NEWTON'S METHOD FOR FRACTIONAL COMBINATORIAL OPTIMIZATION
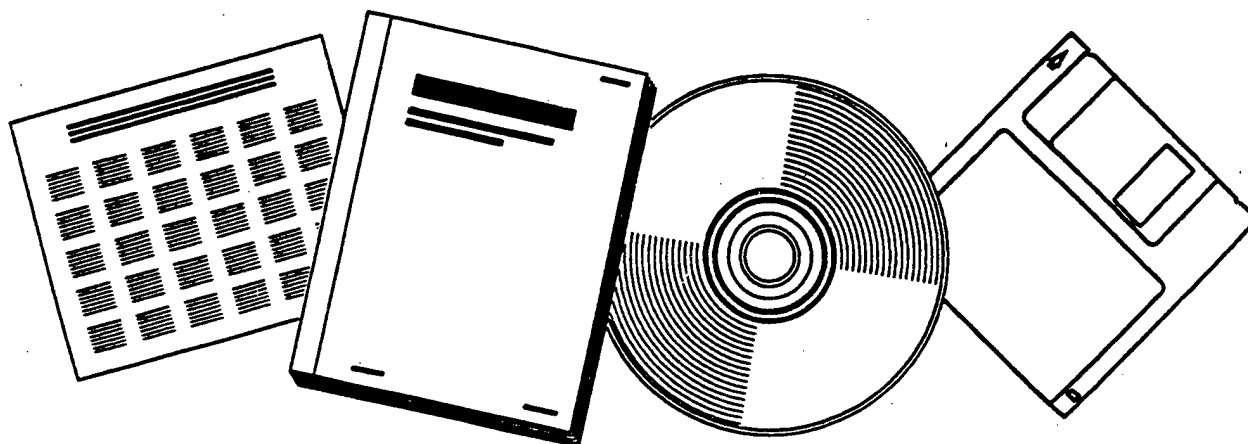
STANFORD UNIV., CA

DTIC QUALITY INSPECTED 2

JAN 92

19970409 020

REPORT DOCUMENTATION PAGE

1. AGENCY USE ONLY	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED

4. TITLE AND SUBTITLE

Newton's Method for Fractional Combinatorial Optimization

5. FUNDING NUMBERS

PB96-152210

6. AUTHOR(S)

Tomasz Radzik

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Computer Science Department
Stanford University
Stanford, CA 94305

8. PERFORMING ORGANIZATION REPORT NUMBER

STAN-CS-92-1406

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

ONR

Arlington, VA 22217

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION / AVAILABILITY STATEMENT

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

We consider Newton's method for the linear fractional combinatorial optimization. First we show a strongly polynomial bound on the number of iterations for the general case. Then we consider the transshipment problem when the maximum arc cost is being minimized. This problem can be reduced to the maximum mean-weight cut problem, which is a special case of the linear fractional combinatorial optimization. We prove that Newton's method runs in $O(m)$ iterations for the maximum mean-weight cut problem. One iteration is dominated by the maximum flow computation, so the overall running time is $\tilde{O}(m^2 n)$. The previous fastest algorithm is based on Meggido's parametric search method and runs in $\tilde{O}(n^3 m)$ time.

14. SUBJECT TERMS

15. NUMBER OF PAGES

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT

BIBLIOGRAPHIC INFORMATION

Abstract: We consider Newton's method for the linear fractional combinatorial optimization. First we show a strongly polynomial bound on the number of iterations for the general case. Then we consider the transshipment problem when the maximum arc cost is being minimized. This problem can be reduced to the maximum mean-weight cut problem, which is a special case of the linear fractional combinatorial optimization. We prove that Newton's method runs in $O(m)$ iterations for the maximum mean-weight cut problem. One iteration is dominated by the maximum flow computation, so the overall running time is $O(m^2 n)$. The previous fastest algorithm is based on Meggido's parametric search method and runs in optimum $O(n^3 m)$ time.

Newton's Method for Fractional Combinatorial Optimization

by

Tomasz Radzik

Department of Computer Science

Stanford University
Stanford, California 94305

Newton's Method for Fractional Combinatorial Optimization

Tomasz Radzik [*]

Abstract

We consider Newton's method for the linear fractional combinatorial optimization. First we show a strongly polynomial bound on the number of iterations for the general case. Then we consider the transshipment problem when the maximum arc cost is being minimized. This problem can be reduced to the maximum mean-weight cut problem, which is a special case of the linear fractional combinatorial optimization. We prove that Newton's method runs in $O(m)$ iterations for the maximum mean-weight cut problem. One iteration is dominated by the maximum flow computation, so the overall running time is $\bar{O}(m^2 n)$. The previous fastest algorithm is based on Meggido's parametric search method and runs in $\bar{O}(n^3 m)$ time.

1 Introduction

A *linear fractional combinatorial optimization* (LFCO) problem is defined as follows. A specification of a system $\mathcal{X} \subseteq \{0,1\}^p$ and two vectors $(a_1, a_2, \ldots, a_p)$, $(b_1, b_2, \ldots, b_p) \in \mathbb{R}^p$ are given.

$$\text{Maximize} \quad \frac{a_1 x_1 + a_2 x_2 + \cdots + a_p x_p}{b_1 x_1 + b_2 x_2 + \cdots + b_p x_p}, \quad \text{subject to} \quad (x_1, x_2, \ldots, x_p) \in \mathcal{X}.$$

$\mathcal{X}$ belongs to some specified class of systems. Vector $(x_1, x_2, \ldots, x_p) \in \mathcal{X}$, represents some structure, a subset of some underlying set of p objects. a_i and b_i are *the cost* and *the weight of object i*. $\sum_{i=1}^p a_i x_i$, $\sum_{i=1}^p b_i x_i$, and $(\sum_{i=1}^p a_i x_i)/(\sum_{i=1}^p b_i x_i)$ are *the cost, weight, and mean-weight cost of the structure* represented by $(x_1, x_2, \ldots, x_p)$. In an LFCO problem we are asked to compute the maximum mean-weight cost of a structure in $\mathcal{X}$. We also want to find a structure which has the maximum mean-weight cost.

For example the maximum mean-weight spanning tree problem is an LFCO problem. In this problem the class of systems is the class of sets of all spanning trees in graphs:

$$\{\{T \mid T - \text{spanning tree in } G\} \mid G - \text{graph}\}.$$

The description of graph $G = (V, E)$ is the specification of $\mathcal{X}_G \subseteq \{0,1\}^m$, the set of (the characteristic vectors of) all spanning trees in G ($m = |E|$). The underlying set of objects for $\mathcal{X}_G$ is the set of edges in G. Vector $(x_1, x_2, \ldots, x_m) \in \mathcal{X}_G$ represents spanning tree T in G such that edge i belongs to T iff $x_i = 1$. Numbers $a_1, a_2, \ldots, a_m$ are the costs of edges and $b_1, b_2, \ldots, b_m$ are the weights. The task is to find a spanning tree in G with the maximum mean-weight cost.

We will use bold letters to denote vectors and corresponding italic letters to denote their coordinates, for example $\mathbf{x} = (x_1, x_2, \ldots, x_p)$. For two vectors $\mathbf{a}$ and $\mathbf{x}$ of equal length, $\mathbf{ax}$ denotes the scalar product of $\mathbf{a}$ and $\mathbf{x}$, i.e., $\mathbf{ax} = a_1 x_1 + a_2 x_2 + \cdots + a_p x_p$. With this notation an LFCO problem is to

$$\text{(P1)} \qquad \text{maximize} \quad \frac{\mathbf{ax}}{\mathbf{bx}}, \quad \text{subject to} \quad \mathbf{x} \in \mathcal{X}.$$

We assume that $\mathbf{bx} > 0$ for all $\mathbf{x} \in \mathcal{X}$, and $\mathbf{ax} > 0$ for some $\mathbf{x} \in \mathcal{X}$. Problem (P1) can be equivalently formulated in the following way:

$$\text{(P2)} \quad \text{minimize } \delta, \quad \text{subject to} \quad (\mathbf{ax}) - \delta(\mathbf{bx}) \leq 0, \quad \text{for all } \mathbf{x} \in \mathcal{X}.$$

Let δ^* denote the minimum we are looking for. Let

$$h(\delta) = \max_{\mathbf{x} \in \mathcal{X}} \{(\mathbf{ax}) - \delta(\mathbf{bx})\}. \tag{1}$$

1

Function $h(\delta)$ is convex, piecewise linear, decreasing, and δ^* is its only root. We assume that we can maximize any linear function over $\mathcal{X}$, so for any δ we can compute $h(\delta)$ and $\mathbf{x} \in \mathcal{X}$ such that $(\mathbf{a} - \delta\mathbf{b})\mathbf{x} = h(\delta)$. Let $\mathcal{A}$ be an algorithm which does such computation and let T be its running time. Having such an algorithm enables us to compute δ^* with Newton's method. We show that Newton's method finds δ^* in $\bar{O}(p^4)$ iterations. This bound on the number of iterations is independent of the complexity of the class of systems which $\mathcal{X}$ belongs to, and independent of the complexity of computing $h(\delta)$. One iteration is dominated by algorithm $\mathcal{A}$, so the overall time is $\bar{O}(p^4 T)$.

The other way of computing δ^* is Megiddo's parametric search method [8]. This approach gives an algorithm with running time $O(T'_Q(T' \log Q + Q))$, where T' and T'_Q refer to the sequential running time and the parallel running time on Q processors of computing the sign of $h(\delta)$ for a given δ. Thus an efficient algorithm for an LFCO problem can be obtained if an efficient parallel algorithm for the corresponding nonfractional problem is available.

If costs and weights are integers from $[-U, U]$, then using the straightforward binary search we can find in $O(\log(pU))$ iterations an interval I of length less than $1/(pU)^2$ which contains δ^*. One iteration is dominated by computing the sign of $h(\delta)$ for an appropriate δ. An interval of length less than $1/(pU)^2$ can contain at most one mean-weight cost. The overall time of this approach is $O(T' \log(pU) + T)$.

In the second part of the paper we consider the *minimum maximum arc cost transshipment* (MAC) problem, which is the transshipment problem when the maximum of the arc costs is being minimized. This problem can be reduced to the *maximum mean-weight cut* (MWC) problem, which is an LFCO problem. The underlying nonfractional problem is the standard maximum flow problem. This problem does not have an efficient parallel algorithm. Nevertheless, up to now, Megiddo's method gave the best known upper bound, namely $\bar{O}(mn^3)$ [8], for the MWC and MAC problems. We show that Newton's method applied to the MWC problem runs in $O(m)$ iterations. One iteration is dominated by maximum flow computation, so the overall running time is $\bar{O}(m^2 n)$. The same bound holds for the MAC problem.

2 Newton's Method for Linear Fractional Combinatorial Optimization

The following is Newton's method for computing the root δ^* of $h(\delta)$ (defined in (1)).
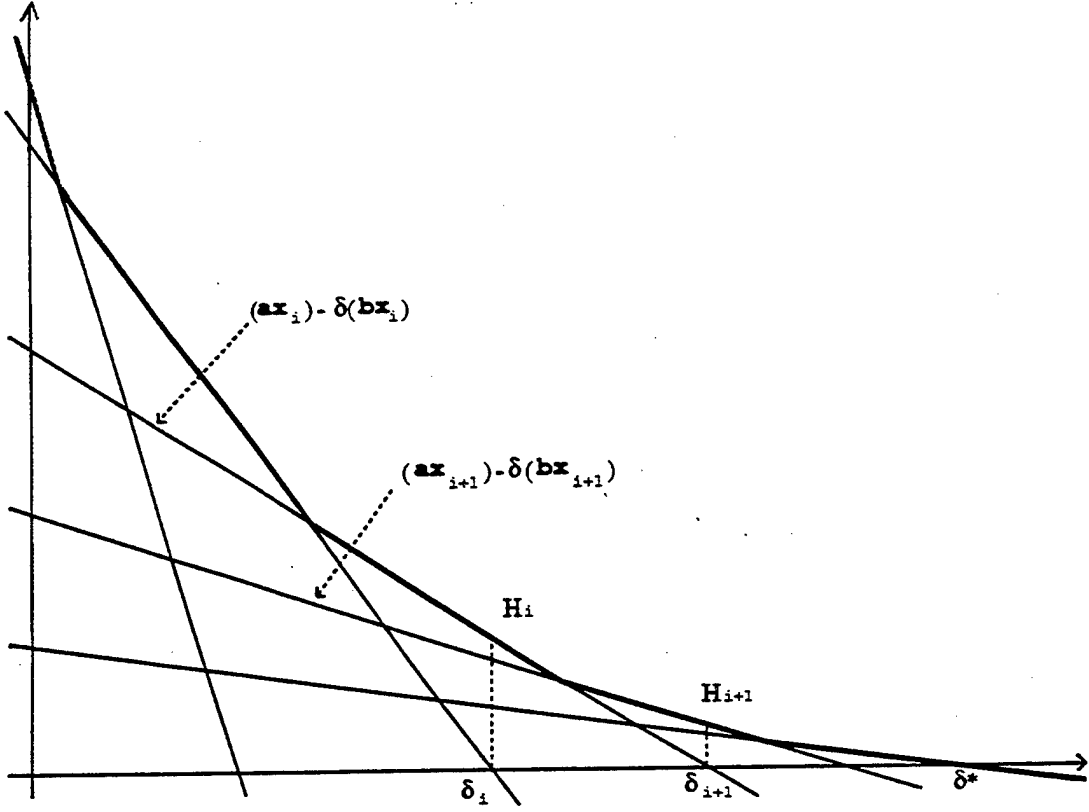
Figure 1: Newton's method for solving $h(\delta) = 0$

Let $\bar{\delta} \leq \delta^*$ be the current estimation to δ^*. Initially $\bar{\delta} = 0$. (Observe that our assumptions guarantee $\delta^* > 0$.) During one iteration we compute $h(\bar{\delta})$ and $\bar{x} \in \mathcal{X}$ such that $h(\bar{\delta}) = (a\bar{x}) - \bar{\delta}(b\bar{x})$, *i.e.*, we maximize linear function $(a - \bar{\delta}b)x$ over $\mathcal{X}$. This is done by algorithm $\mathcal{A}$. If $h(\bar{\delta}) = 0$, then $\delta^* = \bar{\delta}$ and the algorithm terminates. Otherwise we compute the next estimation $\bar{\delta} \leftarrow a\bar{x}/b\bar{x}$, the mean-weight cost of $\bar{x}$, and go to the next iteration. The process is illustrated in Figure 1.

Let δ_i be the value of $\bar{\delta}$ at the beginning of ith iteration, and x_i, H_i, and B_i, be $\bar{x}$, $(a - \delta_i b)\bar{x}$, and $b\bar{x}$ from this iteration.

$$H_i = (a - \delta_i b)x_i = \max\{(a - \delta_i b)x \mid x \in \mathcal{X}\},$$
$$B_i = bx_i,$$
$$\delta_{i+1} = \frac{ax_i}{bx_i},$$

and it can be easily derived that

$$\delta_{i+1} - \delta_i = \frac{H_i}{B_i}. \tag{2}$$

3

The following lemma indicates fast convergence of the above algorithm.

Lemma 1

$$\frac{H_{i+1}}{H_i} + \frac{B_{i+1}}{B_i} \leq 1. \tag{3}$$

Proof. Vector x_i maximizes $(a - \delta_i b)x$, so

$$(a - \delta_i b)x_i \geq (a - \delta_i b)x_{i+1}.$$

Therefore

$$
\begin{aligned}
H_i &= (a - \delta_i b)x_i \geq (a - \delta_i b)x_{i+1} \\
&= (a - \delta_{i+1} b)x_{i+1} + (\delta_{i+1} - \delta_i)bx_{i+1} \\
&= H_{i+1} + \frac{H_i}{B_i} B_{i+1}.
\end{aligned}
$$

This implies Inequality (3). ∎

The above lemma gives immediately the following bound on the number of iterations.

Theorem 1 If in an LFCO problem the coordinates of vectors a and b are integers from $[-U, U]$, then Newton's method finds the optimum in $O(\log(pU))$ iterations.

Proof. Inequality (3) implies

$$\frac{H_{i+1}B_{i+1}}{H_i B_i} \leq \frac{1}{4}. \tag{4}$$

For any i for which $H_i > 0$,

$$(pU)^2 \geq H_i B_i \geq \frac{1}{pU}. \tag{5}$$

The second inequality holds because

$$H_i = \frac{a x_i b x_{i-1} - a x_{i-1} b x_i}{b x_{i-1}} \geq \frac{1}{pU}.$$

Inequalities (4) and (5) imply the bound of $O(\log(pU))$ on the number of iteration. ∎

The special case with uniform weights, *i.e.*, $b = (1, 1, \ldots, 1)$, is of independent interest. The task here is to find the *maximum mean cost of a structure*. Sequence (B_i), excluding the last iteration, is decreasing. This implies the following strongly polynomial bound, which was observed by Karzanov [11] and, in the context of the maximum mean cut problem, by McCormick and Ervolina [7].

Theorem 2 If the weights are uniform, then Newton's method runs in at most $p+1$ iterations.

4

Now we prove a strongly polynomial bound on the number of iterations for the general case, when both costs and weights can be arbitrary real numbers. The intuition behind the analysis is as follows. Lemma 1 suggests that there are some sequences related to the rate of convergence of the method, which tend to zero at least geometrically fast. The elements of these sequences are obtained from only $2p$ numbers, the costs $a_1, a_2, \ldots, a_p$, and the weights $b_1, b_2 \ldots, b_p$, using only $O(p)$ additions/subtractions and at most one or two multiplications/divisions. We show that because of this limiting use of arithmetical operations, the lengths of such sequences cannot be too big.

To expand this intuition a little bit, let us assume that $B_{i+1}/B_i \leq 1 - \alpha$, for some positive constant α, that is sequence (B_i) tends to zero at least geometrically fast. Each element of sequence (B_i) is a sum of different elements from $\{b_1, b_2, \ldots, b_p\}$. Let us further assume that $b_1 \geq b_2 \geq \cdots \geq b_p \geq 0$. Obviously $B_1 \leq pb_1$. Since (B_i) decreases at least geometrically, for some $l = O(\log p)$, $B_l < b_1$. It means that b_1 is not a term in B_l, nor is it in B_i, for any $i \geq l$. $B_l \leq (p-1)b_2$, and the next $O(\log p)$ elements of (B_i) exclude b_2, then b_3, and so on. Therefore the length of sequence (B_i) is $O(p \log p)$.

There are two reasons why the general case is more complicated. First, we have to deal with positive and negative numbers. Even if both costs and weights are positive, negative numbers appear because subtractions are used in forming the elements of (H_i). Second, if (B_i) does not decrease fast enough, then we have to use sequence (H_i), whose elements are not just subsums of some given set of numbers. The following lemma is our tool to deal with both positive and negative numbers. Here the coordinates of vector $\mathbf{c}$ are numbers which are used to form sums, and $(\mathbf{y}_k \mathbf{c})$ is a sequence of such sums.

Lemma 2 Let $\mathbf{c} \in \mathbf{R}^p$ and $\mathbf{y}_k \in \{0,1\}^p \backslash \{\mathbf{0}\}$, for $k = 1, 2, \ldots, q$, be such that

$$|\mathbf{y}_{k+1} \mathbf{c}| < \frac{1}{p^{3p}} |\mathbf{y}_k \mathbf{c}|, \quad \text{for } k = 1, 2, \ldots, q-1. \tag{6}$$

Then $q \leq p$.

Proof. Assume that $q > p$. Let Y denote the matrix whose rows are vectors $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_q$. Since $q > p$, there is a nonzero vector $\mathbf{z} \in \mathbf{R}^q$ such that $\mathbf{z}Y = \mathbf{0}$. Moreover, $\mathbf{z}$ can be chosen in such a way that all its coordinates are rational numbers with denominators and enumerators being the determinants of submatrices of Y. Let $\mathbf{z} = (z_1, z_2, \ldots, z_q)$ be such a vector. Thus, for $i = 1, 2, \ldots, q$, either $z_i = 0$ or

$$\frac{1}{p^p} < |z_i| < p^p. \tag{7}$$

We have
$$0 = zYc = z_1(y_1c) + z_2(y_2c) + \cdots + z_q(y_qc).$$

But this is not possible, because the nonzero terms on the right side cannot cancel. Since $y_q \neq 0$, so for some $1 \leq i \leq q-1$, $z_i \neq 0$, and $z_iy_ic \neq 0$. If $1 \leq i < j \leq q$ and $z_iy_ic \neq 0$, then (6) and (7) imply

$$|z_j(y_jc)| < p^{2p}|z_i|\frac{1}{p^{3p}}|y_ic| = \frac{1}{p^p}|z_i(y_ic)|.$$

$\blacksquare$

Lemma 3 There are at most $O(p^2\log p)$ iterations k such that $B_{k+1} \leq \frac{2}{3}B_k$.

Proof. Consider the sequence of all iterations k such that $B_{k+1} \leq \frac{2}{3}B_k$. Take from this sequence every lth iteration, where $l = \lceil 6p\log p \rceil$. Let they be iterations $i_1, i_2, \ldots, i_q$. For $1 \leq k \leq q-1$,

$$B_{i_{k+1}} \leq \left(\frac{2}{3}\right)^l B_{i_k} \leq \frac{1}{p^{3p}}B_{i_k}.$$

Lemma 2 implies $q \leq p$, which implies $O(p^2\log p)$ bound on the number of iterations k such that $B_{k+1} \leq \frac{2}{3}B_k$. $\blacksquare$

Lemma 4 There are at most $O(p^2\log p)$ consecutive iterations k such that $B_{k+1} \geq \frac{2}{3}B_k$.

Proof. Consider a sequence of consecutive iterations i, $i+1$, ..., j, such that for each $i \leq k \leq j-1$:

$$B_{k+1} \geq \frac{2}{3}B_k. \tag{8}$$

This and Inequality (3) imply that for each $i \leq k \leq j-1$:

$$H_{k+1} \leq \frac{1}{3}H_k.$$

Therefore

$$\delta_{k+2} - \delta_{k+1} = \frac{H_{k+1}}{B_{k+1}} \leq \frac{1}{2}\frac{H_k}{B_k} = \frac{1}{2}(\delta_{k+1} - \delta_k). \tag{9}$$

Let $l = \lceil 3p\log p \rceil + 1$. Consider the sequence of every lth iteration out of iterations i, $i+1$, ..., j. Let they be iterations $i_1, i_2, \ldots, i_q$. We show that Lemma 2 can be applied to the sequence $(x_{i_k-1}(-a + \delta_{i_q}b))$, where $1 \leq k \leq q$. This is a sequence of subsums of $\sum_{r=1}^p(-a_r + \delta_{i_q}b_r)$. First we estimate the rate of convergence of sequence (δ_{i_k}) to δ_{i_q}. For $1 \leq k \leq q-2$,

$$\delta_{i_q} - \delta_{i_{k+1}} \leq (\delta_{i_q} - \delta_{i_q-1}) + (\delta_{i_q-1} - \delta_{i_q-2}) + \cdots + (\delta_{i_{k+1}+1} - \delta_{i_{k+1}})$$

6

$$\leq (\delta_{i_{k+1}+1} - \delta_{i_{k+1}})(1 + \frac{1}{2} + \frac{1}{4} + \cdots + \frac{1}{2^{(i_q - i_{k+1} - 1)}})$$

$$< 2(\delta_{i_{k+1}+1} - \delta_{i_{k+1}})$$

$$\leq 2\left(\frac{1}{2}\right)^{(i_{k+1} - i_k)}(\delta_{i_k+1} - \delta_{i_k})$$

$$\leq 2\left(\frac{1}{2}\right)^l (\delta_{i_q} - \delta_{i_k})$$

$$\leq \frac{1}{p^{3p}}(\delta_{i_q} - \delta_{i_k}).$$

The inequalities in the second and fourth lines follow from Inequality (9).

Recall that $\delta_{i+1} = (\mathbf{a}\mathbf{x}_i)/(\mathbf{b}\mathbf{x}_i)$, so for $k = 1, 2, \ldots, q$,

$$\mathbf{x}_{i_k - 1}(-\mathbf{a} + \delta_{i_q}\mathbf{b}) = (\delta_{i_q} - \delta_{i_k})\mathbf{b}\mathbf{x}_{i_k - 1}.$$

Put

$$\mathbf{c} = -\mathbf{a} + \delta_{i_q}\mathbf{b},$$

$$\mathbf{y}_k = \mathbf{x}_{i_k - 1}, \quad \text{for} \quad k = 1, 2, \ldots, q.$$

We have for $k = 1, 2, \ldots, q - 1$,

$$\mathbf{y}_{k+1}\mathbf{c} = B_{i_{k+1}-1}(\delta_{i_q} - \delta_{i_{k+1}}) < B_{i_k - 1}\frac{1}{p^{3p}}(\delta_{i_q} - \delta_{i_k}) = \frac{1}{p^{3p}}\mathbf{y}_k\mathbf{c}.$$

Thus vectors $\mathbf{c}$ and $\mathbf{y}_k$'s satisfy the conditions of Lemma 2, so $q \leq p$. It means that there are at most $O(p^2 \log p)$ consecutive iterations for which Inequality (8) holds. ∎

Lemmas 3 and 4 immediately imply the following strongly polynomial bound on the number of iterations.

Theorem 3 *Newton's method applied to an LFCO problem finds the optimum in $O(p^4 \log^2 p)$ iterations.*

If the weights are nonnegative, then there are at most $O(p \log p)$ iterations with $B_{k+1} \leq \frac{2}{3}B_k$. Therefore in such a case Newton's method works in $\bar{O}(p^3)$ iterations.

3 Minimum maximum arc cost transshipment problem

A *(transshipment) network* $G = (V, E, u, d)$ is a digraph (V,E) with a capacity function $u : E \longrightarrow \mathbf{R} \cup \{+\infty\}$, and a demand function $d : V \longrightarrow \mathbf{R}$ such that $\sum_{v \in V} d(v) = 0$. We

assume, for convenience in presentation but without losing generality, that E is symmetric and u is nonnegative. Negative $d(v)$ means that v is a *source* - a vertex with supply, positive $d(v)$ means that v is a *sink* - a vertex with demand. Let n and m denote the cardinality of V and the half of the cardinality of E, respectively. We assume $m \geq n$.

We adopt the convention that any function $\varphi : F \longrightarrow \mathbf{R}$ is extended to all finite subsets of F: $\varphi(A) = \sum_{x \in A} \varphi(x)$. Thus if $A \subseteq V$, then $d(A) = \sum_{v \in A} d(v)$, which is the net demand in A. The *total demand* in G is $d(G) = \sum_{v:d(v)>0} d(v)$.

A *pseudoflow* in G is an antisymmetric function $f : E \longrightarrow \mathbf{R}$, i.e., for every $(v, w) \in E$, $f(v, w) = -f(w, v)$. A pseudoflow f creates excesses at vertices: the excess at vertex v is $e^f(v) = \sum_{w:(w,v) \in E} f(w, v)$. If $e^f(v) = d(v)$, then we say that pseudoflow f satisfies the demand at vertex v. A *flow* is a pseudoflow f that meets capacity constraints and does not increase demands: $f(w, v) \leq u(w, v)$, for every arc $(w, v) \in E$, and $e^f(v)$ is between 0 and $d(v)$, for every $v \in V$. A *transshipment* is a flow which satisfies demands at all vertices. G is *feasible* if there is a transshipment in G. The *transshipment feasibility* problem is the problem of verifying if a given network G is feasible. For a pseudoflow f in G, the *residual network* G^f is $(V, E, u^f = u - f, d^f = d - e^f)$. $G_{\bar{u}}$ denotes G with the capacity function changed to $\bar{u}$. Thus $G_{\bar{u}}^f = (V, E, \bar{u}^f, d^f)$.

If S and T partition V, then *cut* (S, T) in G is the set of arcs (v, w) of E such that $v \in S$ and $w \in T$. The *capacity* and the *surplus* of a cut (S, T) are $u(S, T) = \sum_{e \in (S,T)} u(e)$ and $surplus(S, T) = d(T) - u(S, T)$, respectively. Our definition of $surplus(S, T)$ is the same as in [7] the definition of $V(T, S)$, the value of a cut (T, S). It is easy to verify that the surplus of a cut is the same in G as in G^f for any flow f. A positive surplus means that the cut blocks the flow. Any flow must leave at S at least the amount $surplus(S, T)$ of the commodity, which is demanded at T. Let $mean(S, T) = surplus(S, T)/|(S, T)|$ be the *mean surplus* of (S, T) in G, and, if a weight function $b : E \longrightarrow \mathbf{R}$ is specified, $mean\_w(S, T) = surplus(S, T)/b(S, T)$ be the *mean-weight surplus* of (S, T) in G.

The *minimum cost transshipment* problem, which is often refer to as the *transshipment problem*, is to compute for a given network G and a given cost function $c : E \longrightarrow \mathbf{R}$, the minimum cost of a transshipment in G, when the cost of transshipment f is $\sum_{e \in E} f(e)c(e)$. This problem has been well studied, often as the *minimum cost circulation* problem (these problems are reducible to each other in linear time). Fastest currently known algorithms for this problem can be found in [1, 2, 4, 9, 12]. In this paper we are interested in the problem of minimizing the maximum arc cost instead of the sum of arc costs. Here is the list of the

8

transshipment problems we consider. They are closely related to each other. We assume that cost functions and weight functions are nonnegative.

Minimum maximum arc cost transshipment problem (MAC)

A transshipment network G and a cost function $c : E \longrightarrow \mathbb{R}$ are given. Compute the minimum of the maximum arc cost over all transshipments in G, that is, compute

$$\delta^* = \min\{\max_{e \in E}\{f(e)c(e)\} \mid f \text{ is a transshipment in } G\}.$$

Minimum maximum capacity violation cost transshipment problem (MCVC)

A transshipment network G is given, which might not be feasible. The objective is to obtain a feasible network by increasing the capacities. A weight function $b : E \longrightarrow \mathbb{R}$ is given, which specifies by how much the capacities of the arcs can be increased in unit cost. It means that the cost of increase of the capacity of arc e from $u(e)$ to $u(e) + \xi$ is $\xi/b(e)$. Compute the minimum of the maximum cost of increasing the capacity of an arc, that is, compute

$$\delta^* = \min\{\delta \geq 0 \mid G_{u+\delta b} \text{ is feasible}\}.$$

Minimum maximum capacity violation transshipment problem (MCV)

This is the MCVC problem with the uniform weight function: $b \equiv 1$.

Maximum mean-weight cut problem (MWC)

A transshipment network G and a weight function $b : E \longrightarrow \mathbb{R}$ are given. Compute the maximum mean-weight surplus over all cuts with nonnegative surplus, that is, compute

$$\delta^* = \max\{\frac{surplus(S,T)}{b(S,T)} \mid (S,T) \text{ cut in } G \text{ with nonnegative surplus}\}.$$

Maximum mean cut problem (MC)

This is the MWC problem with the uniform weight function. It means that we are asked to find

$$\delta^* = \max\{\frac{surplus(S,T)}{|(S,T)|} \mid (S,T) \text{ cut in } G \text{ with nonnegative surplus}\}.$$

A cut with the maximum nonnegative mean-weight surplus and a cut with the maximum nonnegative mean surplus we will call a *maximum mean-weight (surplus) cut* and a

9

maximum mean (surplus) cut, respectively. We formulated the above problems in a way that the minimum or the maximum value is sought, not the corresponding optimal flow or cut. We did it only for convenience in further presentation. The algorithm we discuss in the next section and actually all other algorithms for the above problems compute both the optimal value and the corresponding optimal structure (transshipment or cut).

As mentioned in [8], the problem of minimizing the maximum arc cost appears, for example, when it is desirable not to use an arc at its full capacity. Each arc e has some safety level $s(e)$ and the ratio $f(e)/s(e)$ is sought to be minimized, with the general objective of getting the maximal ratio as small as possible. The MAC problem captures also the following dynamic situation. The network is in continuous activity, that is, the sources continuously keep producing commodity and sending it to the sinks. The objective is to maximize the throughput. More formally, suppose $1/c(e)$ is the amount of the commodity which can be shipped through arc e in unit time. It means that the shipment of $f(e)$ units through e takes $f(e)c(e)$ time. Further suppose that every source s generates $-d(s)$ units of the commodity every τ units of time and sends it out, and every sink t has to get $d(t)$ units of the commodity every τ units of time. Then the optimal δ^* in the formulation of the MAC problem is equal to the minimal possible τ such that no congestion arises at any vertex.

Before discussing the previous work relevant to the problems in which we are interested, we first establish relation between them.

The *value of a flow* f is the amount by which f decreases the total demand: $value(f) = \sum_{v:d(v)>0} e^f(v)$. A *maximum flow* in G is a flow in G of the maximum value. The *maximum flow* problem is the problem of finding a maximum flow in a given transshipment network. A *maximum (surplus) cut* is a cut with the maximum nonnegative surplus. The *maximum (surplus) cut* problem is the problem of finding a maximum cut in a given transshipment network.

The transshipment feasibility problem, the maximum flow problem and the maximum cut problem are the nonfractional/nonparametric problems that correspond to our fractional/parametric problems. The transshipment feasibility problem and the maximum flow problem can be easily reduced to the standard maximum flow problem. A maximum flow in a transshipment network can be used to find a maximum surplus cut, in a similar way as a maximum flow in a standard maximum flow network can be used to identify a minimum capacity cut. Let $T_{\text{MF}}(n, m)$ denote $O(M(n, m))$, where $M(n, m)$ is the time complex-

ity of the standard maximum flow problem. The transshipment feasibility, the maximum flow in a transshipment network, and the maximum surplus cut problems can be solved in $T_{\mathrm{MF}}(n, m)$ time. Currently it is known that $T_{\mathrm{MF}}(n, m) = O(nm \log(n^2/m))$ [3], and slightly better bounds exist for dense graphs.

Let for a transshipment problem P, $T_{\mathrm{P}}(n, m)$ denote its time complexity. The following theorem shows close relation between the MAC problem and the MCVC problem.

Theorem 4

1. $T_{\mathrm{MCVC}}(n, m) \leq O(m) + T_{\mathrm{MAC}}(n, 2m)$.

2. $T_{\mathrm{MAC}}(n, m) \leq O(m \log m) + T_{\mathrm{MF}}(n, m) O(\log m) + T_{\mathrm{MCVC}}(n, m)$.

Proof.

1. For a given instance I of the MCVC problem construct an instance I' of the MAC problem in the following way. Replace each arc e in I with two parallel arcs e_1 and e_2. Put $u'(e_1) = u(e)$, $c'(e_1) = 0$, $u'(e_2) = +\infty$, $c'(e_2) = 1/b(e)$. The demand function in I' is the same as in I. A solution (transshipment) to I' gives in the natural way the solution to I.

2. Let δ^* be the solution to the given instance I of the MAC problem. It is easy to see that $\delta^* \leq \delta$ if and only if $G_{\min\{u, \delta/c\}}$ is feasible. Therefore

$$\delta^* = \min\{\delta \geq 0 \mid G_{\min\{u, \delta/c\}} \text{ is feasible}\}.$$

To get an instance of the MCVC problem, it is enough to know for each arc e, if $u(e)$ is essential, i.e., if $u(e) < \delta^*/c(e)$. We sort the list $\{u(e)c(e) \mid e \in E\}$ and using binary search we find the position of δ^* in it. This takes $T_{\mathrm{MF}}(n, m) O(\log m)$ time. Now we construct an instance I' of the MCVC problem. The underlying graph and the demand function remain without changes. The capacities and weights are defined as follows.

$$(u'(e), b'(e)) = \begin{cases} (u(e), 0) & \text{if } u(e)c(e) \leq \delta^*, \\ (0, 1/c(e)) & \text{if } u(e)c(e) > \delta^*. \end{cases}$$

δ^*, the solution to I, is also the solution to I'.

∎

Let $mean\_w_\delta(S,T)$ denote the mean-weight surplus of (S,T) in $G_{u+\delta b}$. Then

$$mean\_w_\delta(S,T) = mean\_w(S,T) - \delta.$$

It means that the maximum $mean\_w(S,T)$ is equal to the minimum δ for which $G_{u+\delta b}$ does not contain cuts with positive surplus. Therefore the following theorem implies that the MCVC problem and the MWC problem are equivalent (and so are MCV and MC), *i.e.*,

$$\min\{\delta \geq 0 \mid G_{u+\delta b} \text{ is feasible}\} =$$
$$\max\{mean\_w(S,T) \mid (S,T) \text{ cut in } G \text{ with nonnegative surplus}\}.$$

Theorem 5 [5] G is feasible iff there is no cut in G with positive surplus.

The MWC problem is an LFCO problem. $\mathcal{X}_G \subseteq \{0,1\}^{n+m}$ corresponds to the set of all cuts in G. Vector $\mathbf{x} \in \mathcal{X}_G$ represents cut (S,T) such that for $1 \leq i \leq n$, $x_i = 1$ iff $v_i \in T$, and for $1 \leq j \leq m$, $x_{n+j} = 1$ iff $e_j \in (S,T)$, where v_i is the ith vertex and e_j is the jth arc. Vector $\mathbf{a}$ from the formulation of an LFCO problem is here $(\mathbf{d}, -\mathbf{u})$, where $\mathbf{d}$ and $\mathbf{u}$ are the vector representations of functions d and u. If $\mathbf{x}$ represents (S,T), then $\mathbf{ax}$ and $\mathbf{bx}$ are now $surplus(S,T)$ and $b(S,T)$.

Megiddo [8] showed how to solve the MAC problem with his parametric search method. This method gives an $\bar{O}(n^3 m)$ running time and can give a better time only if a breakthrough in parallel maximum flow computation is achieved. Binary search solves MAC with integral data in $O(T_{\mathrm{MF}}(n,m)\log(nU))$ time. More is known about the case with uniform weights. McCormick and Ervolina [7] proposed an algorithm for computing a maximum mean cut. This algorithm is essentially an application of Newton's method. They showed an $O(m)$ bound on the number of iterations (see Theorem 2). One iteration is dominated by maximum flow computation. We were able to prove that the number of iterations in this algorithm is in fact $\Theta(n)$ [10], so the time complexity of MC, as well as MCV and MAC with uniform costs, is $\bar{O}(n^2 m)$. For these problems the bound $O(T_{\mathrm{MF}}(n,m)\log(nU))$ can be improved to $O(nm\log(nU))$ by using approximate maximum flow computation in connection with either Newton's method [10] or binary search [6].

In the next section we prove an $O(m)$ bound on the number of iterations in Newton's method applied to the MCVC problem. This implies an $O(T_{\mathrm{MF}}(n,m)m) = \bar{O}(m^2 n)$ bound on the time complexity of MCVC, MAC and MWC.

4 Newton's method for the MCVC problem

A transshipment network G and a (nonnegative) weight function b are given. We assume that G is not feasible. G_δ stands for $G_{u+\delta b}$, and generally subscript δ indicates that the underlying network is $G_{u+\delta b}$. We want to find

$$\delta^* = \min\{\delta \mid G_\delta \text{ is feasible}\},$$

which, as mentioned in the previous section, is equal to

$$\max\{mean\_w(S,T) \mid (S,T) \text{ cut in } G\}.$$

The function $h(\delta)$ used in the discussion of a general LFCO problem in Section 1 is now

$$h(\delta) = \max\{surplus_\delta(S,T) \mid (S,T) \text{ cut in } G\}.$$

$h(\delta)$ is also equal to the total demand left in G_δ^f, where f is a maximum flow in G_δ.

We describe the way Newton's method finds δ^*, the root of $h(\delta)$. Let $\bar\delta \le \delta^*$ be the current (under)estimation of δ^*. Initially $\bar\delta = 0$. During one iteration we find a maximum cut (S,T) in $G_{\bar\delta}$. This is done by computing a maximum flow $\bar f$ in $G_{\bar\delta}$. If the surplus of (S,T) is zero, i.e., if $\bar f$ is a transshipment, then $\delta^* = \bar\delta$ and algorithm terminates. Otherwise we compute the next estimation $\bar\delta = mean\_w(S,T)$ and go to the next iteration. Moving to the next estimation should be seen as increasing the capacity so that the surplus of (S,T) decreases to zero.

For any flow f in G, the surpluses of cuts are the same in G as in G^f. It means that in the current iteration we can perform computation on any residual network of $G_{\bar\delta}$ instead of on $G_{\bar\delta}$ itself. In the analysis we assume that the network in the current iteration is $G_{\bar\delta}^{\hat f}$, where $\hat f$ is the sum of the maximum flows computed in all previous iterations. It means that $\hat f$ is a maximum flow in $G_{\delta'}$, where δ' is $\bar\delta$ from the previous iteration.

Let δ_i be $\bar\delta$ at the beginning of iteration i, and (S_i, T_i) be the cut found in this iteration. B_i, and H_i from Section 2 have here the following meaning.

$$
\begin{aligned}
B_i &= b(S_i, T_i), \\
H_i &= h(\delta_i) = surplus(S_i, T_i) - \delta_i b(S_i, T_i).
\end{aligned}
$$

Let f_i be the sum of the flows computed through iteration i. Flow f_i is a maximum flow in G_{δ_i}. H_i is the total demand in $G_{\delta_i}^{f_i}$.

Theorem 1 implies that if the capacities and the weights are integers from $[-U, U]$, then the algorithm runs in $O(\log(nU))$ iterations. The weights are nonnegative, so the remark after Theorem 3 implies the strongly polynomial bound of $\tilde{O}(m^3)$ on the number of iterations. In this section we show an $O(m)$ bound on the number of iterations.

An arc e is *unessential* in G, if its capacity is greater than the total demand or its weight is greater than the weight of a maximum surplus cut. When we use expression "e is unessential" in the context of the algorithm, we mean that e is unessential in the current network. If an arc is unessential in G_δ^f, for some $\delta \geq$ and flow f in G_δ, then it is unessential in $G_{\delta+\delta'}^{f+f'}$, for any $\delta \geq 0$ and any flow f' in $G_{\delta+\delta'}^f$: Increasing δ increases the capacity of an arc and decreases the maximum surplus of a cut; augmenting with a flow may decrease the capacity of an arc but by not more than it decreases the total demand. Therefore if at some point in the algorithm some arc is unessential, it remains unessential through the end of the computation. An unessential arc cannot belong to a maximum surplus cut. We show that few iterations are enough to make a new arc unessential.

Lemma 5 From the iteration $i + 2$ on, the (current) capacity of cut (S_i, T_i) is greater than H_{i+1}.

Proof. At the beginning of iteration $i + 1$ the capacity of cut (S_i, T_i) is equal, by definition, to H_i, the remaining total demand. In iteration $i + 1$ the capacity of each cut decreases by at most $H_i - H_{i+1}$ and then increases by $(H_{i+1}/B_{i+1})B$, where B is the weight of this cut. It means that in iteration $i + 1$ the capacity of cut (S_i, T_i) first decreases but does not go below H_{i+1}. Then it increases by more than H_{i+1}, because its weight is greater than B_{i+1}. Therefore at the end of iteration $i + 1$ the capacity of (S_i, T_i) is greater than $2H_{i+1}$. From now on the capacity of (S_i, T_i) is always greater than H_{i+1}, because the total decrease of this capacity cannot be greater than the total demand left after iteration $i + 1$, which is H_{i+1}.

Putting the above argument in a more formal way we get the following bound on the capacity of (S_i, T_i) at the beginning of iteration $i+l$, where $l \geq 2$. According to our notation, the capacity function at the beginning of iteration $i + l$ is $u_{\delta_{i+l}}^{f_{i+l-1}}$.

$$
\begin{aligned}
u_{\delta_{i+l}}^{f_{i+l-1}}(S_i, T_i) &\geq u_{\delta_i}^{f_i}(S_i, T_i) + (\delta_{i+l} - \delta_i)B_i - (H_i - H_{i+l-1}) \\
&= (\delta_{i+l} - \delta_{i+1})B_i + (\delta_{i+1} - \delta_i)B_i - H_i + H_{i+l-1} \\
&= (\delta_{i+l} - \delta_{i+1})B_i + H_{i+l-1} \\
&\geq (\delta_{i+2} - \delta_{i+1})B_i
\end{aligned}
$$

$$> (\delta_{i+2} - \delta_{i+1})B_{i+1}$$
$$= H_{i+1}.$$

∎

We first prove an $O(m \log m)$ bound to show the main idea. To prove an $O(m)$ bound, we will need a finer accounting strategy.

Theorem 6 *Newton's method solves MCVC in $O(m \log m)$ iterations.*

Proof. We use Inequality (4) to show that after $O(\log m)$ iterations a new arc becomes unessential. Let the current iteration be the ith one. Let $l = \lfloor \log m \rfloor + 2$. It follows from Inequality (4) that

$$H_{i+l}B_{i+l} < \frac{1}{m^2}H_{i+1}B_{i+1}. \tag{10}$$

If $B_{i+l} < \frac{1}{m}B_i$, then there exists an arc $e \in (S_i, T_i)$ such that $b(e) > B_{i+l}$. Such an arc is unessential from iteration $i + l$ on.

If $B_{i+l} \geq \frac{1}{m}B_i$, then $B_{i+l} \geq \frac{1}{m}B_{i+1}$ as well, and Inequality (10) implies that $H_{i+l} < \frac{1}{m}H_{i+1}$. Lemma 5 says that the capacity of (S_i, T_i) at the beginning of iteration $i + l + 1$ is greater than H_{i+1}, so its greater than mH_{i+l}. It means that the capacity of some arc in (S_i, T_i) in $G^{f_{i+l}}_{\delta_{i+l+1}}$ (which is the network at the beginning of iteration $i + l + 1$) is greater than H_{i+l}, the total demand in $G^{f_{i+l}}_{\delta_{i+l+1}}$. Such an arc is unessential from iteration $i + l + 1$ on. ∎

Theorem 7 *Newton's method solves MCVC in $O(m)$ iterations.*

Proof. Lemma 1 implies that for each iteration i,

$$\frac{B_i}{B_{i-1}} \leq \frac{1}{2} \tag{11}$$

or

$$\frac{H_i}{H_{i-1}} \leq \frac{1}{2}. \tag{12}$$

We first bound the number of iterations i for which (11) holds. Let $(\mu_i)_{i=1}^q$ be the sequence of B_i's in these iterations and $(\alpha_j)_{j=1}^p$ be the positive b_j's arranged in nonincreasing order. It is easy to see that sequences $(\alpha_j)_{j=1}^p$ and $(\mu_i)_{i=1}^q$ satisfy the conditions of Lemma 6, so $q \leq p \leq m$.

Now we bound the number of iterations i for which (12) holds. Here the argument is more involved, because H_i's are not just subsums of a set of m elements. They are related to the current demands and capacities, which vary from iteration to iteration.

To avoid towering subscripts, we renumber iterations taking into an account only iterations with (12). It means that now iteration i is what used to be the ith iteration with (12). All subscripts refer to the new numbering.

We assign iterations to arcs in such a way that at most 5 iterations are assigned to one arc. We stop the process of assigning when all but at most $q + 2 = \lceil \log m \rceil + 5$ iterations have been assigned. Assume that there are still at least $q + 3$ unassigned iterations. Let iteration i be the first among them. Consider cut (S_i, T_i). Let p be its cardinality. Let for $1 \le j \le p$ and $1 \le l \le q$, $\gamma_{l,j}$ be the capacity of the jth arc in (S_i, T_i) (assuming any order on the arcs in (S_i, T_i)) at the beginning of iteration $i + l + 2$. It follows from Lemma 5 that for each $1 \le l \le q$,

$$\sum_{j=1}^{p} \gamma_{l,j} \ge H_{i+1}. \tag{13}$$

For each arc, the difference between its capacities at the beginning of iterations $k + 1$ and $k + 2$ is not greater than the total demand at the beginning of iteration $k + 1$, which is H_k. It means that

$$|\gamma_{l,j} - \gamma_{l+1,j}| \le H_{i+l+1} \le \frac{1}{2^l} H_{i+1}. \tag{14}$$

The definition of q, (13), and (14) imply that matrix $(\gamma_{l,j}/H_{i+1})$ satisfies the conditions of Lemma 7. It means that there exists $k \ge 3$ such that at least $k - 2$ elements from $\gamma_{k,1}, \gamma_{k,2}, \ldots, \gamma_{k,p}$ are greater than $1/2^k H_{i+1} \ge H_{i+k+1}$. Therefore the arcs corresponding to these $k - 2$ elements are unessential at iteration $i + k + 2$. We assign iterations i through $i + k + 1$ to these arcs. Notice that none of these arcs was unessential at iteration i, so none of the previous iteration was assign to any of them.

During this process of assigning iterations to arcs, no more than 5 iterations are assign to one arc, and no more than $O(\log m)$ iterations are left unassigned. We conclude that there are at most $5m + O(\log m)$ iterations for which (12) holds. ∎

Lemma 6 Let $\alpha_1 \ge \alpha_2 \ge \cdots \ge \alpha_p > 0$ and $\mu_1 > \mu_2 > \ldots > \mu_q > 0$ be such that

1. $\mu_{i+1} \le \frac{1}{2} \mu_i$, for $i = 1, 2, \ldots, q - 1$,

2. $\mu_q \ge \alpha_p$,

3. $\mu_i \le \sum \{\alpha_j \mid \alpha_j \le \mu_i\}$, for $i = 1, 2, \ldots, q$.

Then $q \le p$.

16

Proof. Let $\bar{\alpha}_j = \alpha_j + \alpha_{j+1} + \cdots + \alpha_p$. Condition 3 implies that $\mu_1 \le \bar{\alpha}_1$. Thus, to prove $q \le p$, it is sufficient to show that each of the intervals $(0, \bar{\alpha}_p]$, $(\bar{\alpha}_p, \bar{\alpha}_{p-1}]$, ..., $(\bar{\alpha}_3, \bar{\alpha}_2]$, $(\bar{\alpha}_2, \bar{\alpha}_1]$ contains at most one element from (μ_i). Condition 2 implies that only the last element can be in $(0, \bar{\alpha}_p]$. Let for some $1 \le j \le p - 1$ and $1 \le i \le q - 1$, $\mu_i \in (\bar{\alpha}_{j+1}, \bar{\alpha}_j]$. We show that $\mu_{i+1} \le \bar{\alpha}_{j+1}$.

If $\bar{\alpha}_{j+1} \ge \frac{1}{2}\bar{\alpha}_j$, then

$$\mu_{i+1} \le \frac{1}{2}\mu_i \le \frac{1}{2}\bar{\alpha}_j \le \bar{\alpha}_{j+1}.$$

If $\bar{\alpha}_{j+1} < \frac{1}{2}\bar{\alpha}_j$, then

$$\mu_{i+1} \le \frac{1}{2}\mu_i \le \frac{1}{2}\bar{\alpha}_j = \frac{1}{2}\bar{\alpha}_j + (\alpha_j + \bar{\alpha}_{j+1} - \bar{\alpha}_j) = \left(\bar{\alpha}_{j+1} - \frac{1}{2}\bar{\alpha}_j\right) + \alpha_j < \alpha_j.$$

The above inequality and Condition 3 imply $\mu_{i+1} \le \bar{\alpha}_{j+1}$. ∎

Lemma 7 Let $(\alpha_{i,j})$ be a $q \times p$ matrix such that

1. $q \ge \log p + 3$,

2. the sum of each row is not less than 1,

3. $|\alpha_{i+1,j} - \alpha_{i,j}| \le 1/2^i$, for $1 \le i \le q$ and $1 \le j \le p$.

If $\alpha_{i,j} > 1/2^i$, then we call this and all subsequent elements in column j good elements. There is $k \ge 3$ such that row k contains at least $k - 2$ good elements.

Proof. Condition 3 implies that if $1 \le i' < i'' \le q$ and $1 \le j \le p$, then

$$\alpha_{i'',j} < \alpha_{i',j} + \frac{1}{2^{i'-1}}. \tag{15}$$

If $\alpha_{i,j}$ is the first good element in column j and $i \ge 2$, then

$$\alpha_{i,j} \le \alpha_{i-1,j} + \frac{1}{2^{i-1}} \le \frac{1}{2^{i-1}} + \frac{1}{2^{i-1}} = \frac{1}{2^{i-2}}. \tag{16}$$

Assume that for all $k \ge 3$, row k contains at most $k - 3$ good elements. We will get contradiction by showing that the sum of the last row is less than 1. Let $j_1, j_2, \ldots, j_l$ be the indices of all columns with at least one good element. Let $\alpha_{i_1,j_1}, \alpha_{i_2,j_2}, \ldots, \alpha_{i_l,j_l}$ be the first good elements in these columns. Let $j_1, j_2, \ldots, j_l$ be ordered in such a way that $i_1 \le i_2 \le \cdots \le i_l$. Row i_k contains at least k good elements $(\alpha_{i_k,j_1}, \alpha_{i_k,j_2}, \ldots, \alpha_{i_k,j_k})$ so

$$i_k \ge k + 3. \tag{17}$$

17

The sum of the last row is at most

$$\frac{p-l}{2^q} + \alpha_{q,j_1} + \alpha_{q,j_2} + \cdots + \alpha_{q,j_l}$$

$$\leq \frac{p}{2^q} + (\alpha_{i_1,j_1} + \frac{1}{2^{i_1-1}}) + \cdots + (\alpha_{i_l,j_l} + \frac{1}{2^{i_l-1}})$$

$$\leq \frac{1}{8} + (\frac{1}{2^{i_1-2}} + \frac{1}{2^{i_1-1}}) + \cdots + (\frac{1}{2^{i_l-2}} + \frac{1}{2^{i_l-1}})$$

$$= \frac{1}{8} + 6 \left(\frac{1}{2^{i_1}} + \cdots + \frac{1}{2^{i_l}} \right)$$

$$\leq \frac{1}{8} + 6 \left(\frac{1}{2^4} + \frac{1}{2^8} + \cdots \right)$$

$$< 1.$$

The first inequality follows from (15), the second one from (16) and Condition 1, and the third one from (17). ■

5   Concluding Remarks

We proved that Newton's method solves a linear fractional combinatorial optimization problem in a strongly polynomial number of iterations. We did it by combining the fast convergence of the method with the limiting way the numbers involved in an LFCO problem can be spread along the real line. That "limiting spread" follows from the fact that these numbers are obtained using only few multiplications/divisions. Observe that for example the generalized flow problem does not have this property, and we do not know how our type of analysis could be useful in such a case.

We showed that Newton's method gives an $O(m)$ iteration algorithm for the transshipment problem when the maximum arc cost is minimized. One iteration in this algorithm is dominated by the maximum flow computation, so the overall running time is $\tilde{O}(m^2 n)$. This improves the upper bound achieved with Megiddo's parametric search. We believe that the bound on the number of iterations is $O(n)$, but we have not been able to prove it.

We may have a transshipment network with two cost functions, one related to minimizing the sum of arc costs and the other to minimizing the maximum arc cost. In the same $\tilde{O}(m^2 n)$ time bound we can perform the "double minimization". To minimize the sum of the arc costs over the transshipments which minimize the maximum arc cost, we first minimize the maximum arc cost and get in this way additional capacity constraints. Then we minimize the sum of arc costs taking into an account these additional constraints. To minimize the maximum arc cost over the transshipments which minimize the sum of arc costs, we first

compute a transshipment which minimizes the sum of arc costs, and its dual, and identify all arcs which have to be saturated in all such transshipments. Then we can minimize the maximum arc cost of the remaining arcs.

References

[1] R. K. Ahuja, A. V. Goldberg, J. B. Orlin, and R. E. Tarjan. Finding Minimum-Cost Flows by Double Scaling. Technical Report STAN-CS-88-1227, Department of Computer Science, Stanford University, 1988.

[2] J. Edmonds and R. M. Karp. Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems. *J. Assoc. Comput. Mach.*, 19:248–264, 1972.

[3] A. V. Goldberg and R. E. Tarjan. A New Approach to the Maximum Flow Problem. *J. Assoc. Comput. Mach.*, 35:921–940, 1988.

[4] A. V. Goldberg and R. E. Tarjan. Finding Minimum-Cost Circulations by Successive Approximation. *Math. of Oper. Res.*, 15:430–466, 1990.

[5] A. J. Hoffman. A Generalization of Max Flow - Min Cut. *Math. Prog.*, 6:352–359, 1974.

[6] K. Iwano, S. Misono, S. Tezuka, and S. Fujishige. A New Scaling Algorithm for the Maximum Mean Cut Problem. Unpublished manuscript (To appear in *Algorithmica*), 1990.

[7] S. T. McCormick and T. R. Ervolina. Computing Maximum Mean Cuts. UBC Faculty of Commerce Working Paper 90-MSC-011, 1990.

[8] N. Megiddo. Applying Parallel Computation Algorithms in the Design of Serial Algorithms. *J. Assoc. Comput. Mach.*, 30:852–865, 1983.

[9] J. B. Orlin. A Faster Strongly Polynomial Minimum Cost Flow Algorithm. In *Proc. 20th Annual ACM Symposium on Theory of Computing*, pages 377–387, 1988.

[10] T. Radzik. Minimizing Capacity Violations in a Transshipment Network. In *Proc. 3rd ACM-SIAM Symposium on Discrete Algorithms*, 1992. (To appear).

[11] А. В. Карзанов. О Минимальных по Среднему Весу Разрезах и Циклах Ориентированного Графа. Сб. *Качественные и Приближенные Методы Исследования Операторных Уравнений*, 72–83. Ярославский Государственный Университет, Ярославл, СССР, 1985. Title translation: A. V. Karzanov "On minimal mean cuts and circuits in a digraph," in *Methods for Solving Operator Equations*, Yaroslavl, USSR, 1985.

[12] Е. А. Диниц. Метод Поразрядного Сокращения Невязок и Транспортные Задачи. Сб. *Исследования по Дискретной Математике*. Наука, Москва, 1973. English transcription: E. A. Dinic, "Metod Porazryadnogo Sokrashcheniya Nevyazok i Transportnye Zadachi," *Issledovaniya po Diskretnoi Matematike*, Science, Moscow. Title translation: The Method of Scaling and Transportation Problems.

Reproduced by NTIS
National Technical Information Service
U.S. Department of Commerce
Springfield, VA 22161

This report was printed specifically for your order from our collection of more than 2 million technical reports.

For economy and efficiency, NTIS does not maintain stock of its vast collection of technical reports. Rather, most documents are printed for each order. Your copy is the best possible reproduction available from our master archive. If you have any questions concerning this document or any order you placed with NTIS, please call our Customer Services Department at (703) 387-4660.

Always think of NTIS when you want:
• Access to the technical, scientific, and engineering results generated by the ongoing multibillion dollar R&D program of the U.S. Government.
• R&D results from Japan, West Germany, Great Britain, and some 20 other countries, most of it reported in English.

NTIS also operates two centers that can provide you with valuable information:
• The Federal Computer Products Center - offers software and datafiles produced by Federal agencies.
• The Center for the Utilization of Federal Technology - gives you access to the best of Federal technologies and laboratory resources.

For more information about NTIS, send for our FREE NTIS Products and Services Catalog which describes how you can access this U.S. and foreign Government technology. Call (703) 487-4650 or send this sheet to NTIS, U.S. Department of Commerce, Springfield, VA 22161. Ask for catalog, PR-827.

Name_____

Address_____

_____

_____

Telephone_____

- Your Source to U.S. and Foreign Government Research and Technology