# AN $\epsilon$–RELAXATION ALGORITHM FOR CONVEX NETWORK FLOW PROBLEMS*

RENATO DE LEONE†, ROBERT R. MEYER‡, AND ARMAND ZAKARIAN‡

**Abstract.** A relaxation method for separable convex network flow problems is developed that is well-suited for problems with large variations in the magnitude of the nonlinear cost terms. The arcs are partitioned into two sets, one of which contains only arcs corresponding to strictly convex costs. The algorithm adjusts flows on the other arcs whenever possible, and terminates with primal-dual pairs that satisfy complementary slackness on the strictly convex arc set and $\epsilon$-complementary slackness on the remaining arcs. An asynchronous parallel variant of the method is also developed. Computational results demonstrate that the method is significantly more efficient on ill-conditioned networks than existing methods, solving problems with several thousand nonlinear arcs in one second or less.

**1. Introduction.** Given a directed graph $\mathcal{G}(\mathcal{N}, \mathcal{A})$ with $n$ nodes and $m$ arcs, the minimum–cost network flow problem that we consider is:

$$\begin{array}{lll} & \text{minimize} & \displaystyle\sum_{(i,j)\in\mathcal{A}} f_{ij}(x_{ij}) \\ (1) & & \\ & \text{subject to} & \displaystyle\sum_{j\in\delta_+(i)} x_{ij} - \sum_{j\in\delta_-(i)} x_{ji} = b_i, \quad i \in \mathcal{N} \end{array}$$

where for each node $i$, $\delta_+(i)$ is the subset of nodes $j$ such that $(i,j) \in \mathcal{A}$ and $\delta_-(i)$ is the subset of nodes $j$ such that $(j,i) \in \mathcal{A}$.

Networks with nonlinear objectives arise in a variety of applications including electrical networks (Hu 1966), pipeline networks (Collins, Cooper, Helgason, Kennington & LeBlanc 1978), urban traffic flow (Magnanti 1984) and communications networks (Monma & Segal 1982).

We make the following assumptions with regard to the functions $f_{ij}$.

ASSUMPTION 1.1. *Each $f_{ij}$ is a closed proper convex function and the intersection of* $\operatorname{dom}\left(\sum_{(i,j)\in\mathcal{A}} f_{ij}\right)$ *with the set of vectors satisfying the constraints of (1) is nonempty.*

ASSUMPTION 1.2. *The conjugate functions*

$$f_{ij}^*(t_{ij}) := \sup_{x_{ij}} \{t_{ij}x_{ij} - f_{ij}(x_{ij})\}$$

*are real-valued.*

Assumption 1.1 guarantees that (1) is feasible. Assumption 1.2 implies that the objective function of (1) has bounded level sets and together with Assumption 1.1 guarantees that an optimal solution exists. A linear cost function must have a bounded domain in order to satisfy Assumption 1.1.

* This material is based on research supported by the Air Force Office of Scientific Research Grants AFOSR-89-0410 and F49620-94-1-0036, and by NSF Grants CCR-8907671, CDA-9024618 and CCR-9306807.

† Dipartimento di Matematica e Fisica, Universitá di Camerino, Via Madonna delle Carceri, 62032 Camerino (MC), Italy

‡ Center for Parallel Optimization, Computer Sciences Department, University of Wisconsin—Madison, 1210 W. Dayton St., Madison, WI 53706.

Often the functions $f_{ij}$ imply flow capacity constraints:

$$f_{ij}(x_{ij}) := \begin{cases} \bar{f}_{ij}(x_{ij}) & \text{if } l_{ij} \le x_{ij} \le u_{ij} \\ +\infty & \text{otherwise} \end{cases} \tag{2}$$

Here $\bar{f}_{ij}$ is a real-valued convex function and $l_{ij} \le u_{ij}$ are the (finite) lower and upper bounds on the flow on arc $(i,j) \in \mathcal{A}$. In this case Assumption 1.2 is trivially satisfied.

We denote the left and right derivative of $f_{ij}$ at $x_{ij} \in \mathrm{dom} f_{ij}$ by $f_{ij}^-(x_{ij})$ and $f_{ij}^+(x_{ij})$, respectively. For points $x_{ij}$ to the left of $\mathrm{dom} f_{ij}$ we set $f_{ij}^-(x_{ij}) = f_{ij}^+(x_{ij}) = -\infty$ and for points $x_{ij}$ to the right of $\mathrm{dom} f_{ij}$ we set $f_{ij}^-(x_{ij}) = f_{ij}^+(x_{ij}) = +\infty$. Theorem 24.1 in (Rockafellar 1970) states that $f_{ij}^-$ and $f_{ij}^+$ are nondecreasing functions on $\mathbf{R}$, finite on $\mathrm{int}\,\mathrm{dom} f_{ij}$ and

$$f_{ij}^+(z_{ij}) \le f_{ij}^-(x_{ij}) \le f_{ij}^+(x_{ij}) \le f_{ij}^-(y_{ij})$$

whenever $z_{ij} < x_{ij} < y_{ij}$.

We make an additional assumption regarding the functions $f_{ij}$:

ASSUMPTION 1.3. *$f_{ij}^-(x_{ij}) < +\infty$ and $f_{ij}^+(x_{ij}) > -\infty$ for all $x_{ij} \in \mathrm{dom} f_{ij}$.*

Assumption 1.3 guarantees that the feasible solutions of (1) are *regularly feasible* ((Rockafellar 1984), Section 8D). Together with the first two assumptions it also guarantees the existence of dual optimal solutions ((Rockafellar 1984), Section 8L).

We say that a pair $(x, \pi)$ satisfies complementary slackness (CS) on an arc $(i,j) \in \mathcal{A}$ if

$$f_{ij}^-(x_{ij}) \le \pi_i - \pi_j \le f_{ij}^+(x_{ij}) \tag{3}$$

For arbitrary vector $\pi$ and $f_{ij}$ strictly convex satisfying Assumption 1.2, there exists a unique $x_{ij}$ satisfying (3). A vector $x$ is optimal for (1) if it satisfies the flow conservation constraints and together with some $\pi$ satisfies CS on every arc in $\mathcal{A}$.

A pair $(x, \pi)$ satisfies $\epsilon$-complementary slackness ($\epsilon$-CS) on an arc $(i,j) \in \mathcal{A}$ if

$$f_{ij}^-(x_{ij}) - \epsilon \le \pi_i - \pi_j \le f_{ij}^+(x_{ij}) + \epsilon \tag{4}$$

We will see below that $\epsilon$-complementary slackness corresponds to near optimality for small $\epsilon$.

A statement of the basics of the algorithm is given in the following section. Section 3 provides a convergence analysis and a quantitative interpretation of the significance of $\epsilon$-complementary slackness. Some computational refinements of the basic method are presented in section 4, together with a specialization of the algorithm to the mixed linear-quadratic case. Section 5 presents numerical results and a comparison with other approaches. Finally, parallel versions of the algorithm are discussed in section 6.

**2. The basic algorithm .** We state the algorithm in its simplest form in this section. A brief discussion is given of the main ideas, followed by a formal statement of the method.

The algorithm computes an approximate solution of problem (1) in the sense that it generates a sequence of iterates converging to a feasible solution satisfying $\epsilon$-CS. More precisely, we assume that the set $\mathcal{A}$ has been partitioned into two sets $\mathcal{A}_0$ and $\mathcal{A}_1$ such that only arcs with strictly convex cost functions are in $\mathcal{A}_1$, and we show that in the limit the primal-dual pair produced by the algorithm will satisfy CS on each arc in $\mathcal{A}_1$ and $\epsilon$-CS on each arc in $\mathcal{A}_0$. This approach differs from that of (Bertsekas, Hosein & Tseng 1987) in that it divides the arcs into two sets and focuses on changing single node prices and flows on arcs incident to single nodes.

In **Step 1** below, a node $i$ with positive surplus is identified. Steps 2–4 are then repeatedly applied until the surplus at this node is driven to 0 (which is achievable in a finite number of steps, see discussion following algorithm). The algorithm first tries to "push" this surplus to adjacent nodes using only arcs in $\mathcal{A}_0$ (**Step 2** and **3**). If unsuccessful, it then tries to raise the price (**Step 4**) of node $i$ sufficiently high so that the surplus can be "pushed away" using the incident arcs in $\mathcal{A}_1$ but not so high as to violate $\epsilon$-CS on the arcs in $\mathcal{A}_0$. We assume that $\epsilon$ and $\epsilon'$ are two given numbers such that $0 \leq \epsilon' < \frac{\epsilon}{2}$.

### Algorithm 1

**Step 0** [Initialization] Start with $(x, \pi)$ satisfying $\epsilon$-CS on all arcs in $\mathcal{A}_0$ and CS on all arcs in $\mathcal{A}_1$.
Set $k = 0$.

**Step 1** [Choose node with positive surplus] Choose a node $i$ such that

$$s_i := b_i - \sum_{j \in \delta_+(i)} x_{ij} + \sum_{j \in \delta_-(i)} x_{ji} > 0$$

and increment $k$. If no such node exists, STOP.

**Step 2** [Increase flow on outgoing arc in $\mathcal{A}_0$] If there is an arc $(i, j) \in \mathcal{A}_0$ such that

$$f^+_{ij}(x_{ij}) - (\pi_i - \pi_j) \leq -\frac{\epsilon}{2}$$

then change the flow on arc $(i, j)$,

$$x_{ij} \leftarrow x_{ij} + \min\{s_i,\ \delta\}$$

where $\delta > 0$ is the largest flow increase allowing $\epsilon'$-CS to be satisfied on arc $(i, j)$:

$$f^-_{ij}(x_{ij} + \delta) - \epsilon' \leq \pi_i - \pi_j \leq f^+_{ij}(x_{ij} + \delta) + \epsilon'$$

The flow on the remaining arcs and the dual costs remain unchanged. Update $s_i$ and if its new value is 0 go to Step 1 else return to Step 2.

**Step 3** [Decrease flow on incoming arc in $\mathcal{A}_0$] If there is an arc $(j, i) \in \mathcal{A}_0$ such that

$$f^-_{ji}(x_{ji}) - (\pi_j - \pi_i) \geq \frac{\epsilon}{2}$$

then change the flow on arc $(j, i)$,

$$x_{ji} \leftarrow x_{ji} - \min\{s_i,\ \delta\}$$

where $\delta > 0$ is the least flow decrease allowing $\epsilon'$-CS to be satisfied on arc $(j, i)$:

$$f^-_{ji}(x_{ji} - \delta) - \epsilon' \leq \pi_j - \pi_i \leq f^+_{ji}(x_{ji} - \delta) + \epsilon'$$

The flow on the remaining arcs and the dual costs remain unchanged. Update $s_i$ and if its new value is 0 go to Step 1 else return to Step 3.

**Step 4** [Raise price and change flows in $\mathcal{A}_1$] Compute

$$\Delta_1 := \min_{\{j\in\delta_+(i)\,|\,(i,j)\in\mathcal{A}_0\}} \left\{ f_{ij}^+(x_{ij}) - (\pi_i - \pi_j) \right\} \geq -\frac{\epsilon}{2}$$

$$\Delta_2 := \min_{\{j\in\delta_-(i)\,|\,(j,i)\in\mathcal{A}_0\}} \left\{ -\left( f_{ji}^-(x_{ji}) - (\pi_j - \pi_i) \right) \right\} \geq -\frac{\epsilon}{2}$$

For each $\gamma$ and for each arc $(i,j)\in\mathcal{A}_1$ let $x_{ij}(\gamma)$ be the flow such that

$$f_{ij}^-(x_{ij}(\gamma)) \leq \pi_i + \gamma - \pi_j \leq f_{ij}^+(x_{ij}(\gamma))$$

Similarly for each arc $(j,i)\in\mathcal{A}_1$ let $x_{ji}(\gamma)$ be the flow such that

$$f_{ji}^-(x_{ji}(\gamma)) \leq \pi_j - \pi_i - \gamma \leq f_{ji}^+(x_{ji}(\gamma))$$

and let $\Delta_3 > 0$ be the value of $\gamma$ such that

$$b_i - \sum_{(i,j)\in\mathcal{A}_1} x_{ij}(\gamma) - \sum_{(i,j)\in\mathcal{A}_0} x_{ij} + \sum_{(j,i)\in\mathcal{A}_1} x_{ji}(\gamma) + \sum_{(j,i)\in\mathcal{A}_0} x_{ji} = 0$$

(where $\Delta_3 = +\infty$ if a solution does not exist).
Define:

$$\Delta := \min\{\Delta_1 + \epsilon, \Delta_2 + \epsilon, \Delta_3\}$$

Change the price on node $i$:

$$\pi_i \leftarrow \pi_i + \Delta$$

and adjust the flow on the incident arcs in $\mathcal{A}_1$ so that CS is satisfied. The flow on all other arcs and the prices of all remaining nodes remains unchanged. Update $s_i$ and if its new value is 0 go to Step 1 else return to Step 2.

We now verify that **Step 2** maintains $\epsilon$-CS on arc $(i,j)$ (this can be done similarly for **Step 3**). Let $x'_{ij}$ be the updated value of the flow on $(i,j)$. We have

$$f_{ij}^-(x'_{ij}) - \epsilon \leq f_{ij}^-(x_{ij}+\delta) - \epsilon \leq f_{ij}^-(x_{ij}+\delta) - \epsilon' \leq \pi_i - \pi_j$$

and

$$f_{ij}^+(x'_{ij}) + \epsilon \geq f_{ij}^+(x_{ij}) + \epsilon \geq \pi_i - \pi_j$$

To see that $\Delta > 0$ in **Step 4**, note that $\Delta_1 > -\frac{\epsilon}{2}$ and $\Delta_2 > -\frac{\epsilon}{2}$ since no eligible arcs could be found in **Step 2** and **Step 3**, respectively. Also, $\Delta_3 > 0$ since we want $x_{ij}(\Delta_3) > x_{ij}(0)$ for $(i,j)\in\mathcal{A}_1$ and $x_{ji}(\Delta_3) < x_{ji}(0)$ for $(j,i)\in\mathcal{A}_1$.

We now verify that the price increase in **Step 4** doesn't destroy $\epsilon$-CS on the arcs in $\mathcal{A}_0$. Denote the new value of $\pi_i$ by $\pi'_i$. We have

$$\pi'_i - \pi_j > \pi_i - \pi_j \geq f_{ij}^-(x_{ij}) - \epsilon$$

and

$$\pi_i' - \pi_j \le \pi_i + \Delta_1 + \epsilon - \pi_j = \min_{\{v\in\delta_+(i)\,|\,(i,v)\in\mathcal{A}_0\}} \left\{ f_{iv}^+(x_{iv}) \right\} + \epsilon \le f_{ij}^+(x_{ij}) + \epsilon$$

Similarly, $\epsilon$-CS is preserved on an arc $(j,i) \in \mathcal{A}_0$.

Finally, we show that Steps 2–4 eventually drive the surplus of node $i$ to 0. Assume the opposite. Then $\Delta_3 = +\infty$ in every execution of Step 4 implying that this step increases $\pi_i$ by at least $\frac{\epsilon}{2}$. Hence $\pi_i$ approaches infinity which is impossible as will be shown in the proof of Lemma 3.1 in a more general context.

**3. Convergence analysis.** In this section we establish convergence for **Algorithm 1**.

We refer to the pair $(x,\pi)$ available at the beginning of **Step 1** (just before $k$ is incremented) as $(x^k,\pi^k)$.

The surplus of node $i \in \mathcal{N}$ at the beginning of iteration $k$ is given by

$$s_i^k := b_i - \sum_{j\in\delta_+(i)} x_{ij}^k + \sum_{j\in\delta_-(i)} x_{ji}^k \tag{5}$$

LEMMA 3.1. *The sequence $\{\pi^k\}_{k=0}^\infty$ generated by **Algorithm 1** converges to some point $\pi^*$.*

*Proof.* The sequence $\{\pi_i^k\}_{k=0}^\infty$ is nondecreasing for all $i \in \mathcal{N}$. Hence we only need to show it is bounded above. Assume the contrary, i.e. $\pi_i^k \to \infty$ for all $i \in \mathcal{N}_1$ while $\pi_j^k$ stay bounded for $j \in \mathcal{N}_0$ as $k \to \infty$. Here $\{\mathcal{N}_0, \mathcal{N}_1\}$ is a partition of $\mathcal{N}$ with $\mathcal{N}_0 \neq \emptyset$ since there is always a node with negative surplus which is never chosen in **Step 1**. Pick an arc $(i,j)$ with $i \in \mathcal{N}_1$ and $j \in \mathcal{N}_0$. We have that $\pi_i^k - \pi_j^k \to \infty$ which together with the $\epsilon$-CS condition

$$f_{ij}^-(x_{ij}^k) - \epsilon \le \pi_i^k - \pi_j^k \le f_{ij}^+(x_{ij}^k) + \epsilon$$

implies that $f_{ij}^+(x_{ij}^k) \to +\infty$. Hence for any $\bar{x}_{ij} \in \operatorname{int}\operatorname{dom} f_{ij}$ there exist an iteration number $k_{ij}$ such that $x_{ij}^k \ge \bar{x}_{ij}$ whenever $k \ge k_{ij}$. Also, if $\bar{x}_{ij} \in \operatorname{dom} f_{ij}$ is the right endpoint of $\operatorname{dom} f_{ij}$ then $f_{ij}^-(\bar{x}_{ij}) < +\infty$ (Assumption 1.3) hence $f_{ij}^+(x)$ is uniformly bounded above on $\operatorname{int}\operatorname{dom} f_{ij}$ implying that $x_{ij}^k = \bar{x}_{ij}$ for $k \ge k_{ij}$.

Similarly if $(j,i)$ is an arc with $j \in \mathcal{N}_0$, $i \in \mathcal{N}_1$ and $\bar{x}_{ji} \in \operatorname{dom} f_{ji}$ there exist an iteration number $k_{ji}$ such that $x_{ji}^k \le \bar{x}_{ji}$ whenever $k \ge k_{ji}$.

Now choose for each arc $(i,j)$, $i \in \mathcal{N}_1$, $j \in \mathcal{N}_0$ the number $\bar{x}_{ij} \in \operatorname{dom} f_{ij}$ and for each arc $(j,i)$, $j \in \mathcal{N}_0$, $i \in \mathcal{N}_1$ the number $\bar{x}_{ji} \in \operatorname{dom} f_{ji}$ such that

$$\sum_{i\in\mathcal{N}_1} b_i \quad - \sum_{\{(i,j)|i\in\mathcal{N}_1, j\in\mathcal{N}_0\}} \bar{x}_{ij} + \sum_{\{(j,i)|i\in\mathcal{N}_1, j\in\mathcal{N}_0\}} \bar{x}_{ji} \quad \le \quad 0$$

(This is always possible since the problem is feasible.) For sufficiently large $k$ we now have

$$\begin{aligned} \sum_{i\in\mathcal{N}_1} s_i^k \quad &= \quad \sum_{i\in\mathcal{N}_1} b_i - \sum_{\{(i,j)|i\in\mathcal{N}_1, j\in\mathcal{N}_0\}} x_{ij}^k + \sum_{\{(j,i)|i\in\mathcal{N}_1, j\in\mathcal{N}_0\}} x_{ji}^k \quad \le \\ & \quad\quad \sum_{i\in\mathcal{N}_1} b_i - \sum_{\{(i,j)|i\in\mathcal{N}_1, j\in\mathcal{N}_0\}} \bar{x}_{ij} + \sum_{\{(j,i)|i\in\mathcal{N}_1, j\in\mathcal{N}_0\}} \bar{x}_{ji} \quad \le \quad 0 \end{aligned}$$

which contradicts the assumption that the nodes in $\mathcal{N}_1$ are chosen infinitely many times in **Step 1**. ∎

LEMMA 3.2. *The sequence $\{x^k\}_{k=0}^{\infty}$ generated by **Algorithm 1** converges to some $x^* \in \mathrm{dom}\left(\sum_{(i,j)\in\mathcal{A}} f_{ij}\right)$.*

*Proof.* We start by showing that the sequence $\{x^k\}_{k=0}^{\infty}$ is bounded. First note that the quantity $\sum_{i\in\mathcal{N}} |s_i^k|$ is a nonincreasing function of $k$. Assume that the flow on an arc $(i,j) \in \mathcal{A}$ is unbounded (i.e. $x_{ij}^{k_l} \to \infty$ where $\{k_l\}$ is an appropriately chosen index sequence). Hence there is a cycle $Y$ containing $(i,j)$ such that the flows on the forward arcs of $Y$ (denoted by $Y^+$) approach $+\infty$ while the flows on the backward arcs of $Y$ (denoted by $Y^-$) approach $-\infty$. We assume that $\{k_l\}$ has been further thinned so that

$$\begin{aligned} x_{uw}^{k_l} &\to +\infty, \quad (u,w) \in Y^+ \\ x_{wu}^{k_l} &\to -\infty, \quad (w,u) \in Y^- \end{aligned}$$

By summing the $\epsilon$-CS condition

$$f_{uw}^-(x_{uw}^{k_l}) - \epsilon \le \pi_u^{k_l} - \pi_w^{k_l} \le f_{uw}^+(x_{uw}^{k_l}) + \epsilon$$

along Y we get that

$$\sum_{(u,w)\in Y^+} f_{uw}^-(x_{uw}^{k_l}) \quad - \sum_{(w,u)\in Y^-} f_{wu}^+(x_{wu}^{k_l}) \le n\epsilon$$

which is a contradiction with Assumption 1.2 since $x_{uw}^{k_l} \to +\infty$ implies $f_{uw}^-(x_{uw}^{k_l}) \to +\infty$ while $x_{wu}^{k_l} \to -\infty$ implies $f_{wu}^+(x_{wu}^{k_l}) \to -\infty$.

We now show that $\{x_{ij}^k\}_{k=0}^{\infty}$ converges for each $(i,j) \in \mathcal{A}_1$. Since $f_{ij}$ is proper closed strictly convex function, its conjugate $f_{ij}^*$ is essentially smooth hence (using Assumption 1.2) (continuously) differentiable. We have

$$x_{ij}^k = \nabla f_{ij}^*(\pi_i^k - \pi_j^k)$$

Lemma 3.1 shows that $\pi_i^k - \pi_j^k \to \pi_i^* - \pi_j^*$ hence $\{x_{ij}^k\}$ converges.

To see that $\{x_{ij}^k\}_{k=0}^{\infty}$ converges for $(i,j) \in \mathcal{A}_0$, note that for sufficiently large $k$ $(i,j)$ can be chosen only in **Step 2** or only in **Step 3** but not in both of them. Hence starting with some $k$, $\{x_{ij}^k\}$ is monotone which together with its boundedness implies convergence.

Finally, $x_{ij}^* \in \mathrm{dom} f_{ij}$ for each $(i,j) \in \mathcal{A}$ because $(x^*, \pi^*)$ satisfy $\epsilon$-CS implying that $f_{ij}^-(x_{ij}^*)$ and $f_{ij}^+(x_{ij}^*)$ cannot both be $+\infty$ or $-\infty$. ∎

The following assumption regarding the way nodes are chosen in **Step 1** is needed in establishing convergence to a feasible point.

ASSUMPTION 3.1. *If $s_i^k > 0$ for some $k$ then node $i$ is chosen in **Step 1** in some iteration numbered higher than $k$.*

LEMMA 3.3. *The sequence $\{s^k\}_{k=0}^{\infty}$ generated by **Algorithm 1** converges to* 0.

*Proof.* Let $i \in \mathcal{N}$ be a node that is chosen infinitely many times in **Step 1** and let $\{k_l\}$ be the corresponding sequence of iteration numbers. We have

$$0 = s_i^{k_l+1} \le s_i^{k_l+2} \le \ldots \le s_i^{k_{l+1}}$$

and so

$$\limsup_{k\to\infty} s_i^k = \limsup_{l\to\infty} s_i^{k_l} \le \limsup_{l\to\infty} \|x^{k_l} - x^{k_l+1}\|_1 = 0$$

(using that the change in a node surplus doesn't exceed the sum of changes of arc flows). Hence $s_i^k \to 0$.

On the other hand, if $j \in \mathcal{N}$ is a node with $s_j^k \le 0$ for all $k$ we have

$$s_j^k \ge \sum_{\{u\in\mathcal{N} \,|\, s_u^k<0\}} s_u^k \quad = \quad - \sum_{\{u\in\mathcal{N} \,|\, s_u^k\ge 0\}} s_u^k \quad \to \quad 0$$

hence $s_j^k \to 0$. ∎

The preceding three lemmas constitute the proof of the first part of the main convergence theorem. The second part is concerned with the case when $\mathcal{A}_0 = \mathcal{A}$. In this case we define the *admissible network* w.r.t. some pair $(x, \pi)$ to be $\mathcal{G}(\mathcal{N}, \hat{\mathcal{A}})$ where $(i, j) \in \hat{\mathcal{A}}$ iff

$$\begin{aligned}
&(i,j) \in \mathcal{A},\ \ f_{ij}^+(x_{ij}) - (\pi_i - \pi_j) \le -\tfrac{\epsilon}{2} \\
&\text{or} \\
&(j,i) \in \mathcal{A},\ \ f_{ji}^-(x_{ji}) - (\pi_j - \pi_i) \ge \tfrac{\epsilon}{2}
\end{aligned}$$

Less formally, the admissible network contains an arc $(i, j)$ if flow can be pushed from node $i$ to node $j$ according to the rules of the algorithm.

ASSUMPTION 3.2. *$(x^0, \pi^0)$ are chosen so that the admissible network w.r.t. $(x^0, \pi^0)$ is acyclic.*

The above assumption is trivially satisfied if $(x^0, \pi^0)$ satisfy CS, for then the set of arcs of the admissible network is empty.

THEOREM 3.4. *The sequence $\{(x^k, \pi^k)\}_{k=0}^{\infty}$ generated by **Algorithm 1** converges to some $(x^*, \pi^*)$ such that $x^*$ is feasible for (1) and together with $\pi^*$ satisfies $\epsilon$-CS. Furthermore, if $\mathcal{A}_1 = \emptyset$ the algorithm terminates finitely with a primal–dual pair satisfying the same properties.*

*Proof.* The first part follows directly from Lemmas 3.1–3.3. For the second part we first show that if the admissible network is acyclic w.r.t. the initial choice of $(x, \pi)$, it remains acyclic throughout the algorithm. Indeed, **Steps 2** and **3** obviously cannot add new arcs to the admissible network. **Step 4** increases $\pi_i$ by more than $\frac{\epsilon}{2}$ and so may add some arcs originating at $i$ to the admissible network. At the same time all arcs entering $i$ leave the admissible network and so no cycle can be formed in it.

Now assume that the algorithm doesn't terminate. Since $\pi_i^k \to \pi_i^*$ and $\Delta > \frac{\epsilon}{2}$ in **Step 4**, we see that for sufficiently large $k$, $\pi_i^k = \pi_i^*$ for each $i \in \mathcal{N}$. Hence from some point on no new arcs are added to the admissible network. This together with its acyclicity guarantees that **Algorithm 1** terminates finitely. ∎

The pair $(x^*, \pi^*)$ computed by **Algorithm 1** is nearly optimal. The following two lemmas (Corollary 3.1 and Proposition 3.6 from (Bertsekas et al. 1987), respectively) give the precise statement. In them, $f(x)$ and $q(\pi)$ are used to denote the primal and dual objective values of (1):

$$\begin{aligned}
f(x) &:= \sum_{(i,j)\in\mathcal{A}} f_{ij}(x_{ij}), \\
q(\pi) &:= \pi^T b \ - \sum_{(i,j)\in\mathcal{A}} f_{ij}^*(\pi_i - \pi_j).
\end{aligned}$$

LEMMA 3.5. *Let $(x(\epsilon), \pi(\epsilon))$ satisfy $\epsilon$-CS and let $x(\epsilon)$ satisfy the flow conservation constraints. Then $f(x(\epsilon)) - q(\pi(\epsilon)) \to 0$ as $\epsilon \to 0$.*

LEMMA 3.6. *Let $(x(\epsilon), \pi(\epsilon))$ satisfy the assumptions of Lemma 3.5 and in addition assume that each $f_{ij}$ is of the form (2) (i.e. finite capacity constraints). Then*

$$0 \leq f(x(\epsilon)) - q(\pi(\epsilon)) \quad \leq \quad \epsilon \sum_{(i,j)\in\mathcal{A}} (u_{ij} - l_{ij})$$

In the case the $f_{ij}$ satisfy for some $d_{ij} > 0$ the inverse Lipschitz condition

$$d_{ij}(x_{ij}^1 - x_{ij}^2) \leq f_{ij}^-(x_{ij}^1) - f_{ij}^+(x_{ij}^2) \qquad \text{for} \;\; x_{ij}^1 > x_{ij}^2$$

(for example, $f_{ij}$ are piecewise quadratic) we can derive a bound on the distance from the point computed by **Algorithm 1** to the optimal solution set of (1).

LEMMA 3.7. *Let $(x^*, \pi^*)$ satisfy $\epsilon$-CS and let $x^*$ be feasible for (1). Let $\bar{x}$ be any optimal solution of (1). Then for every $(i,j) \in \mathcal{A}$*

$$d_{ij}|x_{ij}^* - \bar{x}_{ij}| \leq n\epsilon$$

*Proof.* Fix $(i,j) \in \mathcal{A}$ and assume that $x_{ij}^* \neq \bar{x}_{ij}$. According to the Conformal Realization Theorem (Bertsekas 1991) $x^* - \bar{x}$ can be decomposed into a sum of conforming simple cycle flows. Let $Y$ be any of the cycles containing $(i,j)$. By summing the $\epsilon$-CS condition (4) along $Y$ (as in the proof of Lemma 3.2) we obtain

$$\sum_{(u,w)\in Y^+} f_{uw}^-(x_{uw}^*) \quad - \sum_{(w,u)\in Y^-} f_{wu}^+(x_{wu}^*) \leq n\epsilon$$

Similarly by summing the CS condition (3) along $Y$ we get

$$\sum_{(u,w)\in Y^+} f_{uw}^+(\bar{x}_{uw}) \quad - \sum_{(w,u)\in Y^-} f_{wu}^-(\bar{x}_{wu}) \geq 0$$

Now by subtracting the above two inequalities we get

$$(6) \qquad \sum_{(u,w)\in Y^+} \left\{ f_{uw}^-(x_{uw}^*) - f_{uw}^+(\bar{x}_{uw}) \right\} \quad - \sum_{(w,u)\in Y^-} \left\{ f_{wu}^+(x_{wu}^*) - f_{wu}^-(\bar{x}_{wu}) \right\} \leq n\epsilon$$

For an arc $(u,w) \in Y^+$ we have $\bar{x}_{uw} < x_{uw}^*$ while for an arc $(w,u) \in Y^-$ we have $x_{wu}^* < \bar{x}_{wu}$ and so the conclusion of the Lemma follows immediately. □

**4. Computational considerations.** In this section we discuss some issues related to the implementation of **Algorithm 1** and some extensions to it.

One way of enforcing Assumption 3.1 in practice is to consider the nodes in **Step 1** in some fixed order that includes all nodes in $\mathcal{N}$. A more economical solution is used in the linear code $\epsilon$-RELAX (Bertsekas 1991). There the nodes that have positive surplus are kept in a queue. Nodes are "selected" from the front of the queue; if during the course of an iteration the surplus of an adjacent node (not already in the queue) becomes positive, that node is added to the end of the queue.

**Algorithm 1**, as described, consists only of "up" iterations (i.e. a node with a positive surplus is selected and its price may be raised). It is possible to define in a symmetrical way a "down" iteration which starts by selecting a node with negative surplus and may decrease its price. (Bertsekas & Tsitsiklis 1989) has an example of the $\epsilon$-relaxation algorithm cycling if "up" and "down" iterations are mixed arbitrarily. If we assume, however, that either the number of "up" iterations or the number of "down" iterations started at any given node is finite, cycling cannot occur. Indeed, this assumption implies that from some point in the course of the algorithm on, either only "up" or only "down" iterations are executed at any given node and our convergence proof can be used with slight modifications. The computational results in Section 5 demonstrate that the performance of the algorithm for nonlinear problems is significantly improved by mixing both "up" and "down" iterations.

The above assumption can be computationally checked by using a device similar to the one used in the code $\epsilon$-RELAX-N (Bertsekas 1991). For a node $i$, let $p_i^k$ be the number of "up" iterations executed at node $i$ before iteration number $k$. Also let

$$S^k := \sum_{i \in \mathcal{N}} |s_i^k|$$

We can now enforce the assumption by not allowing a "down" iteration for node $i$ at iteration number $k$ if

$$p_i^k > C_1 + C_2(S^0 - S^k)$$

where $C_1$ and $C_2$ are any two positive constants. The quantity $C_1 + C_2(S^0 - S^k)$ is nondecreasing and bounded above and is used as a measure of the progress made by the algorithm. As long as the algorithm makes sufficient progress, "up" and "down" iterations are allowed to be mixed. If, however, too many "up" iterations are executed at a given node without achieving progress, no more "down" iterations are allowed at this node.

Finally, we should note that the solution of problem (1) usually consists of more than one application of **Algorithm 1**. The algorithm is executed for some value of $\epsilon$ and terminated when the surplus of all nodes becomes relatively small. Then $\epsilon$ is reduced and the process is repeated. This is very similar to the way cost scaling algorithms operate.

We now describe a specialization of **Algorithm 1** for network flow problems with mixed linear and quadratic cost functions:

$$\text{(7)} \qquad \begin{array}{ll} \text{minimize} & \displaystyle\sum_{(i,j) \in \mathcal{A}} \bar{f}_{ij}(x_{ij}) \\ \text{subject to} & \displaystyle\sum_{j \in \delta_+(i)} x_{ij} - \sum_{j \in \delta_-(i)} x_{ji} = b_i, \quad i \in \mathcal{N} \\ & \qquad 0 \le x_{ij} \le u_{ij} \end{array}$$

where $\bar{f}_{ij}(x_{ij})$ are convex continuously differentiable quadratic functions:

$$\bar{f}_{ij}(x_{ij}) = \frac{1}{2} d_{ij}[x_{ij}]^2 + c_{ij} x_{ij}$$

with $d_{ij} \geq 0$.

Again, let $\{\mathcal{A}_0, \mathcal{A}_1\}$ be a partition of $\mathcal{A}$ such that $\mathcal{A}_1$ contains only arcs $(i,j)$ with $d_{ij} > 0$.

**Algorithm 2**

**Step 0** [Initialization] Start with $(x, \pi)$ such that

$$x_{ij} = \begin{cases} 0 & \text{if } c_{ij} - (\pi_i - \pi_j) > \epsilon \\ u_{ij} & \text{if } d_{ij}u_{ij} + c_{ij} - (\pi_i - \pi_j) < -\epsilon \end{cases}$$

For all the remaining arcs, the initial choice of $x$ must satisfy the capacity constraints for the arcs with linear cost,

$$\max\left\{0, \min\left\{u_{ij}, -\frac{\epsilon}{d_{ij}} - \frac{c_{ij} - (\pi_i - \pi_j)}{d_{ij}}\right\}\right\} \leq x_{ij} \leq \min\left\{u_{ij}, \max\left\{0, \frac{\epsilon}{d_{ij}} - \frac{c_{ij} - (\pi_i - \pi_j)}{d_{ij}}\right\}\right\}$$

for the remaining arcs in $\mathcal{A}_0$ and

$$x_{ij} = \max\left\{0, \min\left\{u_{ij}, -\frac{c_{ij} - (\pi_i - \pi_j)}{d_{ij}}\right\}\right\}$$

for the arcs in $\mathcal{A}_1$.
Set $k = 0$.

**Step 1** [Choose node with positive surplus] Choose a node $i$ such that

$$s_i := b_i - \sum_{j \in \delta_+(i)} x_{ij} + \sum_{j \in \delta_-(i)} x_{ji} > 0$$

and increment $k$. If no such node exists, STOP.

**Step 2** [Increase flow on outgoing arc in $\mathcal{A}_0$] If there is an arc $(i, j) \in \mathcal{A}_0$ such that

$$\begin{gathered} r_{ij} := d_{ij}x_{ij} + c_{ij} - (\pi_i - \pi_j) \leq -\tfrac{\epsilon}{2}, \\ x_{ij} < u_{ij} \end{gathered}$$

then change the flow on arc $(i, j)$,

$$x_{ij} \leftarrow x_{ij} + \min\left\{s_i,\ u_{ij} - x_{ij},\ -\frac{r_{ij}}{d_{ij}}\right\}$$

The flow on the remaining arcs and the dual costs remain unchanged. Update $s_i$ and if its new value is 0 go to Step 1 else return to Step 2.

**Step 3** [Decrease flow on incoming arc in $\mathcal{A}_0$] If there is an arc $(j, i) \in \mathcal{A}_0$ such that

$$\begin{gathered} r_{ji} := d_{ji}x_{ji} + c_{ji} - (\pi_j - \pi_i) \geq \tfrac{\epsilon}{2}, \\ x_{ji} > 0 \end{gathered}$$

then change the flow on arc $(j, i)$,

$$x_{ji} \leftarrow x_{ji} - \min\left\{s_i,\ x_{ji},\ \frac{r_{ji}}{d_{ji}}\right\}$$

The flow on the remaining arcs and the dual costs remain unchanged. Update $s_i$ and if its new value is 0 go to Step 1 else return to Step 3.

**Step 4** [Raise price and change flows in $\mathcal{A}_1$] Compute

$$\Delta_1 := \min_{\{j \in \delta_+(i) \,|\, (i,j) \in \mathcal{A}_0,\, x_{ij} < u_{ij}\}} \Big\{ c_{ij} - (\pi_i - \pi_j) \Big\}$$

$$\Delta_2 := \min_{\{j \in \delta_-(i) \,|\, (j,i) \in \mathcal{A}_0,\, x_{ji} > 0\}} \Big\{ - (c_{ji} - (\pi_j - \pi_i)) \Big\}$$

For each $\gamma$ and for each arc $(i,j) \in \mathcal{A}_1$ let

$$x_{ij}(\gamma) = \max\left\{ 0, \min\left\{ u_{ij}, -\frac{c_{ij} - (\pi_i + \gamma - \pi_j)}{d_{ij}} \right\} \right\}$$

Similarly for each arc $(j,i) \in \mathcal{A}_1$ let

$$x_{ji}(\gamma) = \max\left\{ 0, \min\left\{ u_{ji}, -\frac{c_{ji} - (\pi_j - \pi_i - \gamma)}{d_{ji}} \right\} \right\}$$

and let $\Delta_3$ be the value of $\gamma$ such that

$$b_i - \sum_{(i,j) \in \mathcal{A}_1} x_{ij}(\gamma) - \sum_{(i,j) \in \mathcal{A}_0} x_{ij} + \sum_{(j,i) \in \mathcal{A}_1} x_{ji}(\gamma) + \sum_{(j,i) \in \mathcal{A}_0} x_{ji} = 0$$

(where $\Delta_3 = +\infty$ if a solution does not exist).
Define:

$$\Delta := \min\{\Delta_1 + \epsilon, \Delta_2 + \epsilon, \Delta_3\}$$

Change the price on node $i$:

$$\pi_i \leftarrow \pi_i + \Delta$$

and adjust the flow on the incident arcs in $\mathcal{A}_1$ so that CS is satisfied. The flow on all other arcs and the prices of all remaining nodes remains unchanged. Update $s_i$ and if its new value is 0 go to Step 1 else return to Step 2.

**5. Numerical results.** In this section we describe computational experience using **Algorithm 2** to solve network flow test problems with mixed linear/quadratic and strictly quadratic cost functions. The code was written in C and run on a SPARCstation 20.

The quality of the computed primal-dual pair $(x^*, \pi^*)$ can be measured in terms of the flow conservation constraints violation and relative accuracy of the computed objective value. For all runs reported below we had

$$\max_{i \in \mathcal{N}} \left| b_i - \sum_{j \in \delta_+(i)} x^*_{ij} + \sum_{j \in \delta_-(i)} x^*_{ji} \right| \le 10^{-8}$$

which translates to relative feasibility violation given by

$$\max_{i \in \mathcal{N}} \left| b_i - \sum_{j \in \delta_+(i)} x^*_{ij} + \sum_{j \in \delta_-(i)} x^*_{ji} \right| / \|b\|_\infty \le 10^{-12}$$

The computed objective value was accurate to nearly ten significant digits:

$$\left|\frac{f(x^*) - q(\pi^*)}{f(x^*)}\right| \leq 10^{-10}$$

We obtained our test problems by modifying the standard NETGEN (Klingman, Napier & Stutz 1974) problems 1–10 and 16–25. The first ten problems are transportation problems while the rest are transshipment problems. The transshipment problems differ in number of transshipment sources and sinks, percentage of capacitated arcs and size of upper bounds. The last four problems in Table 1 were obtained by scaling the input data for the corresponding standard NETGEN problems by a factor of eight. By adding quadratic terms to the objective functions we generated the following three groups of test problems:

- "50% linear/50% quadratic." A quadratic term with coefficient 10.0 was added to the cost function of 50% of the arcs (randomly chosen). In the corresponding runs we chose $\mathcal{A}_0$ to be the set of linear arcs and $\mathcal{A}_1$ to be the set of quadratic arcs.
- "50% large quadratic/50% small quadratic." A quadratic term with coefficient 10.0 was added to half of the arc cost functions and a quadratic term with coefficient 0.001 was added to the other half. This corresponds to a strictly convex problem with an ill-conditioned Hessian. In the algorithm we chose $\mathcal{A}_1$ to be the set of arcs with large quadratic factors and $\mathcal{A}_0$ the set of arcs with small quadratic factors.
- "100% quadratic." A quadratic term with coefficient 10.0 was added to all arc cost functions. In the corresponding runs we chose $\mathcal{A}_0 = \emptyset$ and $\mathcal{A}_1 = \mathcal{A}$. (With these choices the algorithm is practically equivalent to the relaxation method for strictly convex problems.)

Table 1 presents the sizes of the test problems and the execution times of **Algorithm 2** in seconds for each of the three test groups. The results for the first and third groups are comparable (after taking into account the computer performance differential) to those reported in (Bertsekas et al. 1987) for similarly modified NETGEN problems. However, the method of (Bertsekas et al. 1987) had difficulties solving problems similar to "50% large quadratic/50% small quadratic," being more than five times slower for them compared to the corresponding "50% linear/50% quadratic" problems. Our method, on the other hand, solves problems in the second group in roughly the same time as corresponding problems in the first group. This shows that our algorithm is as efficient for ill-conditioned quadratic objectives as for mixed linear-quadratic objectives.

It is worth mentioning that mixing "up" and "down" iterations is very important for achieving good computational efficiency in practice. This is illustrated in Table 1 where numbers in parentheses correspond to runs where only "up" iterations were allowed (for some problems in the "100% quadratic" case executing only "up" iterations led to very large times not shown in the table). This should be contrasted to the observation that mixing "up" and "down" iterations does not lead to significant improvement in running time when solving linear min-cost problems using the $\epsilon$-relaxation algorithm.

We also present the solution times for the smaller test problems using the piecewise linear approximation algorithm for separable convex network flow problems (Kamesam & Meyer 1984). This algorithm constructs a sequence of problems having piecewise

TABLE 1

*Solution times for modified NETGEN problems using the $\epsilon$-relaxation algorithm (figures in parentheses correspond to doing only "up iterations.")*

| NETGEN Number | Number of Nodes | Number of Arcs | 50% linear 50% quadratic | | 50% large q. 50% small q. | | 100% quadratic | |
|---|---|---|---|---|---|---|---|---|
| 1 | 200 | 1308 | 0.22 | (0.99) | 0.24 | (0.83) | 0.14 | (4.72) |
| 2 | 200 | 1511 | 0.23 | | 0.25 | | 0.14 | |
| 3 | 200 | 2000 | 0.39 | | 0.34 | | 0.15 | |
| 4 | 200 | 2200 | 0.29 | | 0.30 | | 0.17 | |
| 5 | 200 | 2900 | 0.44 | (1.46) | 0.40 | (1.54) | 0.19 | (5.77) |
| 6 | 300 | 3174 | 0.37 | | 0.41 | | 0.24 | |
| 7 | 300 | 4519 | 0.84 | | 0.71 | | 0.29 | |
| 8 | 300 | 5169 | 0.91 | | 0.85 | | 0.37 | |
| 9 | 300 | 6075 | 1.16 | | 0.98 | | 0.48 | |
| 10 | 300 | 6320 | 1.27 | | 1.11 | | 0.50 | |
| 16 | 400 | 1306 | 1.25 | (1.77) | 0.90 | (1.67) | 0.28 | |
| 17 | 400 | 2443 | 1.10 | | 0.80 | | 0.25 | |
| 18 | 400 | 1306 | 1.46 | | 0.83 | | 0.28 | |
| 19 | 400 | 2443 | 1.05 | | 0.84 | | 0.26 | |
| 20 | 400 | 1416 | 1.0 | | 0.82 | | 0.29 | |
| 21 | 400 | 2836 | 1.49 | (3.0) | 1.04 | (2.17) | 0.27 | |
| 22 | 400 | 1416 | 1.16 | | 0.81 | | 0.29 | |
| 23 | 400 | 2836 | 1.29 | | 0.87 | | 0.27 | |
| 24 | 400 | 1382 | 0.85 | | 0.98 | | 0.28 | |
| 25 | 400 | 2676 | 0.75 | | 0.73 | | 0.27 | |
| 6S | 2400 | 25336 | 7.4 | | 7.1 | | 9.6 | |
| 10S | 2400 | 50535 | 12.4 | | 11.5 | | 12.1 | |
| 24S | 3200 | 11056 | 7.0 | | 7.0 | | 5.7 | |
| 25S | 3200 | 21408 | 12.6 | | 8.7 | | 5.5 | |

linear objective functions which converge to the original nonlinear objective and solves these problems using the network simplex method. Except for a few of the linear-quadratic problems, the times in Table 2 are larger than the $\epsilon-$relaxation times. Some additional testing demonstrated that the piecewise-linear approach was competitive for problems in which most arcs had linear costs, but since such problem classes were not the main focus of this research, we do not present these results here.

Finally, our relaxation algorithm consistently outperformed by a significant factor the general nonlinear network optimization package LSNNO (Toint & Tuyttens 1992) on the test problems described above.

**6. Parallel versions of the algorithm.** An iteration of **Algorithm 1** only involves a node in the given network and its adjacent arcs. Consequently, an obvious approach to parallelization would be to assign a node (or a group of nodes) to each processor in a parallel computing environment. Iterations at several different nodes would then be able to proceed simultaneously if only iterations at independent (i.e. no two are joined by an arc) nodes were allowed to proceed concurrently. A potential problem with this approach might be limited parallel efficiency due to insufficient number of independent nodes with non-zero surplus available at any given time.

TABLE 2

*Solution times for modified NETGEN problems using the piecewise-linear approximation method*

| NETGEN Number | Number of Nodes | Number of Arcs | 50% linear 50% quadratic | 50% large q. 50% small q. | 100% quadratic |
|---|---|---|---|---|---|
| 1 | 200 | 1308 | 0.74 | 1.69 | 2.17 |
| 2 | 200 | 1511 | 0.93 | 1.99 | 2.38 |
| 3 | 200 | 2000 | 1.18 | 2.84 | 3.36 |
| 4 | 200 | 2200 | 1.25 | 2.61 | 4.01 |
| 5 | 200 | 2900 | 1.46 | 3.49 | 5.26 |
| 6 | 300 | 3174 | 2.33 | 5.06 | 7.36 |
| 7 | 300 | 4519 | 2.65 | 8.31 | 10.3 |
| 8 | 300 | 5169 | 2.76 | 7.76 | 11.1 |
| 9 | 300 | 6075 | 3.65 | 9.06 | 14.3 |
| 10 | 300 | 6320 | 3.76 | 9.35 | 13.9 |
| 16 | 400 | 1306 | 1.31 | 8.26 | 5.44 |
| 17 | 400 | 2443 | 1.57 | 11.5 | 10.1 |
| 18 | 400 | 1306 | 0.97 | 6.27 | 4.04 |
| 19 | 400 | 2443 | 1.38 | 8.35 | 7.96 |
| 20 | 400 | 1416 | 1.86 | 17.2 | 7.36 |
| 21 | 400 | 2836 | 1.57 | 24.7 | 16.2 |
| 22 | 400 | 1416 | 1.49 | 7.57 | 5.14 |
| 23 | 400 | 2836 | 1.53 | 10.6 | 10.8 |
| 24 | 400 | 1382 | 1.98 | 11.0 | 6.18 |
| 25 | 400 | 2676 | 1.51 | 9.95 | 15.8 |

Another approach (described below) removes the node independency requirement by allowing iterations at two adjacent nodes to be executed in parallel. In this case, the two nodes may set the flow on the arc joining them to two different values. As the algorithm progresses, the difference between the two nodes' estimates of the flow approaches zero. This algorithm may be viewed as a generalization of the totally asynchronous algorithm for linear network flow problems of (Bertsekas & Tsitsiklis 1989), to nonlinear cost functions.

At time step $t$ any node $i$ has the following data:

- the price $\pi_i(t)$
- for any node $j \in \delta_+(i) \cup \delta_-(i)$, the price for node $j$ at some previous time step (denoted by $\pi_j(i,t)$)
- for any arc $(i,j) \in \delta_+(i)$, its estimate of the flow $x_{ij}(i,t)$ and for any arc $(j,i) \in \delta_-(i)$, its estimate of the flow $x_{ji}(i,t)$

The surplus of node $i$ at time $t$ is defined to be

$$s_i(t) := b_i - \sum_{j \in \delta_+(i)} x_{ij}(i,t) + \sum_{j \in \delta_-(i)} x_{ji}(i,t)$$

At any time step $t$ node $i$ does one of the following:

- If $s_i(t) > 0$, executes an "up iteration" (with $\epsilon' = 0$) and updates the values of $\pi_i(t)$, $x_{ij}(i,t)$ for $(i,j) \in \delta_+(i)$ and $x_{ji}(i,t)$ for $(j,i) \in \delta_-(i)$.
- Receives messages from adjacent nodes $j$ containing values $\pi_j(t')$ and $x_{ij}(j,t')$ for arcs $(i,j) \in \delta_+(i)$ or $x_{ji}(j,t')$ for arcs $(j,i) \in \delta_-(i)$ where $t' < t$. It

uses these values to update $\pi_j(i,t)$ and $x_{ij}(i,t)$ for $(i,j) \in \delta_+(i)$ ($x_{ji}(i,t)$ for $(j,i) \in \delta_-(i)$) according to the **update rule** outlined below.

**Update rule**: If $\pi_j(t') < \pi_j(i,t)$, do nothing. Else set $\pi_j(i,t) \leftarrow \pi_j(t')$. For an arc $(i,j) \in \delta_+(i)$, denote the largest flow that satisfies CS together with $\pi_i(t)$ and $\pi_j(i,t)$ by $y_{ij}(i,t)$. Now if $x_{ij}(j,t') < x_{ij}(i,t)$, set $x_{ij}(i,t)$ to the projection of $y_{ij}(i,t)$ on $[x_{ij}(j,t'), x_{ij}(i,t)]$. Similarly, for an arc $(j,i) \in \delta_-(i)$, denote the largest flow that satisfies CS together with $\pi_i(t)$ and $\pi_j(i,t)$ by $y_{ji}(i,t)$. If $x_{ji}(j,t') > x_{ji}(i,t)$, set $x_{ji}(i,t)$ to the projection of $y_{ji}(i,t)$ on $[x_{ji}(i,t), x_{ji}(j,t')]$.

The initial values of $(x,\pi)$ should satisfy

$$\pi_i(0) \geq \pi_i(j,0) \quad \forall j \neq i,$$

$$x_{ij}(i,0) \geq x_{ij}(j,0) \quad \forall (i,j)$$

together with the usual CS/$\epsilon$-CS conditions.

The structure of the "up iteration", the update rule and the initial conditions imply the following properties of the iterates:

1. The sequence $\{\pi_i(t)\}_{t=0}^{\infty}$ is nondecreasing and

$$\pi_i(t) \geq \pi_i(j,t') \quad \forall t' \leq t$$

2. $\epsilon$-CS is satisfied "locally" at each node $i$:

$$\begin{aligned} f_{ij}^-(x_{ij}(i,t)) - \epsilon \leq \pi_i(t) - \pi_j(i,t) &\leq f_{ij}^+(x_{ij}(i,t)) + \epsilon, \quad (i,j) \in \delta_+(i) \\ f_{ji}^-(x_{ji}(i,t)) - \epsilon \leq \pi_j(i,t) - \pi_i(t) &\leq f_{ji}^+(x_{ji}(i,t)) + \epsilon, \quad (j,i) \in \delta_-(i) \end{aligned}$$

3. The start node estimate of the flow on an arc dominates the end node estimate:

$$x_{ij}(i,t) \geq x_{ij}(j,t) \quad \forall t$$

Proof: Let $t$ be the first time step when the above fails. This can only happen if at time $t$ node $i$ received a message from $j$ causing $x_{ij}(i,t)$ to decrease or $j$ received a message from $i$ causing $x_{ij}(j,t)$ to increase. Without loss of generality assume the former. Let $t'$ be the time when the message in question was sent by $j$. Also let $s \leq t$ be the last time step when $j$ received a message from $i$ that increased $x_{ij}(j,t)$ and let $s'$ be the time when this message was sent. We necessarily have $s > t'$ for otherwise

$$x_{ij}(i,t) \geq x_{ij}(j,t') \geq x_{ij}(j,t).$$

Now using that $x_{ij}(j,t) > x_{ij}(i,t) \geq y_{ij}(i,t)$ and the definition of $y_{ij}(i,t)$ we get

$$\begin{aligned} f_{ij}^-(x_{ij}(j,t)) &> \pi_i(t) - \pi_j(t') \geq \pi_i(s') - \pi_j(s) \\ &\geq f_{ij}^-(y_{ij}(j,s)) \geq f_{ij}^-(x_{ij}(j,s)) \geq f_{ij}^-(x_{ij}(j,t)), \end{aligned}$$

a contradiction.

4. There exists a node which never executes an "up iteration." This follows from the fact that once nonnegative, the surplus of a node remains nonnegative and from the following inequality

$$\sum_i s_i(t) = \sum_{(i,j)} \{x_{ij}(j,t) - x_{ij}(i,t)\} \leq 0 \quad \forall t$$

In order to establish convergence of the algorithm, the usual assumption is needed that information in the system never becomes "too old." Using the above Property 4, it is easy to prove in a manner similar to Lemma 3.1 that the node prices converge.

We now show that the flow estimates of both end nodes of each arc converge to the same value:

$$x_{ij}(i,t) \to x^*_{ij}, \qquad x_{ij}(j,t) \to x^*_{ij}.$$

Boundedness of both sequences can be established in the same way as for the sequential algorithm. Denote the largest flow value that satisfies CS together with $\pi^*_i$ and $\pi^*_j$ (where $\pi^*_i$ and $\pi^*_j$ denote the limit values for the prices of nodes $i$ and $j$, respectively) by $y^*_{ij}$. Using that $y_{ij}(i,t)$ is the largest element of $\partial f_{ij}(\pi_i(t) - \pi_j(i,t))$ and the continuity properties of the subdifferential mapping we get

$$y_{ij}(i,t) \to y^*_{ij}, \qquad y_{ij}(j,t) \to y^*_{ij}.$$

Three cases need to be considered:

1. Both nodes $i$ and $j$ execute an infinite number of "up iterations" that affect their estimates of the flow on $(i,j)$. From the logic of the "up iteration" and the update rule we get

$$\limsup_t x_{ij}(i,t) \leq y^*_{ij} \leq \liminf_t x_{ij}(j,t)$$

which together with Property 3 establishes the result.

2. Node $i$ changes $x_{ij}(i,t)$ through "up iterations" only finite number of times while node $j$ changes $x_{ij}(j,t)$ through "up iterations" infinitely many times. Then $\{x_{ij}(i,t)\}$ is nonincreasing for sufficiently large $t$ and hence converges to some $x^*_{ij}$. We have

$$y^*_{ij} \leq \liminf_t x_{ij}(j,t) \leq x^*_{ij}$$

so by the update rule

$$\liminf_t x_{ij}(j,t) = x^*_{ij}$$

Hence $\{x_{ij}(j,t)\}$ also converges to $x^*_{ij}$.

3. Both nodes $i$ and $j$ change their respective estimates for the flow on $(i,j)$ only finitely many times through "up iterations." Hence $\{x_{ij}(i,t)\}$ is nonincreasing while $\{x_{ij}(j,t)\}$ is nondecreasing for sufficiently large $t$. By the update rule they can only converge to the same value (projections of a converging sequence on a sequence of nested segments).

Finally, feasibility of the common limit points for the original problem can be established in the same way as for the serial algorithm.

**7. Conclusions.** We have developed an $\epsilon$-relaxation method that is significantly more efficient on ill-conditioned networks than existing methods for nonlinear networks. Computational results with the procedure demonstrate that it solves networks with several thousand nonlinear arcs in one second or less. This method is also extended to an asynchronous parallel version that is amenable to parallel computation.

## REFERENCES


Bertsekas, Dimitri P., Patrick A. Hosein & Paul Tseng (1987), 'Relaxation methods for network flow problems with convex arc costs', *SIAM Journal on Control and Optimization* **25**(5), 1219–1243.

Bertsekas, D.P. (1991), *Linear Network Optimization:Algorithms and Codes*, The MIT Press, Cambridge, Massachusetts.

Bertsekas, D.P. & J.N. Tsitsiklis (1989), *Parallel and Distributed Computation*, Prentice Hall, Englewood Cliffs, New Jersey.

Collins, M., L. Cooper, R. Helgason, J. Kennington & L. LeBlanc (1978), 'Solving the pipe network analysis problem using optimization techniques', *Management Science* **24**, 747–760.

Hu, T.C. (1966), 'Minimum cost flows in convex cost networks', *Naval Research Logistics Quarterly* **13**, 1–9.

Kamesam, P.V. & R.R. Meyer (1984), 'Multipoint methods for separable nonlinear networks', *Mathematical Programming Study* **22**, 185–205.

Klingman, D., A. Napier & J. Stutz (1974), 'NETGEN—A program for generation large-scale (un)capacitated assignment, transportation and minimum cost network problems', *Management Science* **20**, 814–822.

Magnanti, T.L. (1984), Models and algorithms for predicting urban traffic equilibria, *in* M.Florian, ed., 'Transportation planning models', North–Holland, Amsterdam, pp. 153–186.

Monma, C.L. & M. Segal (1982), 'A primal algorithm for finding minimum-cost flows in capacitated networks with applications', *Bell System Technical Journal* **61**, 449–468.

Rockafellar, R.T (1970), *Convex Analysis*, Princeton University Press, Princeton, NJ.

Rockafellar, R.T. (1984), *Network Flows and Monotropic Optimization*, John Wiley & Sons, New York.

Toint, Ph.L. & D. Tuyttens (1992), 'LSNNO: a Fortran subroutine for solving large-scale nonlinear network opimization problems', *ACM Transactions on Mathematical Software* **18**(3), 308–328.