(D^l + B^T B\right)^{-1} \right)^{-1}.$$

2. Compute v^l for every $l \in K$ in the way they are defined.
3. Using the Sherman-Morrison-Woodbury formula compute for $l \in K \setminus \{s\}$

$$\left(D^l + B^T B\right)^{-1} v^l = D^{l^{-1}} v^l - D^{l^{-1}} B^T C^l B D^{l^{-1}} v^l.$$

4. Using the results of the previous step compute

$$\Delta y = -C \left((D^s)^{-1} v^s + \sum_{l \in K \setminus \{s\}} \left(D^l + B^T B\right)^{-1} v^l \right).$$

   Compute Δx^l and Δz^l as in Step 3 of the GENERIC PROCEDURE. Use the Sherman-Morrison-Woodbury formula as in step 3 when needed.
5. Set $x = x - \Delta x$ and $z = z - \Delta z$. For every $e \in E$ and $l \in K$ if either $\left|x_e^l - \tilde{x}_e^l\right| / \tilde{x}_e^l > \gamma$ or $\left|z_e^l - \tilde{z}_e^l\right| / \tilde{z}_e^l > \gamma$ set $\tilde{x}_e^l = x_e^l$ and $\tilde{z}_e^l = z_e^l$, and update D^l, C^l and C using rank one updates.

CLAIM 4.5. *The complexity of step 1 is $O(kn^3 + km^2 + m^3)$.*

Proof. Because of the structure of B as vertex-edge adjacency matrix each entry of $BD^{l^{-1}}B^T$ can be computed by at most one subtraction depending on whether there is an edge between the two vertices that define the entry. Hence computing each matrix C^l in step 1b can be done in $O(n^3)$ time. Using the same argument, we can prove that each entry of $D^{l^{-1}} B^T C^l B D^{l^{-1}}$ can be computed in constant time using only 4 entries in C^l. All other estimates are standard.□

Claim 4.6. *Each update in step 5 can be done in $O(m^2)$ time.*

Proof. Since each update in step 5 is a simple rank one update, we can prove the claim using the Sherman-Morrison-Woodbury formula. □

Claim 4.7. *Assuming that $r \leq \sqrt{mk}$, the total number of updates done in step 5 in the r iterations between two applications of step 1 is $O(r^2)$.*

Proof is the same as in [6].

Theorem 4.2. *The average complexity of the procedure* Variant-2 *is $O(m^{2.5} + k^{0.5}mn^{1.5} + kn^2)$.*

Proof. The average complexity of step 1 is $O\left((kn^3 + m^3 + km^2)/r\right)$. The complexity of steps 3 and 4 is $O(m^2 + kn^2)$. The average complexity of step 5 is $O(rm^2)$ (using claims 4.6 and 4.7). Choosing r to balance these terms we get the result.□

Theorem 4.3. *Assuming usage of fast matrix multiplication procedure that inverts an $n \times n$ matrix in $O(n^{2.4})$ time the average complexity of the procedure* Variant-2 *is $O(m^{2.2}+k^{0.5}m^2+k^{0.5}mn^{1.2}+kn^2)$.*

5 Starting Point For The Algorithm

In this section, we provide a good starting point for our algorithm that enables us to prove a good complexity for our approximation algorithm. The idea behind the choice of the point is outlined below. The algorithm can be shown to converge rapidly if we start at a point close to the central path. As μ converges to infinity the central path converges to the center of the polytope (the minimizer of the barrier function). Since our constraints are separable it is very easy to calculate this center by splitting the capacity equally between the commodities. Near the center of the polytope, the value of μ changes rapidly on the central path and we can show a polynomial bound on μ, for which our starting point is a θ-approximation. We can set $w^0 = (x, y, z)$ as follows:

$$x_e^l = x_e^s = x_e = \frac{u(e)}{k+1} \qquad \text{denote} \quad \bar{x} = (x_e)_{e \in E}$$

$$y_e = -\frac{\mu}{x_e} + \left[B^T B\bar{x}\right]_e$$

$$z = c + Qx - A^T y$$

Theorem 5.1. *We can find μ such that w^0 is a θ approximation to the central path $\rho(\cdot)$ at μ with $\log \mu = O(\log(\theta^{-1} nUD))$.*

Proof. As mentioned above the values of x are chosen to minimize the barrier function (regardless of the objective function). The values of y are chosen such that

$$z_e^l = c_e^l + \left[B^T B\bar{x}\right]_e - y_e = c_e^l + \frac{\mu}{x_e} = c_e^l + \frac{\mu(k+1)}{u(e)} \qquad \forall l \in K \setminus \{s\}$$

$$z_e^s = -y_e = \frac{\mu}{x_e} - \left[B^T B\bar{x}\right]_e$$

The values of z are set by the feasibility of the dual problem.

The proof of the theorem follows from the next two claims.

Claim 5.1. *If $\mu \geq 2nUD$ then (x, y, z) is primal and dual feasible.*

Proof. The only non-obvious thing to check is $z \geq 0$. Now

$$z_e^l = c_e^l + \frac{\mu(k+1)}{u(e)} \geq -D + \mu\frac{k+1}{U} \geq 0$$

$$z_e^s = \frac{\mu}{x_e} - \left[B^T B\bar{x}\right]_e \geq \frac{\mu(k+1)}{u} - 2n\frac{U}{k+1} \geq 0$$

□

Claim 5.2.

$$\sum_{e\in E, l\in K} (z_e^l x_e^l - \mu)^2 \leq nkD^2U^2 + 4mU^4n^2$$

Proof.

$$\sum_{e\in E, l\in K\setminus\{s\}} (z_e^l x_e^l - \mu)^2 = \sum_{e\in E, l\in K\setminus\{s\}} (c_e^l x_e^l)^2 \leq nkD^2U^2$$

$$\sum_{e\in E}(z_e^s x_e^s - \mu)^2 = \sum_{e\in E}(\mu - x_e\left[B^T B\bar{x}\right]_e - \mu)^2 \leq m\left(\frac{U}{k+1}\cdot 2n\frac{U}{k+1}\right)^2 \leq 4mU^4n^2$$

□

The theorem now follows since we proved that for μ which is polynomial in θ^{-1}, n, d and U the chosen point is a feasible point and a θ approximation to the central path at μ. □

6 Stability

In this section, we will bound the precision needed in performing each arithmetic operation.

Claim 6.1. *A vector x is the global optimum solution of*

$$\min c^T x + \frac{1}{2}x^T Qx - \sum_i \mu_i \ln x_i$$

$$s.\ t.\ \ Ax = b \quad x \geq 0$$

if and only if there exist vectors z and y such that the following Karush-Kuhn-Tucker conditions are met:

$$\forall i \quad x_i z_i = \mu_i$$

$$Ax = b \quad x \geq 0$$

$$-Qx + A^T y + z - c = 0 \quad z \geq 0.$$

Proof. Using standard primal-dual arguments.

Claim 6.2. *Let $w = (x, y, z)$ be a θ approximation to the central path $\rho(\cdot)$ at μ then $\log z_e^l/x_e^l$ is $O\left(\log((1+\theta)\mu nU)\right)$ and $\log x_e^l/z_e^l$ is $O\left(\log((1+\theta)\mu^{-1}U)\right)$ for every edge e and commodity l.*

Proof. Set

$$\mu_e^l = x_e^l z_e^l \in [(1-\theta)\mu, (1+\theta)\mu]. \tag{6.2}$$

Using claim (6.1) we know that x is the minimizer of

$$E(x) - \sum_{e\in E\; l\in K} \mu_e^l \ln x_e^l \tag{6.3}$$

$$\text{s.t. } \forall e \in E \qquad \sum_{l\in K} x_e^l = u(e).$$

For a given $e = vw \in E$ and $l, j \in K$ define

$$p(t) \overset{def}{=} \left(Ex_v(x^l) - h^l(v) - t\right)^2 + \left(Ex_w(x^l) - h^l(w) + t\right)^2 + \left(Ex_v(x^j) - h^j(v) + t\right)^2 + \left(Ex_w(x^j)h^j(w) - t\right)^2 - \mu_e^l \ln(x_e^l + t) - \mu_e^j \ln(x_e^j - t)$$

From the fact that x minimizes the potential function defined in 6.3 we can deduce

$$0 = \frac{\partial}{\partial t} p(t)\bigg|_{t=0} = 2\left(-Ex_v(x^l) + Ex_w(x^l) + Ex_v(x^j) - Ex_w(x^j)\right) - \mu_e^l \frac{1}{x_e^l} + \mu_e^j \frac{1}{x_e^j}$$

Using the bounds in (6.2) we get

$$\frac{1}{x_e^l} \le \frac{1+\theta}{x_e^j} + \frac{2}{\mu(1+\theta)}\left(\left|Ex_v(x^l)\right| + \left|Ex_w(x^l)\right| + \left|Ex_v(x^j)\right| + \left|Ex_w(x^j)\right|\right) \le \frac{1+\theta}{x_e^j} + \frac{2}{\mu(1+\theta)}4nU$$

Since for every e there exists $j \in K$ such that $x_e^j \ge u(e)/(k+1) \ge 1/(k+1)$ we get that for every $e \in E$ and $l \in K$

$$\frac{1}{x_e^l} \le (1+\theta)(k+1) + \frac{2}{\mu(1+\theta)}4(nU + D).$$

The claim follows now from the facts $(1-\theta)\mu \le x_e^l z_e^l \le (1+\theta)\mu$ and $x_e^l \le U$. □

CLAIM 6.3. *In each iteration, the logarithms of the condition numbers of the matrices involved are* $O\left(\log(UDn\mu)\right)$.

Proof. Since the matrix $D^l + B^T B$ is symmetric and positive definite we can estimate

$$\kappa_2(D^l + B^T B) = \frac{\max_{\|x\|_2=1} x^T(D^l + B^T B)x}{\min_{\|x\|_2=1} x^T(D^l + B^T B)x} \le \frac{n + \max_e z_e^l/x_e^l}{\min_e z_e^l/x_e^l}$$

and the claim about this matrix follows from the previous claim. For the matrix $\left((D^s)^{-1} + \sum_{l\in K\setminus\{s\}} \left(D^l + B^T B\right)^{-1}\right)^{-1}$ note that it is the sum of symmetric positive definite matrices and hence the claim follows also for this matrix. All the other matrices that appear in the computations can be bounded in the same way.□

THEOREM 6.1. *To compute a solution with b bits of precision, the computations need to be done with* $O\left(b + \log(nUD)\right)$ *bits of precision.*

Proof. Using the theorem 3.2 we know that if only b bits of precision in the solution are needed we need to continue the algorithm until $\log \mu^{-1}$ becomes $O(b + \log(nUD))$. The theorem now follows from the previous claim. □

There are two conditions that should be satisfied at each iteration, namely, $Ax = u$ and $z = c + Qx - A^T y$. When working with infinite precision the conditions are maintain because of the definitions of $\Delta w = (\Delta x, \Delta y, \Delta z)$. When working with finite precision in order to avoid accumulation of rounding errors we should calculate the following values in the end of each iteration again:

$$x_e^s = u(e) - \sum_{l \in K \setminus \{s\}} x_e^l$$

$$z = c + Qx - A^T y$$

The variables resulting from these corrections can be shown to obey central path conditions since the only errors are those introduced by rounding.

7 Finding an Exact Solution

If we find an ϵ-approximate solution then for sufficiently small ϵ, we can round to an exact solution using the technique described in [17]. The conditions under which the technique is guaranteed to give an exact solution is stated in the following theorem, which is proved in [17].

THEOREM 7.1. *If there exists an optimum solution to a quadratic programming problem in n variables, in which every non zero variable is at least ϵ then an ϵ^2/n-approximate solution can be rounded to an exact optimum solution in $O(n)$ time.*

To use this result we will prove the following theorem:

THEOREM 7.2. *Given a multicommodity flow problem which has a solution, we can find $\epsilon > 0$ such that $\log \epsilon^{-1} = O(m \log m + \log UD)$ and such that there exists a solution in which every non zero flow is at least ϵ.*

Proof. Formulate the multicommodity flow problem as a linear programming feasibility problem, where a variable is associated with every path between a source and a sink of a commodity. We will denote by x_p the variable corresponding to the flow on the path p. The constraints are:

$$\forall p \quad x_p \geq 0$$

$$\forall i \leq k \qquad \sum_{p:p \text{ from } s_i \text{ to } t_i} x_p = d_i$$

$$\forall e \in E \quad \sum_{p:e \in p} x_p \leq u(e)$$

Using the theory of linear programming, we know that every linear optimization problem has a vertex solution which is defined by a solution of a linear system which is a sub-matrix of the constraints matrix. Using Cramer's rule we can give a lower bound on a non-zero entry in the solution by giving an upper bound on the determinant of a sub-matrix of the constraints matrix.

The constraints matrix is exponential in size, but it is easy to check that all the rows associated with the equalities of the type $x_p \geq 0$ do not contribute to the determinant. Hence every subdeterminant is bounded by the determinant of a $(m + k) \times (m + k)$ matrix with entries in $\{0, 1\}$. Hence the theorem follows. □

Note also that for a given flow solution, we can always use the zero vectors for the dual variables (this can be proved either directly from the equations, or intuitively, by noting that the optimal value on the polytope of the objective function is also the global optimal value).

THEOREM 7.3. *An exact solution to the multicommodity flow problem can be obtained by computing an ϵ-approximate solution where $\log \epsilon^{-1} = O(m \log m + \log U + \log D)$.*

8 Extensions

In this section, we show that our results extend to other variants like minimum cost multicommodity flow, concurrent flow and generalized multicommodity flow.

8.1 Minimum Cost Multicommodity Flow. The minimum cost multicommodity flow instance is a multicommodity flow instance with costs $p(e) \geq 0$ on the edges e. The optimum flow is the solution to the multicommodity flow problem which minimizes the cost function $P(x) = \sum_e p(e) \sum_l x_e^l$. Denote by $\bar{p} \in \mathbb{R}^E$ the vector of costs and let P be the maximum cost. In order to find an ϵ-approximate solution to the problem we will use our algorithm on the following objective function:

$$F_\epsilon(x) \overset{def}{=} \frac{PUm}{\epsilon} E(x) + P(x).$$

THEOREM 8.1. *The results that we obtained for the multicommodity flow problem (complexity, starting point, stability and exact solution) also hold for the minimum cost multicommodity flow problem with additional log of the ratio between biggest and smallest cost in the number of iterations.*

To prove the theorem we will state several claims that will show how we can use the results for the multicommodity case for the case of flow with costs.

CLAIM 8.1. *Assume that p^* is the minimum cost of a solution to the multicommodity flow problem and F_ϵ^* is the optimum value of F_ϵ. Also, assume that for a given pre-multiflow x, we have $F_\epsilon(x) \leq F_\epsilon^* + \epsilon$, then*

$$E(x) \leq 2\epsilon$$
$$P(x) \leq p^* + \epsilon.$$

Proof. Let x^* be the minimizer of F_ϵ and $\tilde{x}$ the minimum cost flow, then

$$F_\epsilon(x^*) \leq F_\epsilon(\tilde{x}) = P(\tilde{x}) = p^*.$$

Thus

$$P(x) \leq F_\epsilon(x) \leq F_\epsilon(x^*) + \epsilon \leq p^* + \epsilon.$$

Using the same inequalities

$$\frac{PUm}{\epsilon} E(x) \leq p^* + \epsilon.$$

hence

$$E(x) \leq \epsilon \left(\frac{p^*}{PUm} + \frac{\epsilon}{PUm} \right) \leq 2\epsilon.$$

□

CLAIM 8.2. *The complexity of computing a single iteration for the minimum cost multicommodity flow problem is the same as that for the feasibility problem.*

Proof. Since the only change from the original problem was changing the linear part of the objective function the algorithm stays exactly the same. The linear part does not affect the computation of a single iteration.□

Claim 8.3. *We can find a starting point $w = (x, y, z)$ and μ such that w is a θ approximation to the central path $\rho(\cdot)$ at μ with $\log \mu = O(\log(\theta^{-1} nUDP\epsilon^{-1}))$.*

Proof. Following the same idea as in section 5 we will choose the following starting point for a given μ :

$$x_e^l = x_e^s = x_e = \frac{u(e)}{k+1} \quad \text{denote} \quad \bar{x} = (x_e)_{e\in E}$$

$$y_e = -\frac{\mu}{x_e} + \frac{PUm}{\epsilon}\left[B^T B\bar{x}\right]_e$$

$$z = \frac{PUm}{\epsilon}c + p + \frac{PUm}{\epsilon}Qx - A^T y$$

Calculating the z's explicitly gives (note that slack variables are not multiplied by the prices):

$$z_e^l = \frac{PUm}{\epsilon}c_e^l + p(e) + \frac{PUm}{\epsilon}\left[B^T B\bar{x}\right]_e - y_e = \frac{PUm}{\epsilon}c_e^l + p(e) + \frac{\mu}{x_e} = \frac{PUm}{\epsilon}c_e^l + p(e) + \frac{\mu(k+1)}{u(e)} \quad \forall l \in K\backslash\{s\}$$

$$z_e^s = -y_e = \frac{\mu}{x_e} - \frac{PUm}{\epsilon}\left[B^T B\bar{x}\right]_e$$

In order to check the feasibility of the starting point we need to check if $z > 0$. The z's can be estimated as (compare to claim 5.1)

$$z_e^l = \frac{PUm}{\epsilon}c_e^l + p(e) + \frac{\mu(k+1)}{u(e)} \geq -\frac{DPUm}{\epsilon} + \frac{\mu(k+1)}{U} \quad \forall l \in K \setminus \{s\}$$

$$z_e^s = \frac{\mu}{x_e} - \frac{PUm}{\epsilon}\left[B^T B\bar{x}\right]_e \geq \frac{\mu(k+1)}{U} - \frac{PUm}{\epsilon}2n\frac{U}{k+1}.$$

Hence if $\mu \geq \max(\epsilon^{-1}DPU^2 m, U^3 Pm\epsilon^{-1}2n)$ then (x, y, z) is primal and dual feasible.

Estimating the distance of w from the central path at μ gives (compare to claim 5.2)

$$\sum_{e\in E, l\in K\backslash\{s\}} (z_e^l x_e^l - \mu)^2 = \sum_{e\in E, l\in K\backslash\{s\}} \left(x_e^l\left(\frac{PUm}{\epsilon}c_e^l + p(e)\right)\right)^2 \leq 2m^3U^3P^2D^2\epsilon^{-2}$$

$$\sum_{e\in E}(z_e^s x_e^s - \mu)^2 = \sum_{e\in E}(\mu - x_e\frac{PUm}{\epsilon}\left[B^T B\bar{x}\right]_e - \mu)^2 \leq m\left(\frac{U}{k+1}\cdot\frac{PUm}{\epsilon}2n\frac{U}{k+1}\right)^2 \leq 4m^3U^6n^2P^2\epsilon^{-2}$$

Hence we can find a feasible starting point which is a θ approximation to the central path at μ where μ is a polynomial in n, U, D, P and ϵ^{-1} . □

Claim 8.4. *The stability results proved for the multicommodity flow problem hold for the minimum cost multicommodity flow problem.*

Proof: Following the same proof as in section 6 proves the claim. An additional term of $\log P$ should be added to the number of bits to be used. □

8.2 The Concurrent Flow Problem. A concurrent flow problem involves finding the minimum factor λ such that the demands can be satisfied with capacities scaled to $\lambda u(e)$. In order to find an ϵ-approximate solution to the concurrent flow problem we will define for $\epsilon > 0$ the following objective function

$$G_\epsilon(x, t) \overset{def}{=} E(x) - \epsilon t$$

We will find an approximate solution to the problem by minimizing $G_\epsilon(x,t)$ subject to

$$\forall e \in E \quad \sum_{l \in K} x_e^l + tu(e) = u(e) \quad t \geq 0 \quad x_e^l \geq 0.$$

To ensure that a solution exists we will multiply each capacity by kD.

THEOREM 8.2. *The results that we obtained for the multicommodity flow problem (complexity, starting point, stability and exact solution) apply to the concurrent multicommodity flow problem as well.*

To prove the theorem we will state and prove several claims that will show how to use the same methods obtained for the multicommodity flow problem to the concurrent flow problem.

CLAIM 8.5. *Assume that $\epsilon \leq 1$. Suppose that $\tilde{t}$ is the optimal value of the concurrent flow problem and that G_ϵ^* is the optimal value of G_ϵ subject to the constraints. Assume that for a given pre-flow x and parameter t we have $G_\epsilon(x,t) \leq G_\epsilon^* + \epsilon^2$, then*

$$t \geq \tilde{t} - \epsilon$$

$$E(x) \leq 2\epsilon$$

Proof. Let $(\tilde{x}, \tilde{t})$ be the solution for the concurrent flow problem. Then

$$G_\epsilon^* \leq G_\epsilon(\tilde{x}, \tilde{t}) = -\epsilon\tilde{t}$$

hence

$$-\epsilon t \leq G_\epsilon(x,t) \leq G_\epsilon^* + \epsilon^2 \leq -\epsilon\tilde{t} + \epsilon^2$$

which implies the first inequality. In addition to that

$$E(x) = G_\epsilon(x,t) + \epsilon t \leq G_\epsilon^* + \epsilon^2 + \epsilon t$$

Note that always $G_\epsilon^* \leq 0$, $t \leq 1$, $\epsilon \leq 1$, hence

$$E(x) \leq 2\epsilon$$

□

remark: Note that since we scaled the capacities by kD to ensure the existence of a solution, an ϵ-approximation to our problem is an ϵkD approximation to the original concurrent flow problem.

CLAIM 8.6. *The asymptotic complexity of one iteration in optimizing G_ϵ is the same as in the multicommodity flow problem.*

Proof. The algorithms described in section 4 are can be seen as solving the following linear system:

$$\begin{pmatrix} Z & 0 & X \\ A & 0 & 0 \\ -Q & A^T & I \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \\ \Delta z \end{pmatrix} = \begin{pmatrix} XZe - \hat{\mu}e \\ 0 \\ 0 \end{pmatrix}$$

In the concurrent-flow problem we have two additional variables t and z_t and the matrix to solve is the following:

$$\begin{pmatrix} Z & 0 & X & 0 & 0 \\ 0 & 0 & 0 & z_t & t \\ A & 0 & 0 & \bar{u} & 0 \\ -Q & A^T & I & 0 & 0 \\ 0 & u^T & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \\ \Delta z \\ \Delta t \\ \Delta z_t \end{pmatrix} = \begin{pmatrix} XZe - \hat{\mu}e \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Hence the linear system for the concurrent flow case is the same as the multicommodity flow case, except a constant number of rank one updates. Using the Sherman-Morrison formula, it then follows that the complexity of solving the linear system is a constant multiple of that for the multicommodity flow case. □

CLAIM 8.7. *We can find a starting point $w = (x, y, z, t)$ and μ such that w is a θ approximation to the central path $\rho(\cdot)$ at μ with $\log \mu = O(\log(\theta^{-1} nUD\epsilon^{-1}))$.*

Proof. As in section 5 we will use as a starting point the point that minimizes the barrier function regardless of the objective function. By differentiating the barrier function we get that the starting primal-dual point (x, y, z, t) for a given μ is:

$$t = \frac{1}{m(k+1)+1}$$

$$x_e^l = x_e^s = x_e = (1-t)\frac{u(e)}{k+1} \quad \text{denote} \quad \bar{x} = (x_e)_{e \in E}$$

$$y_e = -\frac{\mu}{x_e} + \left[B^T B \bar{x}\right]_e$$

$$z_e^l = c_e^l + \left[B^T B \bar{x}\right]_e - y_e = c_e^l + \frac{\mu}{x_e} = c_e^l + \frac{\mu(k+1)}{u(e)(1-t)} \quad \forall l \in K \setminus \{s\}$$

$$z_e^s = -y_e = \frac{\mu}{x_e} - \left[B^T B \bar{x}\right]_e$$

$$z_t = -\epsilon - \sum_{e \in E} u(e) y_e = -\epsilon + \mu \sum_{e \in E} \frac{u(e)}{x_e} - \left[B^T B \bar{x}\right]_e = -\epsilon + \mu m \frac{k+1}{1-t} - \left[B^T B \bar{x}\right]_e =$$

$$-\epsilon + \mu(m(k+1)+1) - \left[B^T B \bar{x}\right]_e$$

To ensure that the point (x, y, z, t) is primal-dual feasible we have to check that $z > 0$. Estimating z can be done by (compare to claim 5.1):

$$z_e^l = c_e^l + \frac{\mu(k+1)}{u(e)(1-t)} \geq -D + \frac{\mu(k+1)}{U}$$

$$z_e^s = \frac{\mu}{x_e} - \left[B^T B \bar{x}\right]_e \geq \frac{\mu(k+1)}{U} - 2n\frac{U}{k+1}$$

$$z_t = -\epsilon + \mu(m(k+1)+1) - \left[B^T B \bar{x}\right]_e \geq -1 + \mu(m(k+1)+1) - 2n\frac{U}{k+1}$$

Hence if $\mu \geq \max(UD, 2U^2 n)$ then (x, y, z, t) is primal and dual feasible.

To bound the distance from the central path we can calculate (compare to claim 5.2):

$$\sum_{e \in E, l \in K \setminus \{s\}} (z_e^l x_e^l - \mu)^2 = \sum_{e \in E, l \in K \setminus \{s\}} (c_e^l x_e^l)^2 \leq nkD^2U^2$$

$$\sum_{e \in E} (z_e^s x_e^s - \mu)^2 = \sum_{e \in E} (\mu - x_e \left[B^T B \bar{x}\right]_e - \mu)^2 \leq m\left(\frac{U}{k+1} \cdot 2n\frac{U}{k+1}\right)^2 \leq 4mU^4n^2$$

$$(tz_t - \mu)^2 = (-t\epsilon + t\mu(m(k+1)+1) - t\left[B^T B \bar{x}\right]_e - \mu)^2 = (-t\epsilon - t\left[B^T B \bar{x}\right]_e)^2 \leq \left(\epsilon + 2n\frac{U}{k+1}\right)^2 \leq 5n^2U^2$$

Hence for μ polynomial in n, D, U and θ we know that (x, y, z, t) is a θ approximation to the central path at μ. □

Claim 8.8. *We can find an approximate solution to the concurrent flow problem with the same asymptotic complexity as in the multicommodity flow problem.*

Proof. Given $\epsilon > 0$, in order to find an ϵ approximation to the concurrent flow problem we will find the approximate optimal of $G_\epsilon(\cdot)$. Claim 8.5 proves that this will give an approximate solution to the concurrent flow problem. Combining claim 8.7, theorem 3.1 and theorem 3.2 (adjusted to G_ϵ instead of the original quadratic objective function) assures us that an ϵ approximate solution will be calculated in $O(\sqrt{mk}\log(nDU\epsilon^{-1}))$ iterations. Claim 8.6 assures us that the complexity of each iteration is the same. □

Claim 8.9. *We can find an exact solution to the concurrent flow problem with the same asymptotic complexity as the multicommodity flow problem.*

Proof. Following the same technique as in the multicommodity case, we will show that the number of bits needed to compute an exact solution is $O(m\log m + \log UD)$ which is sufficient to prove the claim.

To bound the number of bits needed to describe an exact solution we will follow the same proof as in section 7, adjusted to the concurrent flow problem. We will associate a variable with every path from a source and a sink of a commodity. In addition slack variables $s(e)$, and a variable t will be defined. The linear constraints defined for each edge will be:

$$\forall e \in E, \quad \sum_{p:e\in p} x_p + s(e) + tu(e) \leq u(e)$$

Estimating the solution in the same way, we have to bound the determinant of an $m+k \times m+k$ matrix in which all entries are 0 or 1, except possibly the column containing the capacities. Hence the claim follows.

8.3 The Generalized Flow Problem.

For the definition of the generalized flow problem see [2]. In this problem, flows moving on edges have different gains or losses. To solve the generalized multicommodity flow problem using our algorithm, all we have to change are the entries in B which instead of being 0 and 1, will now depend on the gain/loss factor on the edges. Let γ be the ratio between the biggest and smallest generalized flow coefficients. The next two theorems can be easily verified:

Theorem 8.3. *Each iteration of our algorithm, as applied to the generalized multicommodity flow problem has the same complexity as for the original multicommodity flow problem. However, for finding the exact solution there is an additional $O(\log\gamma)$ factor in the number of iterations and for finding an approximate solution there is an additional factor of $n\log\gamma$.*

Proof. Since complexity proofs used only the sparseness of B and not the fact that all entries are either 0 or 1, the complexity of a single iteration remains the same in the generalized flow problem. In the estimation of the starting point, μ will depend polynomially on γ, adding a $\log\gamma$ factor to the running time. Also, in converting a pre-flow to a flow, we can gain or lose as much as γ^n factor. Hence, to satisfy $1-\epsilon$ fraction of all the demands , we may need more accurate solutions, giving an additional term of $n\log\gamma$ in the number of iterations. □

Theorem 8.4. *We can solve the single commodity generalized flow problem without using fast matrix multiplication in $O\left(n^2m^{1.5}\log(nUD\gamma)\right)$.*

Proof. We will use the conjugate gradient variant to compute each iteration and hence the complexity of each iteration is $O(mn)$. To estimate the convergence criterion for rounding to the exact solution, we have to bound the largest sub-determinant of the matrix,

$$\begin{pmatrix} Q & A \\ A & 0 \end{pmatrix}.$$

In our case, the matrix is

$$\begin{pmatrix} B^TB & 0 & I \\ 0 & 0 & I \\ I & I & 0 \end{pmatrix}.$$

Since the log of the biggest subdeterminant of B^TB is $O(n \log n\gamma)$ and the number of variables is $2m$ the overall complexity of our algorithm is $O(mn \cdot m^{0.5} \cdot (n \log n\gamma + \log DU))$ and the theorem follows.$\square$

Acknowledgements

We thank Serge Plotkin for helpful discussions.

References

[1] B. Awerbuch and T. Leighton. A simple local-control approximation algorithm for multicommodity flow. In *Proc. 26th Annual ACM Symposium on Theory of Computing*, 1994. To appear.

[2] A. V. Goldberg, S. A. Plotkin, and É. Tardos. Combinatorial Algorithms for the Generalized Circulation Problem. In *Proc. 29th IEEE Annual Symposium on Foundations of Computer Science*, pages 174–185, 1988.

[3] G. H. Golub and C. F. Van Loan. *Matrix Computations.* hopkins, second edition, 1989.

[4] T. C. Hu. Multi-Commodity Network Flows. *J. ORSA*, 11:344–360, 1963.

[5] A. Kamath, O. Palmon, and S. Plotkin. Fast Approximation Algorithm for Minimum Cost Multicommodity Flow. In *Proc. 6th ACM-SIAM Symposium on Discrete Algorithms*, 1995.

[6] S. Kapoor and P. M. Vaidya. Fast Algorithms for Convex Quadratic Programming and Multicommodity Flows. In *Proc. 18th Annual ACM Symposium on Theory of Computing*, pages 147–159, 1986.

[7] N. Karmarkar. A New Polynomial-Time Algorithm for Linear Programming. *Combinatorica*, 4:373–395, 1984.

[8] T. Leighton, F. Makedon, S. Plotkin, C. Stein, É. Tardos, and S. Tragoudas. Fast approximation algorithms for multicommodity flow problems. In *Proc. 23st Annual ACM Symposium on Theory of Computing*, 1991.

[9] T. Leighton and S. Rao. An approximate max-flow min-cut theorem for uniform multicommodity flow problems with applications to approximation algorithms. In *Proc. 29th IEEE Annual Symposium on Foundations of Computer Science*, pages 422–431, 1988.

[10] R. D.C. Monteiro and I. Adler. Interior Path Following Primal-Dual Algorithms. Part II: Convex Quadratic Programming. *Math. Prog.*, 44:43–66, 1989.

[11] S. M. Murray. *An Interior Point Approach to the Generalized Flow Problem with Costs and Related Problems.* PhD thesis, Department of Operations Research, Stanford University, Stanford, CA., 1992.

[12] S. A. Plotkin, D. Shmoys, and É. Tardos. Fast Approximation Algorithms for Fractional Packing and Covering. In *Proc. 32nd IEEE Annual Symposium on Foundations of Computer Science*, 1991.

[13] T. Radzik. Fast Deterministic Approximation for the Multicommodity Flow Problem. In *Proc. 6th ACM-SIAM Symposium on Discrete Algorithms*, 1995.

[14] F. Shahrokhi and D. W. Matula. The maximum concurrent flow problem. Technical Report CSR-183, Department of Computer Science, New Mexico Tech., 1988.

[15] C. Stein. *Approximation algorithms for multicommodity flow and scheduling problems.* PhD thesis, MIT, 1992.

[16] P. M. Vaidya. Speeding up Linear Programming Using Fast Matrix Multiplication. In *Proc. 30th IEEE Annual Symposium on Foundations of Computer Science*, 1989.

[17] Y. Ye. Toward Probabilistic Analysis of Interior-Point Algorithms for Linear Programming. *Math. of Oper. Res.*, pages 38–52, 1994.