ELSEVIER

Operations Research Letters 25 (1999) 205–211

operations research letters

www.elsevier.com/locate/orms

A new scaling algorithm for the minimum cost network flow problem [☆]

Donald Goldfarb[a, *], Zhiying Jin[b]

[a] *Department of Industrial Engineering and Operations Research, Columbia University, New York, NY 10027, USA*
[b] *GTE Laboratories Inc., 40 Sylvan Road, Waltham, MA 02454, USA*

Received 1 December 1996; received in revised form 1 June 1999

Abstract

In this paper, we present a new polynomial time algorithm for solving the minimum cost network flow problem. This algorithm is based on Edmonds–Karp's capacity scaling and Orlin's excess scaling algorithms. Unlike these algorithms, our algorithm works directly with the given data and original network, and dynamically adjusts the scaling factor between scaling phases, so that it performs at least one flow augmentation in each phase. Our algorithm has a complexity of $O(m(m + n\log n)\log(B/(m + n)))$, where n is the number of nodes, m is the number of arcs, and B is the sum of the finite arc capacities and supplies in the network. © 1999 Elsevier Science B.V. All rights reserved.

Keywords: Polynomial algorithm; Excess scaling algorithm; Minimum cost network flow problem

1. Introduction

We present in this paper a new polynomial time algorithm for the *minimum cost network flow* problem in which all supplies and demands and finite arc capacities are integral. Specifically, let $N = (G, u, b, c)$ be a directed network corresponding to a directed graph $G = (V, E)$ with node set V and arc set E. For the sake of simplicity, we assume that G has no multiple or oppositely directed arcs; hence, if there is an arc from node v to node w, it is unique and is denoted by (v, w) and there is no arc (w, v) in G. We note that our results extend readily to networks with multiple and oppositely directed arcs. Each directed arc $(v, w) \in E$ has a real-valued cost $c(v, w)$ and a nonnegative, possibly infinite, integral capacity $u(v, w)$ associated with it. Each node $v \in V$ has an integral supply $b(v)$ ($\geqslant 0$) or demand $b(v)$ (< 0) associated with it, where $\sum_{v \in V} b(v) = 0$. The objective of the minimum cost network flow problem is to find a flow that has the least cost while

☆ This research was supported in part by NSF Grants DMS 94-14438, DMS 95-27124 and CDA-97-26385 and DOE Grant DE-FG02-92ER25126.

* Corresponding author. Fax: +1-212-854-8103.
E-mail addresses: gold@ieor.columbia.edu (D. Goldfarb), zjin@gte.com (Z. Jin)

satisfying flow conservation and capacity constraints. Mathematically, this problem can be formulated as the following linear program.

$$\text{Minimize} \quad \sum_{(v,w)\in E} c(v,w)x(v,w)$$

$$\text{subject to} \quad \sum_{(v,w)\in E} x(v,w) - \sum_{(w,v)\in E} x(w,v) = b(v), \quad \forall v \in V$$

$$0 \leqslant x(v,w) \leqslant u(v,w), \quad \forall (v,w) \in E.$$

The minimum cost network flow problem has been studied extensively, and many polynomial and strongly polynomial algorithms for solving it have been proposed (e.g., see [2]). Edmonds and Karp [7] were the first to propose an algorithm capable of solving this problem in polynomial time. Their algorithm, known as the *capacity-scaling algorithm*, was also the first scaling algorithm to have been proposed. It reduces the minimum cost network flow problem to a sequence of shortest path problems. A variant of Edmond and Karp's capacity scaling algorithm that is known as the *excess-scaling algorithm* was proposed by Orlin [17]. This algorithm is designed for uncapacitated problems (i.e., transshipment problems). It breaks down the computations into a sequence of phases determined by a scaling factor Δ that is divided in half from one phase to the next. During a Δ-scaling phase, the current flow is augmented along shortest paths in the residual network from nodes with at least Δ units of excess to nodes with at least Δ units of deficit. To solve a capacitated problem by Orlin's algorithm, that problem first has to be transformed into an uncapacitated problem. This transformation enlarges the network, adding a node and arc for each capacitated arc, and makes the algorithm less efficient in practice. Both the capacity-scaling and excess-scaling algorithms can be implemented to run in $O(m(m+n\log n)\log U)$ time, where $n=|V|$, $m=|E|$, and $U=\max\{u(v,w),|b(v)|: \forall(v,w)\in E$, and $\forall v\in V\}$.

The new algorithm that we propose in this paper is an excess scaling algorithm. Our algorithm has a worst case complexity of $O(m(m+n\log n)\log(B/(m+n)))$, where $B \leqslant \sum_{b(v)>0} b(v) + \sum_{u(v,w)<\infty} u(v,w)$. Since $B/(m+n) < U$, the complexity of our algorithm is at least as good as the complexity of Edmonds and Karp's capacity scaling and Orlin's excess scaling algorithms. Moreover, it is better than the complexity of the latter algorithms when all arc costs are nonnegative as is the case in most problems, since in such a case $B=\sum_{b(v)>0} b(v)$, which equals the initial total excess in the network. Also, in contrast with the latter algorithms, ours works directly with the given data and network, and is simple and flexible in adjusting the scaling factor. The crucial aspect of our algorithm that enables it to solve a capacitated problem without first transforming it into an uncapacitated problem is its use of arc excess. Arc excess was introduced by Goldfarb and Jin [13] and used by Goldfarb et al. [15] to develop polynomial time algorithms for the generalized circulation problem.

Many scaling-based polynomial algorithms for solving the minimum cost network flow problem can be and have been modified to make them strongly polynomial, but there are also strongly polynomial algorithms for this problem that are not based on scaling (e.g., see [2]). The first strongly polynomial minimum cost network flow algorithm was developed by Tardos [20]. Currently, the fastest strongly polynomial algorithm for this problem is based on scaling and is due to Orlin [17]. Modified versions of this algorithm have been proposed by Orlin et al. [18] and Goldfarb and Jin [14]. Our new algorithm can also be modified to make it strongly polynomial with the same complexity bound as the algorithms in Refs. [14,17]. Since the proof of this is more complicated than the complexity proofs for the above algorithms, which work on uncapacitated problems, we have chosen not to include this strongly polynomial algorithm and its complexity proof in this paper. Moreover, under the similarity assumption that $B=O(n^k)$ for some fixed k, our polynomial-time algorithm is theoretically just as fast.

Recent computational results [4,5,11] show that the cost-scaling algorithms in [5,12,19] are efficient in practice. However, there are very few computational results on capacity-scaling and excess-scaling algorithms except those reported in [16] for the transportation problem and those in [9] for the uncapacitated minimum cost network flow problem.

2. Preliminaries

Throughout the rest of the paper the following is assumed to hold.

Assumption 1. The costs of all uncapacitated arcs are nonnegative.

If some uncapacitated arcs in N have negative costs, then the following procedure will transform them so that the resulting problem has the same solution as the original problem and satisfies Assumption 1. Let $\bar{G}=(V,\bar{E})$ denote the subgraph of G containing only those arcs in E that are uncapacitated — i.e., $\bar{E}=\{(v,w)\in E \mid u(v,w)=\infty\}$. Starting with node labels $d(v)=0$ for all $v\in V$, apply the Bellman–Ford–Moore label correcting algorithm (e.g., see [2]) to $\bar{G}$ to either determine labels $d(v)$ that satisfy $d(w)\leqslant c(v,w)+d(v)$ for all $(v,w)\in\bar{E}$, or detect a negative cycle. This computation, which requires at most $O(mn)$ time, corresponds to first adding to G, an artificial node s and uncapacitated artificial arcs (s,v) with zero costs from node s to all nodes $v\in V$, and then applying the Bellman–Ford–Moore algorithm to the resulting expanded network to determine the shortest paths from node s to all nodes $v\in V$. If no negative cycle is detected, then the costs $c(v,w)$ of all arcs $(v,w)\in E$ are replaced by $c(v,w)+d(v)-d(w)$. If a negative cycle is detected, then the original problem has an unbounded objective value. Rather than maintaining a "primal" feasible flow, the algorithm presented in the next section maintains a "pseudoflow", which is primal infeasible until termination. A *pseudoflow* is a function $f: E\rightarrow R^+$ that satisfies $0\leqslant f(v,w)\leqslant u(v,w)$ for all $(v,w)\in E$. Given a pseudoflow f, the *imbalance* at node v is defined as

$$e_f(v)\equiv b(v)+\sum_{(w,v)\in E} f(w,v)-\sum_{(v,w)\in E} f(v,w). \tag{1}$$

A positive imbalance $e_f(v)$ is referred to as an *excess* and a negative imbalance $e_f(v)$ is referred to as a *deficit*.

To simplify our presentation, we henceforth add to the network, for every arc $(v,w)\in E$, a reverse arc (w,v) with cost $c(w,v)=-c(v,w)$, capacity $u(w,v)=0$ and pseudoflow $f(w,v)=-f(v,w)$. Consequently, in the rest of the paper, E and its cardinality m refer to the set of original and reverse arcs. Given a pseudoflow f, the *residual capacity* of arc (v,w) is defined as $u_f(v,w)=u(v,w)-f(v,w)$. The *residual graph* with respect to a pseudoflow f is defined as the directed graph $G_f=(V,E_f)$ where $E_f=\{(v,w)\in E\colon u_f(v,w)>0\}$. Given a node $k\in G_f$, we denote the set of nodes that are reachable in G_f from k by $V_f(k)$, i.e.,

$$V_f(k)=\{v\in V \mid \text{there is a directed path in } G_f \text{ from } k \text{ to } v\}.$$

A *dual variable* is a function $p: V\rightarrow R$. Given a dual variable p, the *reduced cost* $c_p(v,w)$ of arc (v,w) with respect to p is defined as $c_p(v,w)=c(v,w)+p(v)-p(w)$. Throughout our algorithm the dual feasibility and complementary slackness conditions

$$c_p(v,w)\geqslant 0 \quad \text{for all } (v,w)\in G_f \tag{2}$$

hold for some dual variable p, and an optimal solution is obtained by progressively reducing the total amount of excess and the total amount of deficit in the network until all imbalances are zero and the pseudoflow f is a "primal" feasible flow.

3. The algorithm

Conditions (2) are "optimality" conditions for a feasible flow f to be an optimal flow. Like both the Edmonds and Karp and Orlin scaling algorithms, our algorithm is a scaling version of the successive shortest-path algorithm. That algorithm is based upon the following well-known result (e.g., see [2, Section 9.7] or [17]).

Lemma 1. *Let f be a pseudoflow (or flow) and p be a dual variable that satisfy the optimality conditions (2). Let f' be obtained from f by sending flow along a shortest path from node k to node t in G_f, where the length of an arc (v,w) in G_f is given by its reduced cost $c_p(v,w)$. Let p' be set to $p+d$, where $d(v)$ is the shortest-path distance in G_f from node k to node v, for all nodes $v \in V_f(k)$, and $d(v)=0$, otherwise. Then f' and p' also satisfy the optimality conditions (2).*

Our algorithm begins by saturating all negative cost arcs yielding a pseudoflow that satisfies the optimality conditions (2) for $p=0$. Note that if all arcs have nonnegative costs, then the initial total excess is equal to the sum of the supplies. Our algorithm keeps a scaling factor Δ as does Orlin's algorithm [17]. A period during which Δ remains constant is called a *Δ-scaling phase*. In each Δ-scaling phase, our algorithm repeats the following procedure until there are no nodes with excess greater than Δ. This procedure first finds a shortest path from a node k with $e_f(k) \geqslant \Delta$ to a node t with $e_f(t) < 0$ in G_f, where the length of each arc is defined to be its reduced cost. It then attempts to send Δ units of flow from k to t along this shortest path. Since our algorithm works with the given capacitated network, there may be an arc (v,w) on the shortest path from k to t whose residual capacity $u_f(v,w)$ is less than Δ. In this case, Δ units of flow cannot be pushed over (v,w). To ensure that there are at most $O(m)$ flow augmentations in each phase, our algorithm maintains for each arc (v,w), its *arc excess* $e_f(v,w)$, and uses and modifies node and arc excesses as follows. Instead of associating all of the excess of flow (positive imbalance) that accumulates at a node with that node and calling it node excess, some of this excess is associated with the arcs emanating from that node and is called arc excess. In particular, the node imbalance and arc excess at node v satisfy

$$e_f(v) + \sum_{(v,w)\in E} e_f(v,w) = b(v) + \sum_{(w,v)\in E} f(w,v) - \sum_{(v,w)\in E} f(v,w), \tag{3}$$

rather than (1), which is only true when all arc excesses at node v are zero.

Consider a node v with an excess $e_f(v) \geqslant \Delta$. Let (v,w) be the arc incident to node v on the shortest path in G_f from v to t. First, Δ units of excess are transferred from node v (i.e., from $e_f(v)$) to the tail of arc (v,w) (i.e., to $e_f(v,w)$). Next, as much of this arc excess as possible is sent along arc (v,w) to the head of (v,w). This amount is clearly $\delta = \min\{e_f(v,w), u_f(v,w)\}$, and results in the flow in arc (v,w) and the excess at the head of (v,w), which is equal to the excess $e_f(w,v)$ at the tail of the reverse arc (w,v), being increased by δ and $e_f(v,w)$ being decreased by δ. Finally, as much of the excess $e_f(w,v)$, but no more than Δ, is transferred to node w (i.e., to $e_f(w)$. If $e_f(w)$ now is at least Δ, then the above pseudoflow augmentation procedure continues from node w; otherwise the augmentation is terminated at node w.

At the beginning of each phase, all arc excesses are set to zero. At the end of each phase, all arc excesses are transferred to node excesses by setting, for each $v \in V$, $e_f(v) \leftarrow e_f(v) + \sum_{(v,w)\in E} e_f(v,w)$. At the beginning of each phase, we choose the scaling factor to be $\Delta = \max\{1, \lfloor \sum_{v\in V: e_f(v)>0} e_f(v)/(2(m+n)) \rfloor\}$. A Δ-phase ends when $e_f(v) < \Delta$ for all $v \in V$.

```
**Algorithm**
**begin**
    for each $(v,w) \in E$ **if** $c(v,w) < 0$, **then** set $f(v,w) \leftarrow u(v,w)$ **else** set $f(v,w) \leftarrow 0$;
    compute $e_f(v)$, $\forall v \in V$ and set $e_f(v,w) \leftarrow 0$, $\forall (v,w) \in E$, and $p(v) \leftarrow 0$, $\forall v \in V$;
    **while** there is any node with excess **do begin** $\{\Delta$-scaling phase$\}$
        set $\Delta \leftarrow \max\{1, \lfloor \sum_{v \in V: e_f(v)>0} e_f(v)/(2(m+n)) \rfloor\}$;
        set $D \leftarrow \{v \in V\colon e_f(v) \geqslant \Delta\}$.
        **while** $D \neq \emptyset$ **do begin**
            choose a node $k \in D$;
            compute the shortest paths and shortest path distances $d(v)$ in $G_f$ from node $k$
            to all nodes $v \in V_f(k)$, using the reduced costs $c_p(v,w)$ as arc lengths;
            set $p(v) \leftarrow p(v) + d(v)$ for all nodes $v \in V_f(k)$;
            choose a node $t \in V_f(k)$ with $e_f(t) < 0$;
            **if** no such node $t$ exists **then** STOP (there is no feasible solution)
            **else** set $v \leftarrow k$;
            **while** $v \neq t$ and $e_f(v) \geqslant \Delta$ **do begin**
                let $w$ be the node succeding node $v$ on the shortest path from $k$ to $t$;
                set $e_f(v) \leftarrow e_f(v) - \Delta$, $e_f(v,w) \leftarrow e_f(v,w) + \Delta$;
                compute $\delta = \min\{u_f(v,w), e_f(v,w)\}$;
                set $f(v,w) \leftarrow f(v,w) + \delta$, $f(w,v) \leftarrow -f(v,w)$,
                set $e_f(v,w) \leftarrow e_f(v,w) - \delta$, $e_f(w,v) \leftarrow e_f(w,v) + \delta$;
                compute $\beta = \min\{e_f(w,v), \Delta\}$;
                set $e_f(w,v) \leftarrow e_f(w,v) - \beta$, $e_f(w) \leftarrow e_f(w) + \beta$;
                set $v \leftarrow w$;
            **end**;
            reconstruct $D$;
        **end**;
        set $e_f(v) \leftarrow e_f(v) + \sum_{(v,w) \in E} e_f(v,w)$, for each $v \in V$;
        set $e_f(v,w) \leftarrow 0$, for all $(v,w) \in E$;
    **end**;
**end**.
```

Lemma 2. *In each Δ-scaling phase of the algorithm, $0 \leqslant e_f(v,w) + e_f(w,v) < \Delta$ and $e_f(v,w) < u_f(v,w)$ if $u_f(v,w) > 0$.*

Proof. We prove this by induction. The lemma is true if no flow has been pushed from v to w or w to v since $e_f(v,w) = e_f(w,v) = 0$. Assume that the lemma is true at some point before flow is pushed from v to w or w to v. We will prove that the lemma continues to hold after flow is pushed from v to w (the case that flow is pushed from w to v can be proved similiarly).

Let us distinguish quantities after the push from those before the push by an overbar. Note that after the push, $\bar{u}_f(w,v) \geqslant \delta$. We have the following two cases.

(i) $\delta < u_f(v,w)$. In this case, $\bar{e}_f(v,w) = 0$, $\bar{u}_f(v,w) > 0$ and $\bar{e}_f(v,w) < \bar{u}_f(v,w)$. If $\beta < \Delta$, then $\bar{e}_f(w,v) = 0$ and the lemma holds. If $\beta = \Delta$, then $\bar{e}_f(v,w) + \bar{e}_f(w,v) = e_f(v,w) + \Delta - \delta + e_f(w,v) + \delta - \beta = e_f(v,w) + e_f(w,v) < \Delta$ and $\bar{e}_f(w,v) = e_f(w,v) + \delta - \Delta < \delta \leqslant \bar{u}_f(w,v)$ since $e_f(w,v) < \Delta$. The lemma holds.

(ii) If $\delta = u_f(v,w)$, then $\bar{e}_f(v,w) = e_f(v,w) + \Delta - u_f(v,w) < \Delta$ since $e_f(v,w) < u_f(v,w)$. Moreover, $(v,w) \notin G_{\bar{f}}$ since $\bar{u}_f(v,w) = 0$. If $\beta < \Delta$, then $\bar{e}_f(w,v) = 0$ and the lemma holds. If $\beta = \Delta$, then $\bar{e}_f(v,w) + \bar{e}_f(w,v) = e_f(v,w) +$

$\Delta-\delta+e_f(w,v)+\delta-\beta=e_f(v,w)+e_f(w,v)<\Delta$ and $\bar{e}_f(w,v)=e_f(w,v)+\delta-\Delta<\delta\leqslant\bar{u}_f(w,v)$ since $e_f(w,v)<\Delta$. The lemma holds.

Hence, if the statements of the lemma hold immediately before flow is pushed over an arc, they hold immediately after as well. □

Lemma 3. *There are at least one and at most* $4(n+m)$ *flow augmentations in each scaling phase.*

Proof. Consider the potenial function $F=\sum_{v\in V:e_f(v)>0}\lfloor e_f(v)/\Delta\rfloor$. At the beginning of a Δ-phase, $F\leqslant \sum_{v\in V:e_f(v)>0}e_f(v)/\Delta\leqslant 4(m+n)$ since $\Delta=\max\{1,\lfloor\sum_{v\in V:e_f(v)>0}e_f(v)/(2(m+n))\rfloor\}$. During an augmentation step, when flow is pushed from node v with $e_f(v)\geqslant\Delta$ to node w, F does not increase since $\beta\leqslant\Delta$. On the last push along an arc (v,w) in an augmentation step, F is reduced by at least 1 since $e_f(v)$ is reduced by Δ and $\bar{e}_f(w)<\Delta$ after the push. Therefore, F decreases by at least 1 on every augmentation step, and hence, there are at most $4(m+n)$ flow augmentations in a scaling phase. Since, at the start of a scaling phase, there is at least one node with $e_f(v)\geqslant\Delta$, the lemma follows. □

Lemma 2 implies that at the end of each Δ-scaling phase, the total excess is at most $(n+m)\Delta$. Since the scaling factor is set to $\lfloor\sum_{v\in V:e_f(v)>0}e_f(v)/(2(n+m))\rfloor$ at the beginning of each phase except possibly the last, the scaling factor is reduced by at least a factor of 2 in each phase except possibly the last. Since the initial scaling factor is $\Delta\leqslant\lfloor B/(2(m+n))\rfloor$, after at most $\log(B/(2(m+n)))$ scaling phases, $\Delta\leqslant 1$, and an optimal solution is found since all $e_f(v)$ are integers. Consequently, the number of scaling phases is bounded by $O(\log(B/(2(m+n))))$. Each flow augmentation is dominated by the computation of a shortest-path tree which can be done is $O(m+n\log n)$ time using Fredman and Tarjan's [8] implementation of Dijkstra's [6] algorithm. Thus we have the following.

Theorem 1. *The algorithm solves the minimum cost network flow problem in* $O(m(m+n\log n)\log(B/(2(m+n))))$ *time.*

In Orlin's excess scaling algorithm, it is possible that in some scaling phase there are no flow augmentations, especially when the total excess is relatively small compared with the scaling factor. Our algorithm, on the other hand, performs as indicated by Lemma 3, at least one flow augmentation in each scaling phase. It may also reduce the Δ by a factor of more than 2 from one phase to the next.

The ideas underlying our algorithm, and in particular the use of arc excess, can be adapted to several capacity and excess scaling based algorithms for capacitated network flow problems, such as those proposed by Gabow [10], Ahuja et al. [1], Ahuja and Orlin [3].

References

[1] R.K. Ahuja, A.V. Goldberg, J.B. Orlin, R.E. Tarjan, Finding minimum cost flows by double scaing, Math. Programm. 53 (1992) 243–266.

[2] R.K. Ahuja, T.L. Magnanti, J.B. Orlin, Network Flows: Theory, Algorithms, and Applications, Prentice-Hall, Englewood Cliffs, NJ, 1993.

[3] R.K. Ahuja, J.B. Orlin, A capacity scaling algorithm for the constrained maximum flow problem, Networks 25 (1995) 89–98.

[4] R.G. Bland, J. Cheriyan, D.L. Jensen, L. Ladanyi, An Empirical Study of Min Cost Flow Algorithms, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Vol. 12, 1993, pp. 119–156.

[5] R.G. Bland, D.L. Jensen, On the computational behavior of a polynomial-time network flow algorithm, Math. Programm. 54 (1992) 1–39.

[6] E. Dijkstra, A note on two problems in connexion with graphs, Numer. Math. 1 (1959) 269–271.

[7] J. Edmonds, R.M. Karp, Theoretical improvements in algorithmic efficiency for network flow problems, J. ACM 19 (1972) 248–264.

[8] M.L. Fredman, R.E. Tarjan, Fibonacci heaps and their uses in improved network optimization algorithms, J. ACM 34 (1987) 596–615.

[9] S. Fujishige, K. Iwano, J. Nakano, S. Tezuka, A Speculative Contraction Method for Minimum Cost Flows: Toward a Practical Algorithm, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Vol. 12, 1993, pp. 219–245.

[10] H.N. Gabow, Scaling algorithms for network problems, J. Comput. System Sci. 31 (1985) 148–168.

[11] A.V. Goldberg, M. Kharitonov, On Implementing Scaling Push-Relabel Algorithms for the Minimum-Cost Flow Problem, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Vol. 12, 1993, pp. 157–198.

[12] A.V. Goldberg, R.E. Tarjan, Solving minimum cost flow problems by successive approximation, Math. Oper. Res. 15 (1990) 430–466.

[13] D. Goldfarb, Z. Jin, A faster combinatorial algorithm for the generalized circulation problem, Math. Oper. Res. 21 (1996) 529–539.

[14] D. Goldfarb, Z. Jin, Strongly polynomial excess scaling and dual simplex algorithms for the minimum cost network flow problem, Technical Report, IEOR Department, Columbia University, New York, NY, 1996.

[15] D. Goldfarb, Z. Jin, J.B. Orlin, Polynomial-time highest-gain augmenting path algorithms for the generalized circulation problem, Math. Oper. Res. 22 (1997) 793–802.

[16] Y. Ikura, G.L. Nemhauser, Computational experience with a polynomial-time dual simplex algorithm for the transportation problem, Discrete Appl. Math. 13 (1986) 239–248.

[17] J.B. Orlin, A faster strongly polynomial minimum cost flow algorithm, Oper. Res. 41 (1993) 338–350.

[18] J.B. Orlin, S.A. Plotkin, E. Tardos, Polynomial dual network simplex algorithms, Math. Programm. 60 (1993) 255–276.

[19] H. Rock, Scaling techniques for minimal cost network flows, in: V. Page (Ed.), Discrete Structures and Algorithms, Carl Hansen, Munich, 1980, pp. 181–191.

[20] E. Tardos, A strongly polynomial minimum cost circulation algorithm, Combinatorica 5 (1985) 247–255.