April 1995 LIDS-P-2146

SIAM J. on Optimization, to appear

AN $\mathcal{E}$-RELAXATION METHOD FOR SEPARABLE CONVEX COST NETWORK FLOW PROBLEMS[1]

by

Dimitri P. Bertsekas, Lazaros C. Polymenakos,[2] and Paul Tseng[3]

Abstract

We propose a new method for the solution of the single commodity, separable convex cost network flow problem. The method generalizes the ϵ-relaxation method developed for linear cost problems, and reduces to that method when applied to linear cost problems. We show that the method terminates with a near optimal solution, and we provide an associated complexity analysis. We also present computational results showing that the method is much faster than earlier relaxation methods, particularly for ill-conditioned problems.

[1] Research supported by NSF under Grant CCR-9103804 and Grant 9300494-DMI.

[2] Department of Electrical Engineering and Computer Science, M.I.T., Rm. 35-210, Cambridge, Mass., 02139. Email: dimitrib@mit.edu and lcpolyme@lids.mit.edu

[3] Department of Mathematics, Univ. of Washington, Seattle, Wash., 98195. Email address: tseng@math.washington.edu

1. INTRODUCTION

We consider a directed graph with node set $\mathcal{N} = \{1, \ldots, N\}$ and arc set $\mathcal{A} \subset \mathcal{N} \times \mathcal{N}$. The number of nodes is N and the number of arcs is denoted by A. We denote by x_{ij} the *flow* of the arc (i, j), and we refer to the vector $x = \{x_{ij} \mid (i, j) \in \mathcal{A}\}$ as the flow vector. The separable convex cost network flow problem is

$$\begin{aligned} &\text{minimize} \quad \sum_{(i,j)\in\mathcal{A}} f_{ij}(x_{ij}) && \text{(P)} \\ &\text{subject to} \quad \sum_{\{j \mid (i,j)\in\mathcal{A}\}} x_{ij} - \sum_{\{j \mid (j,i)\in\mathcal{A}\}} x_{ji} = s_i, \qquad \forall\, i \in \mathcal{N}, && (1) \end{aligned}$$

where s_i are given scalars, $f_{ij} : \Re \to (-\infty, \infty]$ are given convex, closed, proper functions (extended real-valued, lower semicontinuous, not identically taking the value ∞). We refer to problem (P) as the *primal* problem. We have implicitly assumed that there exists at most one arc in each direction between any pair of nodes, but this assumption is made for notational convenience and can be easily dispensed with. A flow vector x with $f_{ij}(x_{ij}) < \infty$ for all $(i, j) \in \mathcal{A}$, which satisfies the conservation of flow constraint (1) is called *feasible*. For a given flow vector x, the *surplus* of node i is defined as the difference between the supply s_i and the net outflow from i:

$$g_i = s_i + \sum_{\{j \mid (j,i)\in\mathcal{A}\}} x_{ji} - \sum_{\{j \mid (i,j)\in\mathcal{A}\}} x_{ij}. \tag{2}$$

We will assume that *there exists at least one feasible flow vector x such that*

$$f_{ij}^-(x_{ij}) < \infty \quad \text{and} \quad f_{ij}^+(x_{ij}) > -\infty, \qquad \forall\, (i, j) \in \mathcal{A}, \tag{3}$$

where $f_{ij}^-(x_{ij})$ and $f_{ij}^+(x_{ij})$ denote the left and right directional derivative of f_{ij} at x_{ij} [Roc84, p. 329].

There is a well-known duality framework for this problem, primarily developed by Rockafellar [Roc70], and discussed in several texts; see e.g. [Roc84], [BeT89]. This framework involves a Lagrange multiplier p_i for the ith conservation of flow constraint (1). We refer to p_i as the *price* of node i, and to the vector $p = \{p_i \mid i \in \mathcal{N}\}$ as the *price vector*. The *dual* problem is

$$\begin{aligned} &\text{minimize} \quad q(p) && \text{(D)} \\ &\text{subject to} \quad \text{no constraint on } p, \end{aligned}$$

where the dual functional q is given by

$$q(p) = \sum_{(i,j)\in\mathcal{A}} q_{ij}(p_i - p_j) - \sum_{i\in\mathcal{N}} s_i p_i,$$

and q_{ij} is related to f_{ij} by the conjugacy relation

$$q_{ij}(t_{ij}) = \sup_{x_{ij} \in \Re} \{x_{ij}t_{ij} - f_{ij}(x_{ij})\}.$$

We will assume throughout that f_{ij} *is such that* q_{ij} *is real-valued for all* $(i,j) \in \mathcal{A}$. This is true for example if each function f_{ij} takes the value ∞ outside some compact interval.

It is known (see [Roc84, p. 360]) that, under our assumptions, both the primal problem (P) and the dual problem (D) have optimal solutions and their optimal costs are the negatives of each other. The standard optimality conditions for a feasible flow-price vector pair (x, p) to be primal and dual optimal are

$$f_{ij}^-(x_{ij}) \le p_i - p_j \le f_{ij}^+(x_{ij}), \qquad \forall\ (i,j) \in \mathcal{A}.$$

These, known as the *complementary slackness conditions* (CS conditions for short), may be represented explicitly as

$$(x_{ij}, p_i - p_j) \in \Gamma_{ij}, \qquad \forall\ (i,j) \in \mathcal{A},$$

where

$$\Gamma_{ij} = \big\{(x_{ij}, t_{ij}) \in \Re^2 \mid f_{ij}^-(x_{ij}) \le t_{ij} \le f_{ij}^+(x_{ij})\big\}$$

is the *characteristic curve* associated with arc (i,j), as shown in Fig. 1.

The traditional methods for solving the problem of this paper for the case of linear arc cost functions (when each f_{ij} is linear on some closed interval and is ∞ outside the interval) are primal and dual cost improvement methods, which iteratively improve the primal or the dual cost function. Recently, methods based on the auction approach have gained attention, following the original proposal of the auction algorithm for the assignment problem [Ber79], and the ϵ-relaxation method [Ber86a], [Ber86b]. These methods may not improve the primal or the dual cost at any iteration, and they are based on a relaxed version of the CS conditions, called ϵ-complementary slackness (ϵ-CS for short). Their worst-case computational complexity, when properly implemented, is excellent as shown in [Gol87] (see also [BeE88], [BeT89], [GoT90]). Their practical performance is also very good, particularly for special classes of problems such as assignment and max-flow. Furthermore, these methods are well-suited for parallel implementation (see [BCE95], [LiZ91], [NiZ93]). We will focus on extending one such method, the ϵ-relaxation method, to the general convex cost case.

One possibility for dealing with the convex cost case is to use efficient ways to reduce the problem to an essentially linear cost problem by piecewise linearization of the arc cost functions; see [Mey79], [KaM84], [Roc84]. Another possibility is to use differentiable unconstrained optimization methods, such as coordinate descent [BHT87], conjugate gradient [Ven91], and adaptations of other nonlinear programming methods [HaH93], [Hag92], or fixed point methods [BeE87], [TBT90]. However, these methods require that the dual cost function be differentiable, which essentially amounts to the primal cost functions being strictly convex. A more general alternative, which applies to nondifferentiable dual cost functions as well, is to use an extension of the primal or dual cost improvement methods developed for the linear cost case. In particular, there have been proposals of primal cost improvement methods in [Wei74] and more recently in [KaM93]. There have also been proposals of dual cost improvement methods: the fortified descent method [Roc84] that extends the primal-dual method of Ford and Fulkerson [FoF62], and the relaxation method of [BHT87] that extends the corresponding linear cost relaxation method of [Ber85] and [BeT88]. These methods maintain, together with the price vector, a flow vector that satisfies the ϵ-CS conditions, and progressively work towards primal feasibility. The flow vector becomes feasible at termination.

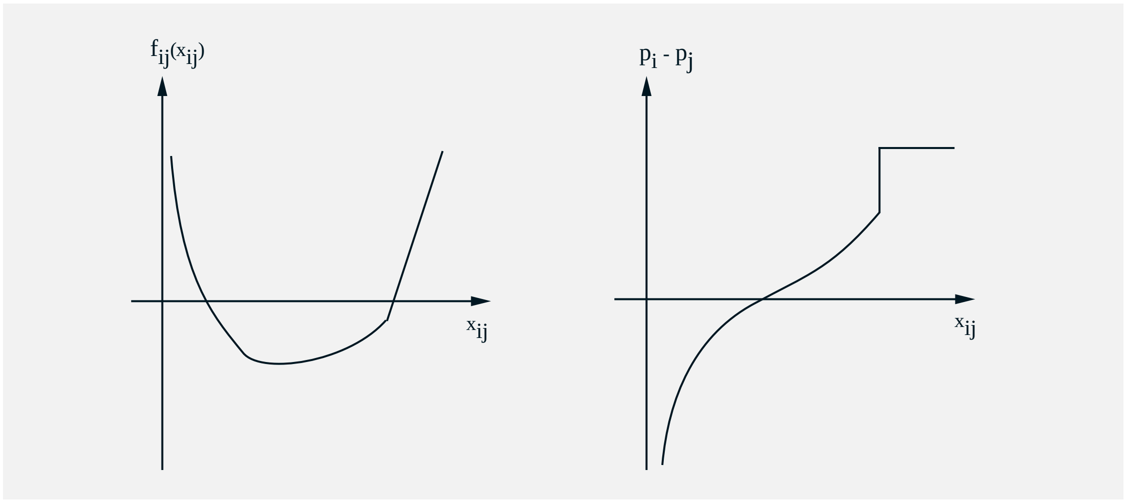

Figure 1: A cost function f_{ij} and its corresponding characteristic curve.

In this paper we develop and analyze the first extension of an auction method, the ϵ-relaxation method, to the convex arc cost case.[1] (An analogous extension of the auction/sequential shortest path method given in [Ber92], which has also been incorporated in the latest release of the RELAX code [BeT94], is developed in the forthcoming Ph.D. thesis of the second author

[1] We have learned that the same method was independently developed and analyzed by DeLeone, Meyer, and Zakarian [DMZ95]. The results of their computational tests qualitatively agree with ours.

[Pol95].) Our method is based on the ϵ-CS conditions first introduced in [BHT87] for the case of convex arc costs. In particular, we say that the flow vector x and the price vector p satisfy ϵ-CS if and only if

$$f_{ij}(x_{ij}) < \infty, \quad \text{and} \quad f^-_{ij}(x_{ij}) - \epsilon \leq p_i - p_j \leq f^+_{ij}(x_{ij}) + \epsilon, \qquad \forall\ (i,j) \in \mathcal{A}. \tag{4}$$

(see Fig. 2).

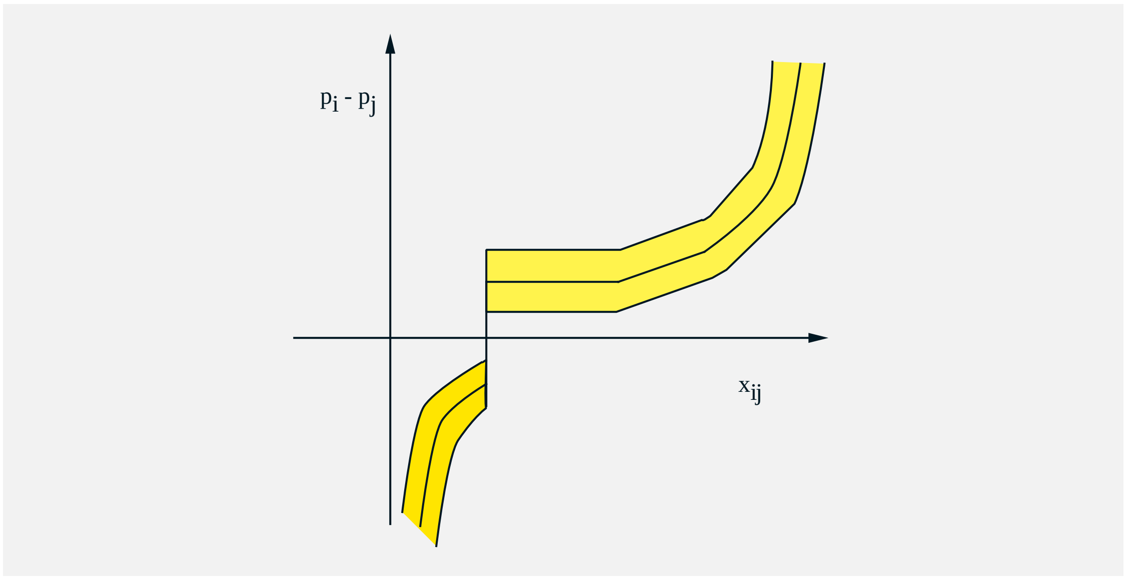

Figure 2: A visualization of the ϵ-CS conditions as a cylinder around the characteristic curve (bold line) The shaded area represents flow-price differential pairs that satisfy the ϵ-CS conditions.

It was shown in [BHT87] that if a feasible flow-price vector pair (x, p) satisfies ϵ-CS, then x and p are primal and dual optimal, respectively, within a factor that is proportional to ϵ (see the following Prop. 6). Our method is similar to the ϵ-relaxation method for linear cost network flow problems. It iteratively modifies the price vector, while effecting attendant flow changes that maintain the ϵ-CS conditions. The method terminates with a feasible flow-price vector pair which, however, satisfies ϵ-CS rather than CS. There is a fundamental difference from the other dual descent methods for nondifferentiable dual cost problems: the price changes are made exclusively along coordinate directions, that is, one price at a time, and a price change need not improve the dual cost. However, because the flow-price vector pair (x, p) maintained by the ϵ-relaxation method satisfies ϵ-CS rather than CS, there is more freedom in adjusting the flow-price vector pair towards feasibility, even though the pair obtained when the method terminates is optimal only within a factor proportional to ϵ.

The method of this paper essentially provides a mechanism to move around the ϵ-CS diagram of Fig. 2 while changing one price at a time, and working towards primal feasibility. There is a variety of mechanisms for effecting such price changes and Fig. 3 illustrates some of the possibilities. In particular, starting from a point on the characteristic curve of arc (i, j), we can follow any direction around that point and change the price p_i or the price p_j, and/or the flow x_{ij} simultaneously until $(x_{ij}, p_i - p_j)$ is either on the characteristic curve or is within a distance of ϵ above or below the characteristic curve of arc (i, j). For example, if node i has positive surplus, by increasing the flow of an outgoing arc (i, j) or by decreasing the flow of an incoming arc (j, i), the surplus of i will be decreased, while the surplus of j will be increased by an equal amount. This is the basic mechanism for moving flow from nodes of positive surplus to nodes of negative surplus, thus working towards primal feasibility. It is possible, however, that node i has positive surplus, while the flow of none of the outgoing arcs (i, j) can be increased and the flow of none of the incoming arcs (j, i) can be decreased without violating the ϵ-CS conditions. In this case the method increases the price of node i in order to "make room" in the ϵ-CS diagram for a subsequent flow change.

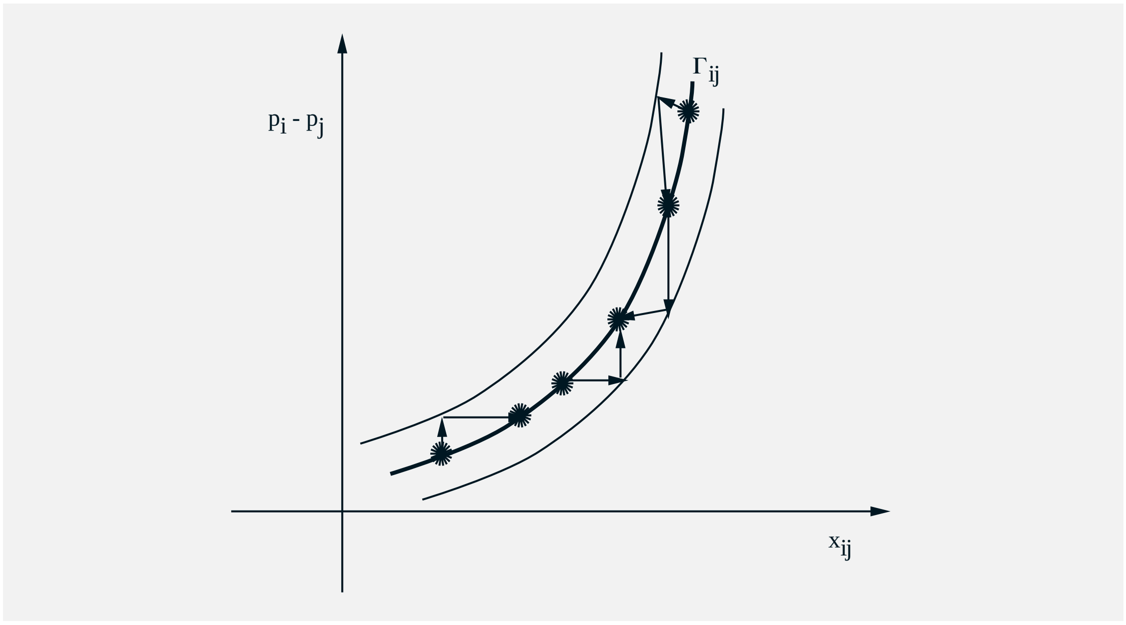

Figure 3: Starting from any point on the characteristic curve (dark points) of arc (i, j), a new point on the characteristic curve can be obtained in a variety of ways. The figure depicts a few such examples where the flow and price differential for arc (i, j) are changed simultaneously according to some linear relation.

The paper is organized as follows. In Section 2 we present the ϵ-relaxation method extended to solve convex cost problems. In Section 3, we show that this method terminates with a near optimal flow-price vector pair and, in Section 4, we provide a complexity analysis for the method. In Section 5, we describe a version of this method that uses both price increase and decrease steps and, in Section 6, we report our computational experience with the methods of Sections 2 and 5 on some convex linear/quadratic cost problems. Our test results show that, on problems where some (possibly all) arcs have strictly convex cost, the new method outperforms, often by an impressive margin, earlier relaxation methods. Significantly, our method seems to be minimally affected by ill-cnditioning in the dual problem. We do not know of any other method for which this is true.

2. THE ϵ-RELAXATION METHOD

For a flow-price vector pair (x, p) satisfying ϵ-CS, we define for each node $i \in \mathcal{N}$ its *push list* as the set of arcs

$$\big\{(i,j) \mid \epsilon/2 < p_i - p_j - f_{ij}^+(x_{ij}) \le \epsilon\big\} \cup \big\{(j,i) \mid -\epsilon \le p_j - p_i - f_{ji}^-(x_{ji}) < -\epsilon/2\big\}. \tag{5}$$

Figure 4 illustrates when an arc (i,j) is in the push list of i and when it is in the push list of j. We note that a more general definition of the push list can be given by replacing the term $\epsilon/2$ with $\beta\epsilon$, where β is a scalar with $0 < \beta < 1$. The subsequent analysis applies, with minor modifications, to the corresponding version of the ϵ-relaxation method to be given shortly.

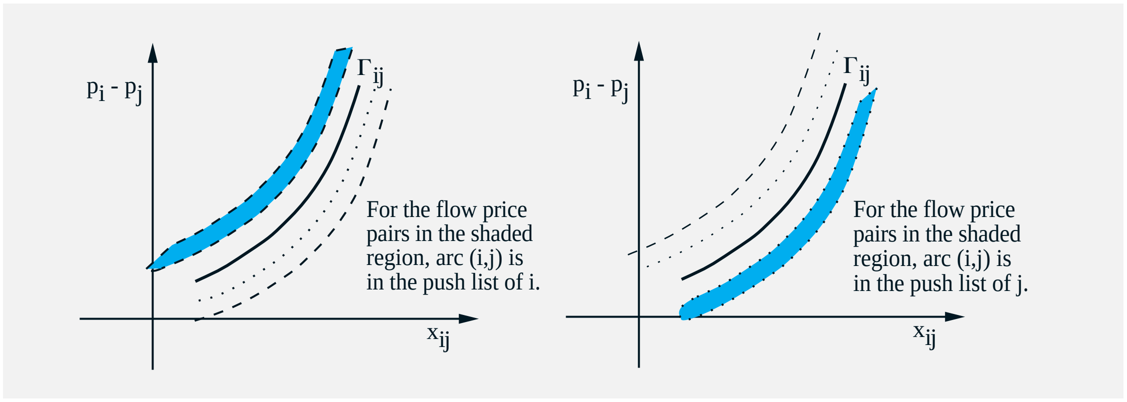
p_i - p_j
Γ_ij
For the flow price pairs in the shaded region, arc (i,j) is in the push list of i.
x_ij
p_i - p_j
Γ_ij
For the flow price pairs in the shaded region, arc (i,j) is in the push list of j.
x_ij

Figure 4: A visualization of the conditions satisfied by a push list arc. The shaded area represents flow-price differential pairs corresponding to a push list arc.

An arc (i,j) [or (j,i)] in the push list of i is said to be *unblocked* if there exists a $\delta > 0$ such that

$$p_i - p_j \geq f_{ij}^+(x_{ij} + \delta),$$

[or $p_j - p_i \leq f_{ji}^-(x_{ji} - \delta)$, respectively]. For an unblocked push list arc, the supremum of δ for which the above relation holds is called the *flow margin* of the arc. The flow margin of an arc (i,j) is illustrated in Fig. 5. An important property is the following:

Proposition 1: The arcs in the push list of a node are unblocked.

Proof: Assume that for an arc $(i,j) \in \mathcal{A}$ we have

$$p_i - p_j < f_{ij}^+(x_{ij} + \delta), \quad \forall\, \delta > 0.$$

Since the function f_{ij}^+ is right continuous, this yields

$$p_i - p_j \leq \lim_{\delta \downarrow 0} f_{ij}^+(x_{ij} + \delta) = f_{ij}^+(x_{ij}),$$

and thus (i,j) cannot be in the push list of node i. A similar argument proves that an arc $(j,i) \in \mathcal{A}$ such that

$$p_j - p_i > f_{ji}^-(x_{ji} - \delta), \quad \forall\, \delta > 0,$$

cannot be in the push list of node i. **Q.E.D.**

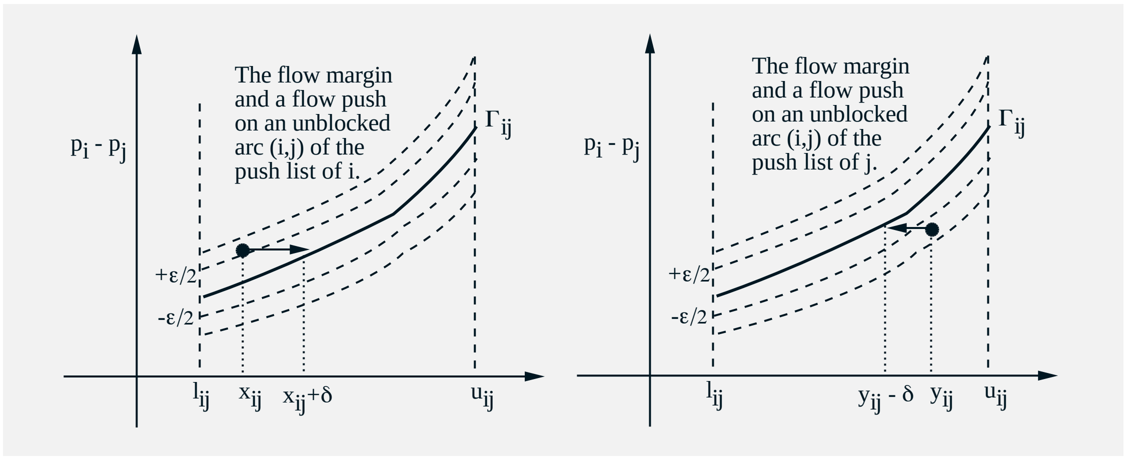
The flow margin and a flow push on an unblocked arc (i,j) of the push list of i.
p_i - p_j
Γ_{ij}
+ε/2
-ε/2
l_{ij}
x_{ij}
x_{ij}+δ
u_{ij}
The flow margin and a flow push on an unblocked arc (i,j) of the push list of j.
p_i - p_j
Γ_{ij}
+ε/2
-ε/2
l_{ij}
y_{ij} - δ
y_{ij}
u_{ij}

Figure 5: The flow margin of an unblocked push list arc.

For a given pair (x,p) satisfying ϵ-CS, consider an arc set $\mathcal{A}^*$ that contains all push list arcs oriented in the direction of flow change. In particular, for each arc (i,j) in the push list of a node

i we introduce an arc (i, j) in $\mathcal{A}^*$ and for each arc (j, i) in the push list of node i we introduce an arc (i, j) in $\mathcal{A}^*$. The set of nodes $\mathcal{N}$ and the set $\mathcal{A}^*$ define the *admissible graph* $G^* = (\mathcal{N}, \mathcal{A}^*)$. Note that an arc can be in the push list of at most one node, so the admissible graph is well defined.

The ϵ-relaxation method starts with a flow-price vector pair (x, p) satisfying ϵ-CS, and such that the corresponding admissible graph G^* is acyclic. One possibility is to select an initial price vector p^0 and to set the initial arc flow for every arc $(i, j) \in \mathcal{A}$ to

$$x_{ij} = \sup\{\xi \mid f_{ij}^+(\xi) \le p_i^0 - p_j^0 - \epsilon/2\}. \tag{6}$$

It can be seen that with this choice, ϵ-CS is satisfied for every arc $(i, j) \in \mathcal{A}$, and that the initial admissible graph is empty and thus acyclic.

In the typical iteration of the method, we select a node i with positive surplus, and we perform one or more of the following two operations:

(a) A *price rise* on node i, which increases the price p_i by the maximum amount that maintains ϵ-CS, while leaving all arc flows unchanged.

(b) A *flow push* (also called a *δ-flow push*) along an arc (i, j) [or along an arc (j, i)], which increases (i, j) [or decreases (j, i)] by an amount $\delta \in (0, g_i]$, while leaving all node prices unchanged.

The iteration is as follows.

Typical Iteration of the ϵ-Relaxation Method

Step 1: Select a node i with positive surplus g_i; if no such node exists, terminate the method.

Step 2: If the push list of i is empty, go to Step 3. Otherwise, choose an arc from the push list of i and perform a δ-flow push towards the opposite node j, where

$$\delta = \min\{g_i, \text{flow margin of arc}\}.$$

If the surplus of i becomes zero, go to the next iteration; otherwise go to Step 2.

Step 3: Increase the price p_i by the maximum amount that maintains ϵ-CS. Go to the next iteration.

We make the following observations about the ϵ-relaxation method:

1. The method preserves ϵ-CS and the prices are monotonically nondecreasing. This is evident from the initialization and Step 3 of the method.

2. Once the surplus of a node becomes nonnegative, it remains nonnegative for all subsequent iterations. The reason is that a flow push at a node i cannot make the surplus of i negative (cf. Step 2), and cannot decrease the surplus of neighboring nodes.

3. If at some iteration a node has negative surplus, then its price must be equal to its initial price. This is a consequence of observation 2 above and the fact that price changes occur only on nodes with positive surplus.

3. TERMINATION OF THE ϵ-RELAXATION METHOD

To prove the termination of the ϵ-relaxation method of Section 2, we first prove that the total number of price rises that the method can perform is bounded.

Proposition 2: Each price rise increment in the ϵ-relaxation method is at least $\epsilon/2$.

Proof: We first note that a price rise on a node i occurs only when its push list is empty. Thus for every arc $(i,j) \in \mathcal{A}$ we have $p_i - p_j - f_{ij}^+(x_{ij}) \leq \epsilon/2$, and for every arc $(j,i) \in \mathcal{A}$ we have $p_j - p_i - f_{ji}^-(x_{ji}) \geq -\epsilon/2$. This implies that all elements of the following sets of positive numbers:

$$S^+ = \left\{p_j - p_i + f_{ij}^+(x_{ij}) + \epsilon \mid (i,j) \in \mathcal{A}\right\},$$

$$S^- = \left\{p_j - p_i - f_{ji}^-(x_{ji}) + \epsilon \mid (j,i) \in \mathcal{A}\right\}$$

are greater than or equal to $\epsilon/2$. Since a price rise at i increases p_i by the increment $\gamma = \min\{S^+ \cup S^-\}$, the result follows. **Q.E.D.**

The following proposition bounds the total number of price increases that the ϵ-relaxation method can perform on any node. The proof is patterned after that for the linear cost case [Ber86a], [BeE88].

Proposition 3: Assume that for some integer $K \geq 1$, the initial price vector p^0 for the ϵ-relaxation method satisfies $K\epsilon$-CS together with some feasible flow vector x^0. Then, the ϵ-relaxation method performs at most $2(K+1)(N-1)$ price rises per node.

Proof: Consider the pair (x,p) at the beginning of an ϵ-relaxation iteration. Since the surplus vector $g = (g_1, \ldots, g_N)$ is not zero, and the flow vector x^0 is feasible, we conclude that for each node s with $g_s > 0$ there exists a node t with $g_t < 0$ and a path H from t to s that contains no cycles and is such that:

$$x_{ij} > x_{ij}^0, \qquad \forall\ (i,j) \in H^+, \tag{7}$$

$$x_{ij} < x_{ij}^0, \qquad \forall\ (i,j) \in H^-, \tag{8}$$

where H^+ is the set of forward arcs of H and H^- is the set of backward arcs of H. [This can be seen from the Conformal Realization theorem ([Roc84] or [Ber91]) as follows. For the flow vector $x - x^0$, the net outflow from node t is $-g_t > 0$ and the net outflow from node s is $-g_s < 0$ (here we ignore the flow supplies), so, by the Conformal Realization Theorem, there is a path H from t to s that contains no cycle and conforms to the flow $x - x^0$, that is, $x_{ij} - x_{ij}^0 > 0$ for all $(i,j) \in H^+$ and $x_{ij} - x_{ij}^0 < 0$ for all $(i,j) \in H^-$. Eqs. (7) and (8) then follow.]

From Eqs. (7) and (8), and the convexity of the functions f_{ij} for all $(i,j) \in \mathcal{A}$, we have

$$f_{ij}^-(x_{ij}) \geq f_{ij}^+(x_{ij}^0), \qquad \forall\ (i,j) \in H^+, \tag{9}$$

$$f_{ij}^+(x_{ij}) \leq f_{ij}^-(x_{ij}^0), \qquad \forall\ (i,j) \in H^-. \tag{10}$$

Since the pair (x,p) satisfies ϵ-CS, we also have that

$$p_i - p_j \in [f_{ij}^-(x_{ij}) - \epsilon, f_{ij}^+(x_{ij}) + \epsilon], \qquad \forall\ (i,j) \in \mathcal{A}. \tag{11}$$

Similarly, since the pair (x^0, p^0) satisfies $K\epsilon$-CS, we have

$$p_i^0 - p_j^0 \in [f_{ij}^-(x_{ij}^0) - K\epsilon, f_{ij}^+(x_{ij}^0) + K\epsilon], \qquad \forall\ (i,j) \in \mathcal{A}. \tag{12}$$

Combining Eqs. (9)-(12), we obtain

$$p_i - p_j \geq p_i^0 - p_j^0 - (K+1)\epsilon, \qquad \forall\ (i,j) \in H^+,$$

$$p_i - p_j \leq p_i^0 - p_j^0 + (K+1)\epsilon, \qquad \forall\ (i,j) \in H^-.$$

Applying the above inequalities for all arcs of the path H, we get

$$p_t - p_s \geq p_t^0 - p_s^0 - (K+1)|H|\epsilon, \tag{13}$$

where $|H|$ denotes the number of arcs of the path H. We observed earlier that if a node has negative surplus at some time, then its price is unchanged from the beginning of the method until that time. Thus $p_t = p_t^0$. Since the path contains no cycles, we also have that $|H| \leq N - 1$. Therefore, Eq. (13) yields

$$p_s - p_s^0 \leq (K+1)|H|\epsilon \leq (K+1)(N-1)\epsilon. \tag{14}$$

Since only nodes with positive surplus can increase their prices and, by Prop. 2, each price rise increment is at least $\epsilon/2$, we conclude from Eq. (14) that the total number of price rises that can be performed for node s is at most $2(K+1)(N-1)$. **Q.E.D.**

The result of the preceding proposition is remarkable in that the bound on the number of price changes is independent of the cost functions, but depends only on

$$K^0 = \min\{K \in \{0,1,2,\ldots\} \mid (x^0,p^0) \text{ satisfies } K\epsilon\text{-CS for some feasible flow vector } x^0 \},$$

which is the minimum multiplicity of ϵ by which CS is violated by the starting price together with some feasible flow vector. This result will be used later to prove a particularly favorable complexity bound for the method. Note that K^0 is well defined for any p^0 because, for all K sufficiently large, $K\epsilon$-CS is satisfied by p^0 and the feasible flow vector x satisfying Eq. (3).

In order to show that the number of flow pushes that can be performed between successive price increases is finite, we first prove that the method maintains the acyclicity of the admissible graph.

Proposition 4: The admissible graph remains acyclic throughout the ϵ-relaxation method.

Proof: We use induction. Initially, the admissible graph G^* is empty, so it is trivially acyclic. Assume that G^* remains acyclic for all subsequent iterations up to the mth iteration for some m. We will prove that after the mth iteration G^* remains acyclic. Clearly, after a flow push the admissible graph remains acyclic, since it either remains unchanged, or some arcs are deleted from it. Thus we only have to prove that after a price rise at a node i, no cycle involving i is created. We note that, after a price rise at node i, all incident arcs to i in the admissible graph at the start of the mth iteration are deleted and new arcs incident to i are added. We claim that i cannot have any incoming arcs which belong to the admissible graph. To see this, note that, just before a price rise at node i, we have from (4) that

$$p_j - p_i - f_{ji}^-(x_{ji}) \leq \epsilon, \qquad \forall\ (j,i) \in \mathcal{A},$$

and since each price rise is at least $\epsilon/2$, we must have

$$p_j - p_i - f_{ji}^-(x_{ji}) \leq \frac{\epsilon}{2}, \qquad \forall\ (j,i) \in \mathcal{A},$$

after the price rise. Then, by Eq. (5), (j,i) cannot be in the push list of node j. By a similar argument, we have that (i,j) cannot be in the push list of j for all $(i,j) \in \mathcal{A}$. Thus, after a price increase at i, node i cannot have any incoming incident arcs belonging to the admissible graph, so no cycle involving i can be created. **Q.E.D.**

We say that a node i is a *predecessor* of a node j in the admissible graph G^* if a directed path from i to j exists in G^*. Node j is then called a *successor* of i. Observe that flow is pushed towards the successors of a node and since G^* is acyclic, flow cannot be pushed from a node to any of its predecessors. A δ-flow push along an arc in G^* is said to be *saturating* if δ is equal to the flow margin of the arc. By our choice of δ (see Step 2 of the method), a nonsaturating flow push always exhausts (i.e., sets to zero) the surplus of the starting node of the arc. Thus we have the following proposition.

Proposition 5: The number of flow pushes between two successive price increases (not necessarily at the same node) performed by the ϵ-relaxation method is finite.

Proof: We observe that a saturating flow push along an arc removes the arc from the admissible graph, while a nonsaturating flow push does not add a new arc to the admissible graph. Thus the number of saturating flow pushes that can be performed between successive price increases is at most A. It will thus suffice to show that the number of nonsaturating flow pushes that can be performed between saturating flow pushes is finite. Assume the contrary, that is, there is an infinite sequence of successive nonsaturating flow pushes, with no intervening saturating flow push. Then, the surplus of some node i^0 must be exhausted infinitely often during this sequence. This can happen only if the surplus of some predecessor i^1 of i^0 is exhausted infinitely often during the sequence. Continuing in this manner we construct an infinite succession of predecessor nodes $\{i^k\}$. Thus some node in this sequence must be repeated, which is a contradiction since the admissible graph is acyclic. **Q.E.D.**

By refining the proof of Prop. 5, we can further show that the number of flow pushes between successive price increases is at most $(N+1)A$, from which a complexity result for the ϵ-relaxation method may be derived. However, we will defer the analysis of complexity to Section 4, where an implementation of the method with sharper complexity bound will be presented.

Propositions 3 and 5 prove that the ϵ-relaxation method terminates. Upon termination, we have that the flow-price vector pair satisfies ϵ-CS and that the flow vector is feasible since the surplus of all nodes will be zero. The following proposition, due to [BHT87], shows that the flow vector and the price vector obtained upon termination are primal optimal and dual optimal within a factor that is essentially proportional to ϵ.

Proposition 6: For each $\epsilon > 0$, let $x(\epsilon)$ and $p(\epsilon)$ denote any flow and price vector pair satisfying ϵ-CS with $x(\epsilon)$ feasible and let $\xi(\epsilon)$ denote any flow vector satisfying CS together with

$p(\epsilon)$ [note that $\xi(\epsilon)$ need not be feasible]. Then

$$0 \leq f\big(x(\epsilon)\big) + q\big(p(\epsilon)\big) \leq \epsilon \sum_{(i,j)\in\mathcal{A}} |x_{ij}(\epsilon) - \xi_{ij}(\epsilon)| \,.$$

Furthermore, $f\big(x(\epsilon)\big) + q\big(p(\epsilon)\big) \to 0$ as $\epsilon \to 0$.

Proposition 6 does not give an estimate of how small ϵ has to be in order to achieve a certain degree of optimality. However, in the common case where finiteness of the arc cost functions f_{ij} imply lower and upper bounds on the arc flows, that is,

$$-\infty < b_{ij} = \inf_{\xi}\{\xi \mid f_{ij}(\xi) < \infty\} \leq \sup_{\xi}\{\xi \mid f_{ij}(\xi) < \infty\} = c_{ij} < \infty,$$

Prop. 6 together with the fact $q\big(p(\epsilon)\big) \geq -f^*$ yields the estimate

$$0 \leq f\big(x(\epsilon)\big) - f^* \leq \epsilon A \max_{(i,j)\in\mathcal{A}} |c_{ij} - b_{ij}|,$$

where f^* is the optimal cost of (P). Similarly, we obtain

$$0 \leq q\big(p(\epsilon)\big) - q^* \leq \epsilon A \max_{(i,j)\in\mathcal{A}} |c_{ij} - b_{ij}|,$$

where q^* is the optimal cost of (D).

4. COMPLEXITY ANALYSIS FOR THE ϵ-RELAXATION METHOD

We now derive a bound on the running time of the ϵ-relaxation method. Because the cost functions are convex, it is not possible to express the size of the problem in terms of the problem data. To deal with this difficulty, we introduce a set of simple operations performed by the method, and we estimate the number of these operations. In particular, in addition to the usual arithmetic operations with real numbers, we consider the following operations:

(a) Given the flow x_{ij} of an arc (i,j), calculate the cost $f_{ij}(x_{ij})$, the left derivative $f_{ij}^-(x_{ij})$, and the right derivative $f_{ij}^+(x_{ij})$.

(b) Given the price differential t_{ij} of an arc (i,j), calculate $\sup\{\xi \mid f_{ij}^+(\xi) \leq t_{ij}\}$ and $\inf\{\xi \mid f_{ij}^-(\xi) \geq t_{ij}\}$.

Operation (a) is needed to compute the push list of a node and a price increase increment; operation (b) is needed to compute the flow margin of an arc and the flow initialization of Eq. (6). We will thus estimate the total number of simple operations performed by the method (see the following Prop. 8).

To obtain a sharper complexity bound, we introduce an order in which the nodes are chosen in iterations. This rule is based on the *sweep implementation* of the ϵ-relaxation method, which was introduced in [Ber86a] and was analyzed in more detail in [BeE88], [BeT89], and [BC91] for the linear cost network flow problem. All the nodes are kept in a linked list T, which is traversed from the first to the last element. The order of the nodes in the list is consistent with the successor order implied by the admissible graph, that is, if a node j is a successor of a node i, then j must appear after i in the list. If the initial admissible graph is empty, as is the case with the initialization of Eq. (6), the initial list is arbitrary. Otherwise, the initial list must be consistent with the successor order of the initial admissible graph. The list is updated in a way that maintains the consistency with the successor order. In particular, let i be a node on which we perform an ϵ-relaxation iteration, and let N_i be the subset of nodes of T that are after i in T. If the price of i changes, then node i is removed from its position in T and placed in the first position of T. The next node chosen for iteration, if N_i is nonempty, is the node $i' \in N_i$ with positive surplus which ranks highest in T. Otherwise, the positive surplus node ranking highest in T is picked. It can be shown (see the references cited earlier) that with this rule of repositioning nodes following a price change, the list order is consistent with the successor order implied by the admissible graph throughout the method.

A *sweep cycle* is a set of iterations whereby all nodes are chosen once from the list T and an ϵ-relaxation iteration is performed on those nodes that have positive surplus. The idea of the sweep implementation is that an ϵ-relaxation iteration at a node i that has predecessors with positive surplus may be wasteful, since the surplus of i will be set to zero and become positive again through a flow push at a predecessor node.

Our complexity analysis follows the line of the corresponding analysis for the linear cost problem. First we have a proposition that estimates the number of sweep cycles required for termination.

Proposition 7: Assume that for some integer $K \geq 1$, the initial price vector p^0 for the sweep implementation of the ϵ-relaxation method satisfies $K\epsilon$-CS together with some feasible flow vector x^0. Then, the number of sweep cycles up to termination is $O(KN^2)$.

Proof: Consider the start of any sweep cycle. Let N^+ be the set of nodes with positive surplus that have no predecessor with positive surplus; let N^0 be the set of nodes with nonpositive surplus that have no predecessor with positive surplus. Then, as long as no price change takes place during the cycle, all nodes in N^0 remain in N^0, and an iteration on a node $i \in N^+$ moves i from N^+ to N^0. So if no node changed price during the cycle, then all nodes in N^+ will be moved to N^0 and the method terminates. Therefore, there is a price change in every cycle except possibly the last one. Since by Prop. 3 there are $O(KN^2)$ price changes, the result follows. **Q.E.D.**

By using Prop. 7, we now bound the running time for the sweep implementation of the ϵ-relaxation method. The dominant computational requirements are:

(1) The computation required for price increases.

(2) The computation required for saturating δ-flow pushes.

(3) The computation required for nonsaturating δ-flow pushes.

Proposition 8: Assume that for some $K \geq 1$ the initial price vector p^0 for the sweep implementation of the ϵ-relaxation method satisfies $K\epsilon$-CS together with some feasible flow vector x^0. Then, the method requires $O(KN^3)$ operations up to termination.

Proof: According to Prop. 3, there are $O(KN)$ price increases per node, so the requirements for (1) above are $O(KNA)$ operations. Furthermore, whenever a flow push is saturating, it takes at least one price increase at one of the end nodes before the flow on that arc can be changed again. Thus the total requirement for (2) above is $O(KNA)$ operations also. Finally, for (3) above we note that for each sweep cycle there can be only one nonsaturating δ-flow push per node. Thus a time bound for (3) is $O(N \cdot \text{total number of sweep cycles})$ which, by Prop. 7, is $O(KN^3)$ operations. Adding the computational requirements for (1), (2), and (3), and using the fact $A \leq N^2$, the result follows. **Q.E.D.**

It is well known that the theoretical and the practical performance of the ϵ-relaxation method can be improved by scaling. One possibility is *cost scaling* (see [BlJ92], [EdK72], [Roc80]). An analysis of cost scaling applied to ϵ-relaxation for the linear network flow problem is given in [BeE87] and also in [BeE88]. In the convex cost case, however, cost scaling may be difficult to implement since the arc cost functions may be unbounded. A second scaling approach in connection with the ϵ-relaxation method for linear cost problems, is *ϵ-scaling.* This approach was originally introduced in [Ber79] as a means of improving the performance of the auction algorithm for the assignment problem. Its complexity analysis was given in [Gol87] and [GoT90].

The key idea of ϵ-scaling is to apply the ϵ-relaxation method several times, starting with a large value of ϵ and to successively reduce ϵ up to a final value that will give the desirable degree of accuracy to our solution. Furthermore, the price and flow information from one application of the method is transferred to the next.

The procedure is as follows: First we choose a scalar $\theta \in (0,1)$, a price vector p^0, and a desirable value $\bar{\epsilon}$ for ϵ on termination. Next we choose a sufficiently large ϵ^0 so that p^0 satisfies ϵ^0-CS with some feasible flow vector x^0. Then, for $k = 1, 2, \ldots$, we set $\epsilon^k = \theta\epsilon^{k-1}$ and, for $k = 1, 2, \ldots \bar{k}$, we apply the ϵ-relaxation method with $\epsilon = \epsilon^{k-1}$, where $\bar{k}$ is the first positive integer k for which ϵ^{k-1} is below $\bar{\epsilon}$. Let (x^k, p^k) be the flow-price vector pair obtained at the kth application of the method, for $k = 1, 2, ..., \bar{k}$. Then x^k is feasible and satisfies ϵ^{k-1}-CS with p^k. Furthermore, the admissible graph after the kth application of the method is acyclic. The initial price vector for the $(k+1)$st application is p^k and the initial flow is x^k_{ij} for the arcs (i,j) that satisfy ϵ^k-CS with p^k, and

$$\sup\left\{\xi \mid f^+_{ij}(\xi) \le p^k_i - p^k_j - \epsilon^k/2\right\}$$

otherwise. This choice of initial flows does not introduce any new arcs to the admissible graph, so the initial admissible graph for the $(k+1)$st application of the method is acyclic. For the 1st application of the method, the initial price vector is p^0 and the initial flow vector is chosen so that the initial admissible graph is acyclic.

We observe that, for the $(k+1)$st application of the method $(k = 0, 1, \ldots \bar{k}-1)$, the initial price vector p^k satisfies ϵ^k/θ-CS with the feasible flow vector x^k. Thus, based on Prop. 8, we conclude that the $(k+1)$st application of the method has a running time of $O\big(\lceil 1/\theta\rceil N^3\big)$, which is $O(N^3)$ since θ is a fixed scalar. The method will be applied at most $\bar{k} = \lceil \log_\theta(\epsilon^0/\bar{\epsilon})\rceil$ times. We have thus obtained the following:

Proposition 9: The running time of the ϵ-relaxation method using the sweep implementation and ϵ-scaling as described above is $O\big(N^3 \ln(\epsilon^0/\bar{\epsilon})\big)$ operations.

We note that a complexity bound of $O\big(NA\ln(N)\ln(\epsilon^0/\bar{\epsilon})\big)$ operations was derived in [KaM93] for the tighten and cancel method. For relatively dense problems where $A = \Theta(N^2/\ln N)$, our complexity bound for the ϵ-relaxation method is more favorable, while for sparse problems, where $A = \Theta(N)$, the reverse is true.

5. THE REVERSE AND FORWARD-REVERSE ϵ-RELAXATION METHODS

The ϵ-relaxation method we presented in the previous sections performed iterations only on nodes with positive surplus. We will refer to it as the *forward* method. We can also define a method, namely the *reverse* method, which performs iterations on nodes of negative surplus. This involves a simple reformulation of the flow and price changing operations we introduced in previous sections for the forward method. The reverse ϵ-relaxation method is the "mirror image" of the forward method that we developed in the previous sections. Naturally, it has similar properties to the forward method and its validity follows from a similar analysis.

It is possible to combine the forward and the reverse method so that the resulting method will operate on both positive and negative surplus nodes. Our intuition is that if we perform ϵ-relaxation iterations on both sources and sinks, we will be able to find the optimal solution faster for certain classes of problems. We refer to the resulting method as the *forward-reverse method*. We initialize the arc flows and node prices in the same way we initialized them for the forward and the reverse methods so that the initial admissible graph is acyclic. The forward-reverse method operates as follows:

Typical Iteration of the Forward-Reverse ϵ-Relaxation Method

Pick a node i with nonzero surplus; if no such node exists then terminate. If i has positive surplus then perform an iteration of the forward ϵ-relaxation method. If i has negative surplus then perform an iteration of the reverse ϵ-relaxation method.

The idea of the forward-reverse method is recurrent in many relaxation-like methods. Termination of the method can be proved with an analysis similar to the one in Section 3, provided that we also make the following assumption:

Assumption: The number of times the surplus of a node changes sign is finite.

The above assumption can be enforced to hold by various mechanisms, some of which are discussed in [Tse86] for the relaxation method and in [Pol94] for the auction shortest path algorithm.

6. COMPUTATIONAL RESULTS

We have developed and tested two experimental Fortran codes implementing the methods of this paper for convex cost problems. The first code, named NE-RELAX-F, implements the forward ϵ-relaxation method with the sweep implementation and ϵ-scaling as described in Section 4. The second code, named NE-RELAX-FV, implements the forward-reverse version of NE-RELAX-F as described in Section 5. These codes are based on the ϵ-relaxation code for linear cost problems described in Appendix 7 of [Ber91], which has been shown to be quite efficient. Several changes and enhancements were introduced in the codes for convex cost problems: All computations are done in real rather than integer arithmetic, and ϵ-scaling, rather than arc cost scaling, is used. Also, the updating of the push lists and prices are changed to improve efficiency. Otherwise, the sweep implementation and the general structure of the codes for linear and convex cost problems are identical. Initial testing on linear cost problems showed that the codes for convex cost problems perform as well as, and often better than, their counterparts for linear cost problems, which indicates that these codes are written efficiently. (The superior performance of the codes for convex cost problems may be due to the latter's efficient management of the push lists and the speed of floating point computations of the machine on which the codes were run.)

The codes NE-RELAX-F and NE-RELAX-FV were compared to two existing Fortran codes NRELAX and MNRELAX from [BHT87]. The latter implement the relaxation method for, respectively, strictly convex cost and convex cost problems, and are believed to be quite efficient. All codes were compiled and run on a Sun Sparc-5 workstation with 24 megabytes of RAM under the Solaris operating system. We used the -O compiler option in order to take advantage of the floating point unit and the design characteristics of the Sparc-5 processor. Unless otherwise indicated, all codes terminated according to the same criterion, namely, the cost of the feasible flow vector and the cost of the price vector agree in their first 12 digits.

For our testing, we used convex linear/quadratic problems corresponding to the case of (P) where

$$f_{ij}(x_{ij}) = \begin{cases} a_{ij}x_{ij} + b_{ij}x_{ij}^2 & \text{if } 0 \le x_{ij} \le c_{ij}, \\ \infty & \text{otherwise}, \end{cases}$$

for some a_{ij}, b_{ij}, and c_{ij} with $-\infty < a_{ij} < \infty$, $b_{ij} \ge 0$, and $c_{ij} \ge 0$. We call a_{ij}, b_{ij}, and c_{ij} the linear cost coefficient, the quadratic cost coefficient, and the capacity, respectively, of arc (i,j). We created the test problems using two Fortran problem generators. The first is the public-domain generator NETGEN, written by Klingman, Napier and Stutz [KNS74], which generates linear-cost assignment/transportation/transshipment problems having a certain random structure. The second is the generator CHAINGEN, written by the second author, which generates transshipment problems having a chain structure as follows: starting with a chain through all the nodes, a user-specified number of forward arcs are added to each node (for example, if the user specifies 3 additional arcs per node then the arcs $(i, i+2)$, $(i, i+3)$, $(i, i+4)$ are added for each node i) and, for a user-specified percentage of nodes i, a reverse arc $(i, i-1)$ is also added. The graphs thus created have long diameters and earlier tests on linear cost problems showed that the created problems are particularly difficult for all methods. As the above two generators create only linear cost problems, we modified the created problems as in [BHT87] so that, for a user-specified percent of the arcs, a nonzero quadratic cost coefficient is generated in a user-specified range.

Our tests were designed to study two key issues:

(a) The performance of the ϵ-relaxation methods relative to the relaxation methods, and the dependence of this performance on network topology and problem ill-conditioning.

(b) The sensitivity of the ϵ-relaxation methods to problem ill-conditioning.

Ill-conditioned problems were created by assigning to some of the arcs have much smaller (but nonzero) quadratic cost coefficients compared to other arcs. When the arc cost functions have this structure, ill-conditioning in the traditional sense of unconstrained nonlinear programming tends to occur.

We experimented with three sets of test problems: the first set comprises well-conditioned strictly convex quadratic cost problems generated using NETGEN (Table 1); the second set comprises well-conditioned strictly convex quadratic cost problems generated using CHAINGEN (Table 2); the third set comprises ill-conditioned strictly convex quadratic cost problems and mixed linear/quadratic cost problems generated using NETGEN (Table 3). The running time of the codes on these problems are shown in the last three to four columns of Tables 1–3. In all problems, the ϵ-relaxation codes were run to the point where they yielded higher or comparable solution accuracy than the relaxation codes. From the running times we can draw the following conclusions: First, the ϵ-relaxation codes NE-RELAX-F and NE-RELAX-FV have similar performance and both consistently outperform, by a factor of at least 3 and often much more, the relaxation codes NRELAX and MNRELAX on all test problems, independent of network topology and problem ill-conditioning. In fact, on the CHAINGEN problems, the ϵ-relaxation codes outperform the relaxation codes by an order of magnitude or more. Other than the favorable complexity results that we obtained in this paper, we have no clear explanation of this phenomenon.

REFERENCES

[Ber79] Bertsekas, D. P., “A Distributed Algorithm for the Assignment Problems,” Laboratory for Information and Decision Systems Working Paper, M.I.T., Cambridge, MA, 1979.

[Ber85] Bertsekas, D. P., “A Unified Framework for Minimum Cost Network Flow Problems,” Mathematical Programming, Vol. 32, 1985, pp. 125-145.

[Ber86a] Bertsekas, D. P., “Distributed Relaxation Methods for Linear Network Flow Problems,” Proceedings of 25th IEEE Conference on Decision and Control, 1986, pp. 2101-2106.

[Ber86b] Bertsekas, D. P., "Distributed Asynchronous Relaxation Methods for Linear Network Flow Problems," Laboratory for Information and Decision Systems Report P-1606, M.I.T., Cambridge, MA, 1986.

[Ber91] Bertsekas, D. P., Linear Network Optimization: Algorithms and Codes, M.I.T. Press, Cambridge, MA, 1991.

[Ber92] Bertsekas, D. P., "An Auction/Sequential Shortest Path Algorithm for the Min Cost Flow Problem," Laboratory for Information and Decision Systems Report P-2146, M.I.T., Cambridge, MA, 1992.

[BC91] Bertsekas, D. P., Castanon, D. A. "A Generic Auction Algorithm for the Minimum Cost Network Flow Problem," Laboratory for Information and Decision Systems Report LIDS-P-2084, M.I.T., Cambridge, MA, 1991, Compuatational Optimization and Applications, 1994.

[BCE95] Bertsekas, D. P., Castanon, D., Eckstein, J., and Zenios, S. A., "Parallel network optimization survey", to appear in Encyclopedia of OR.

[BeE87] Bertsekas, D. P., and Eckstein, J., "Distributed Asynchronous Relaxation Methods for Linear Network Flow Problems," Proceedings of IFAC '87, Munich, Germany, July 1987.

[BeE88] Bertsekas, D. P., and Eckstein, J., "Dual Coordinate Step Methods for Linear Network Flow Problems," Mathematical Programming, Vol. 42, 1988, pp. 203-243.

[BeE87] Bertsekas, D. P., and El Baz, D., "Distributed Asynchronous Relaxation Methods for Convex Network Flow Problems," SIAM Journal on Control and Optimization, Vol. 25, 1987, pp. 74-85.

[BHT87] Bertsekas, D. P., Hosein, P. A., and Tseng, P., "Relaxation Methods for Network Flow Problems with Convex Arc Costs," SIAM Journal on Control and Optimization, Vol. 25, 1987, pp. 1219-1243.

[BeT88] Bertsekas, D. P., and Tseng, P., "Relaxation Methods for Minimum Cost Ordinary and Generalized Network Flow Problems," Operations Research, Vol. 36, 1988, pp. 93-114.

[BeT94] Bertsekas, D. P., and Tseng, P., " RELAX-IV: A Faster Version of the RELAX Code for Solving Minimum Cost Flow Problems," Laboratory for Information and Decision Systems Report P-2276, M.I.T., Cambridge, MA, 1994.

[BeT89] Bertsekas, D. P., and Tsitsiklis, J. N., Parallel and Distributed Computation: Numerical Methods, Prentice-Hall, Englewood Cliffs, NJ, 1989.

[BlJ92] Bland, R. G., and Jensen, D. L., "On the Computational Behavior of a Polynomial-Time Network Flow Algorithm," Mathematical Programming, Vol. 54, 1992, pp. 1-39.

[DMZ95] De Leone, R., Meyer, R. R., and Zakarian, A., "An ϵ-Relaxation Algorithm for Convex Network Flow Problems," Computer Sciences Department Technical Report, University of Wisconsin, Madison, WI, 1995.

[EdK72] Edmonds, J., and Karp, R. M., "Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems," Journal of the ACM, Vol. 19, 1972, pp. 248-264.

[FoF62] Ford, L. R., Jr., and Fulkerson, D. R., Flows in Networks, Princeton University Press, Princeton, NJ, 1962

[GoT90] Goldberg, A. V., and Tarjan, R. E., "Solving Minimum Cost Flow Problems by Successive Approximation," Mathematics of Operations Research, Vol. 15, 1990, pp. 430-466.

[Gol87] Goldberg, A. V., "Efficient Graph Algorithms for Sequential and Parallel Computers," Laboratory for Computer Science Technical Report TR-374, M.I.T., Cambridge, MA, 1987.

[Hag92] Hager, W. W., "The Dual Active Set Algorithm," in Advances in Optimization and Parallel Computing, Edited by P. M. Pardalos, North-Holland, Amsterdam, Netherland, 1992, pp. 137-142.

[HaH93] Hager, W. W., and Hearn, D. W., "Application of the Dual Active Set Algorithm to Quadratic Network Optimization," Computational Optimization and Applications, Vol. 1, 1993, pp. 349-373.

[KaM84] Kamesam, P. V., and Meyer, R. R., "Multipoint Methods for Separable Nonlinear Networks," Mathematical Programming Study, Vol. 22, 1984, pp. 185-205.

[KaM93] Karzanov, A. V., and McCormick, S. T., "Polynomial Methods for Separable Convex Optimization in Unimodular Linear Spaces with Applications to Circulations and Co-circulations in Network," Faculty of Commerce Report, University of British Columbia, Vancouver, BC, 1993; to appear in SIAM Journal on Computing.

[KNS74] Klingman, D., Napier, A., and Stutz, J., "NETGEN - A Program for Generating Large Scale (Un) Capacitated Assignment, Transportation, and Minimum Cost Flow Network Problems," Management Science, Vol. 20, 1974, pp. 814-822.

[LiZ91] Li, X., and Zenios, S. A., "Data Parallel Solutions of Min-Cost Network Flow Problems Using ϵ-Relaxations," Department of Decision Sciences Report 1991-05-20, University of

Pennsylvania, Philadelphia, PA, 1991.

[Mey79] Meyer, R. R., "Two-Segment Separable Programming," Management Science, Vol. 25, 1979, pp. 285-295.

[NiZ93] Nielsen, S. S., and Zenios, S. A., "On the Massively Parallel Solution of Linear Network Flow Problems," in Network Flow and Matching: First DIMACS Implementation Challenge, Edited by D. Johnson and C. McGeoch, American Mathematical Society, Providence, RI, 1993, pp. 349-369.

[Pol94] Polymenakos, L. C. "Parallel Shortest Path Auction Algorithms," Parallel Computing, Vol. 20, 1994, pp. 1221-1247.

[Pol95] Polymenakos, L. C. "ϵ-Relaxation and Auction Algorithms for the Convex Cost Network Flow Problem," Electrical Engineering and Computer Science Department Ph.D. Thesis, M.I.T., Cambridge, MA, 1995.

[Roc80] Röck, H., "Scaling Techniques for Minimal Cost Network Flows," in Discrete Structures and Algorithms, Edited by U. Pape, Carl Hanser, München, Germany, 1980, pp. 181-191.

[Roc70] Rockafellar, R. T., Convex Analysis, Princeton University Press, Princeton, NJ, 1970.

[Roc84] Rockafellar, R. T., Network Flows and Monotropic Programming, Wiley-Interscience, New York, NY, 1984.

[Tse86] Tseng, P., "Relaxation Methods for Monotropic Programming Problems," Operations Research Center Ph.D. Thesis, M.I.T., Cambridge, MA, 1986.

[TBT90] Tseng, P., Bertsekas, D. P., and Tsitsiklis, J. N., "Partially Asynchronous, Parallel Algorithms for Network Flow and Other Problems," SIAM Journal on Control and Optimization, Vol. 28, 1990, pp. 678-710.

[Ven91] Ventura, J. A., "Computational Development of a Lagrangian Dual Approach for Quadratic Networks," Networks, Vol. 21, 1991, pp. 469-485.

[Wei74] Weintraub, A., "A Primal Algorithm to Solve Network Flow Problems with Convex Costs," Management Science, Vol. 21, 1974, pp. 87-97.