October, 1983 LIDS-P-1339

# RELAXATION METHODS FOR MINIMUM COST NETWORK FLOW PROBLEMS†

by

Dimitri P. Bertsekas*

and

Paul Tseng*

## ABSTRACT

We view the optimal single commodity network flow problem with linear arc costs and its dual as a pair of monotropic programming problems, i.e. problems of minimizing the separable sum of scalar extended real-valued convex functions over a subspace. For such problems directions of cost improvement can be selected from among a finite set of directions--the elementary vectors of the constraint subspace. The classical primal simplex, dual simplex, and primal-dual methods turn out to be particular implementations of this idea. This paper considers alternate implementations leading to new dual descent algorithms which are conceptually related to coordinate descent and Gauss-Seidel relaxation methods for unconstrained optimization or solution of equations. Contrary to primal simplex and primal-dual methods, these algorithms admit a natural extension to network problems with strictly convex arc costs.

Our first coded implementation of relaxation methods is compared with mature state-of-the-art primal simplex and primal-dual codes and is found to be substantially faster on most types of network flow problems of practical interest.

---

†This work has been supported by the National Science Foundation under Grant NSF-ECS-8217668. Many thanks are due to Tom Magnanti who supplied us with the primal-dual code KILTER and to Michael Grigoriadis who supplied us with the primal simplex code RNET for the purposes of comparative testing. They also, together with John Mulvey and Bob Gallager, clarified several questions for us. Support provided by Alphatech, Inc. is also gratefully acknowledged.

**The authors are with the Laboratory for Information and Decision Systems and the Operations Research Center, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139.

## 1. Introduction

Consider a directed graph with set of nodes $N$ and set of arcs $A$. Each arc $(i,j)$ has associated with it an integer $a_{ij}$ referred to as the cost of $(i,j)$. We denote by $f_{ij}$ the flow of the arc $(i,j)$ and consider the classical minimum cost flow problem

$$\text{minimize} \quad \sum_{(i,j)\in A} a_{ij} f_{ij} \tag{MCF}$$

subject to

$$\sum_{\substack{m \\ (m,i)\in A}} f_{mi} - \sum_{\substack{m \\ (i,m)\in A}} f_{im} = 0, \quad \forall\, i\in N \quad \text{(Conservation of Flow)} \tag{1}$$

$$\ell_{ij} \leq f_{ij} \leq c_{ij}, \quad \forall\, (i,j)\in A \quad \text{(Capacity constraint)} \tag{2}$$

where $\ell_{ij}$ and $c_{ij}$ are given integers. We assume throughout that there exists at least one feasible solution of (MCF). We formulate a dual problem to (MCF).

We associate a Lagrange multipler $p_i$ (referred to as the price of node i) with the ith conservation of flow constraint (1). By denoting by f and p the vectors with elements $f_{ij}$, $(i,j)\in A$ and $p_i$, $i\in N$ respectively, we can write the corresponding Lagrangian function

$$L(f,p) = \sum_{(i,j)\in A} (a_{ij}+p_j-p_i) f_{ij}.$$

The dual problem is

$$\text{maximize} \quad q(p) \tag{3}$$

subject to no constraints on p,

where the dual functional $q$ is given by

$$q(p) = \min_{\ell_{ij} \le f_{ij} \le c_{ij}} L(f,p) \tag{4}$$

$$= \sum_{(i,j)\in A} \min_{\ell_{ij} \le f_{ij} \le c_{ij}} \{(a_{ij}+p_j-p_i)f_{ij}\}.$$

Given any price vector p we consider the corresponding tension vector t having elements $t_{ij}$, $(i,j)\in S$ defined by

$$t_{ij} = p_i - p_j, \qquad \forall\ (i,j)\in S. \tag{5}$$

Since the dual functional as well as subsequent definitions, optimality conditions and algorithms depend on the price vector p only through the corresponding tension vector t we will often make no distinction between p and t in what follows.

For any price vector p we say that an arc (i,j) is:

$$\text{Inactive} \quad \text{if} \quad t_{ij} < a_{ij} \tag{6}$$

$$\text{Balanced} \quad \text{if} \quad t_{ij} = a_{ij} \tag{7}$$

$$\text{Active} \quad \text{if} \quad t_{ij} > a_{ij}. \tag{8}$$

For any flow vector f the scalar

$$d_i = \sum_{\substack{m \\ (i,m)\in A}} f_{im} - \sum_{\substack{m \\ (m,i)\in A}} f_{mi} \qquad \forall\, i\in N \tag{9}$$

will be referred to as the deficit of node i. It represents the difference of total flow exported and total flow imported by the node.

The optimality conditions in connection with (MCF) and its dual given by (3), (4) state that (f,p) is a primal and dual optimal solution pair if and only if

$$f_{ij} = \ell_{ij} \qquad \text{for all inactive arcs } (i,j) \tag{10}$$

$$\ell_{ij} \le f_{ij} \le c_{ij} \qquad \text{for all balanced arcs } (i,j) \tag{11}$$

$$f_{ij} = c_{ij} \qquad \text{for all active arcs } (i,j) \tag{12}$$

$$d_i = 0 \qquad \text{for all nodes i.} \tag{13}$$

Conditions (10)-(12) are the complementary slackness conditions.

In what follows in this section we describe how (MCF) and the algorithms of this paper can be viewed within the context of monotropic programming theory [2].

We can rewrite (MCF) as

$$\text{minimize} \quad \sum_{(i,j)\in A} g_{ij}(f_{ij}) \tag{P}$$

$$\text{subject to } f\in C$$

where $g_{ij}: R \to (-\infty,+\infty]$ is the convex function

$$g_{ij}(f_{ij}) = \begin{cases} a_{ij}f_{ij} & \text{if } \ell_{ij} \le f_{ij} \le c_{ij} \\ +\infty & \text{otherwise} \end{cases} \tag{14}$$

and $C$ is the circulation subspace of the network

$$C = \{f \mid \sum_{(m,i)\in A} f_{mi} - \sum_{(i,m)\in A} f_{im} = 0, \quad \forall\, i\in N\}. \tag{15}$$

From (5) we see that the dual functional $q(p)$ can be written explicitly as

$$q(p) = - \sum_{(i,j)\in A} g^*_{ij}(p_i - p_j)$$

where the convex, piecewise linear functions $g^*_{ij}$ are given by

$$g^*_{ij}(t_{ij}) = \begin{cases} (t_{ij}-a_{ij})c_{ij} & \text{if } t_{ij} \geq a_{ij} \\ (t_{ij}-a_{ij})\ell_{ij} & \text{if } t_{ij} \leq a_{ij} \end{cases} \tag{16}$$

(see Figure 1). As suggested by the notation, $g^*_{ij}$ is actually the conjugate convex function of $g_{ij}$ (in the usual sense of convex analysis [1])

$$g^*_{ij}(t_{ij}) = \sup_{f_{ij}} \{t_{ij}\, f_{ij} - g_{ij}(f_{ij})\}, \qquad \forall\, (i,j)\in A\ ,$$

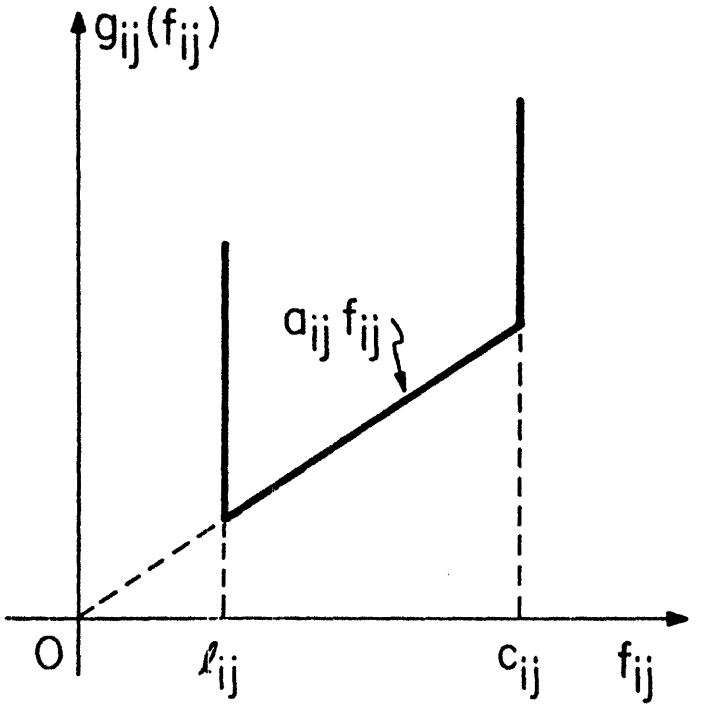
as the reader can easily verify.

We can now write the dual problem (3) in a form which is symmetric to (P)

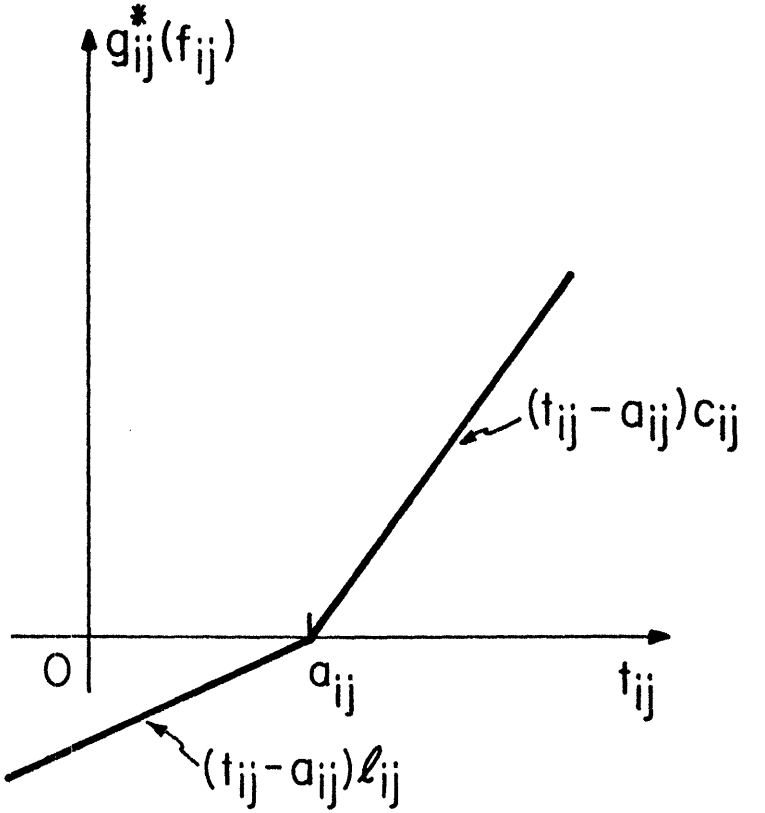
$$\text{minimize} \quad \sum_{(i,j)\in A} g^*_{ij}(t_{ij}) \tag{D}$$

$$\text{subject to } t\in C^{\perp}$$

where $C^{\perp}$ is the subspace




Primal Cost Function




Dual Cost Function

Figure 1

$$C^{\perp} = \{t \mid \text{for some price vector } p,\ t_{ij} = p_i - p_j,\ \forall\ (i,j)\in A\}. \qquad (17)$$

Our notation is justified since it can be easily seen that $C^{\perp}$ is the orthogonal complement of $C$. (For each i multiply (1) with $p_i$ and add over $i\in N$ to obtain that the inner product of any $f\in C$ and $t\in C^{\perp}$ is zero).

Problems (P) and (D) constitute a pair of dual monotropic programming problems as introduced in Rockafellar [2]. It was shown there in a more general setting that these programs have the same optimal value and their solutions are related via the conditions (10)-(13). An important special property of these programs is that at each nonoptimal point it is possible to find descent directions among a finite set of directions--the elementary vectors of the subspace $C$ [in the case of (P)] or the subspace $C^{\perp}$ [in the case of (D)]. The notion of an elementary vector of a subspace was introduced in [3] (see also [1], p. 205) where it was defined as a vector in the subspace having minimal support (i.e. a minimal number of nonzero coordinates). The same references give a characterization of the elementary vectors of $C$ and $C^{\perp}$.

The elementary vectors of $C$ are associated on a one-to-one basis (modulo scalar multiplication) with simple cycles of the network as follows:

Let Y be any undirected simple cycle of the network and arbitrarily define one of the directions of traversing the cycle as the positive direction. The elementary vector $\{f_{ij} \mid (i,j)\in A\}$ corresponding to Y is given by

$$f_{ij} = \begin{cases} 1 & \text{if } (i,j)\in Y \text{ and is oriented in the positive direction} \\ -1 & \text{if } (i,j)\in Y \text{ and is oriented in the negative direction} \\ 0 & \text{otherwise.} \end{cases}$$

The elementary vectors of $C^{\perp}$ are associated (modulo scalar multiplication) with cutsets of the network as follows ([1], p. 205):

Let S be a nonempty strict subset of $N$ such that when the set of arcs $\{(i,j)\,|\,i\in S,\ j\notin S\}\cup\{(i,j)\,|\,i\notin S,\ j\in S\}$ is removed the network separates into exactly two connected components. The elementary vector $\{v_{ij}\,|\,(i,j)\in A\}$ corresponding to S is given by

$$v_{ij} = \begin{cases} 1 & \text{if } i\notin S,\ j\in S \\ -1 & \text{if } i\in S,\ j\notin S \\ 0 & \text{otherwise.} \end{cases} \tag{18}$$

In view of the fact that at any nonoptimal vector for either (P) or (D) it is possible to find an elementary vector of either C or $C^{\perp}$ respectively that is a direction of descent, it is natural to consider algorithms that make cost improvements at each iteration by moving along elementary vectors of descent. This philosophy when applied to (P) yields simplex-like algorithms that proceed by making flow corrections along simple cycles. When applied to (D) this philosophy yields as special cases the classical dual simplex and primal-dual methods which proceed by making price corrections across cutsets constructed in special ways. However it yields also several other new methods which are the subject of the present paper.

In the next section we characterize the descent properties of elementary vectors for problem (D) and show that the price changes made

by the classical primal-dual method are along an elementary vector of maximal rate of descent.

In Section 3 we introduce a class of algorithms for solving (D) where price changes are again made along elementary vectors. Generally these changes lead to a cost improvement but there is an important exception where the cost remains constant. A key fact which has been substantiated only through experiment is that the elementary descent directions used by these algorithms lead to comparable improvements per iteration as the direction of maximal rate of descent (the one used by the classical primal-dual method), but are computed with considerably less computational overhead.

In Section 4 we provide the natural extension of the linear cost algorithm to problems with strictly convex, differentiable arc costs. We show that the algorithm reduces to the classical coordinate descent (or relaxation) method for solving an associated continuously differentiable dual problem analogous to (D).

Sections 5 through 7 are devoted to computational comparisons of our own experimental computer codes based on relaxation methods with mature state-of-the-art codes based on the primal simplex and primal-dual methods. One main conclusion is that relaxation outperforms by a large margin the primal-dual method. Comparisons with primal simplex are less clear but the emerging conclusions are that relaxation is competitive with primal-simplex for uncapacitated or lightly capacitated problems and substantially outperforms it for assignment, transportation, heavily capacitated, and piecewise linear

cost problems.

The performance of a specialized relaxation algorithm for assignment problems first given in [5] is apparently far better than any other method known to us.

The linear arc cost algorithms of this paper are closely related to the class of algorithms in Bertsekas [6] where the conceptual similarity with relaxation methods was pointed out. The present paper emphasizes the dual descent viewpoint and provides computational results. A special case for the assignment problem has been considered in Bertsekas [5]. The relaxation algorithm for differentiable strictly convex arc costs is a special case of a multicommodity flow algorithm due to Stern [10]. However its interpretation in terms of duality and coordinate descent methods (cf. Proposition 2) is new.

## 2. Characterization of Descent Directions

Consider a connected strict subset S of $N$ and any vector $t \in C^{\perp}$. Define

$$C(S,t) = \sum_{(i,j)\in A} e_{ij}(S,t_{ij}) \tag{19}$$

where for all $(i,j)\in A$

$$e_{ij}(S,t_{ij}) = \begin{cases} \ell_{ij} & \text{if } i\in S,\ j\notin S, \text{ and } (i,j) \text{ is inactive or balanced} \\ -\ell_{ij} & \text{if } i\notin S,\ j\in S, \text{ and } (i,j) \text{ is inactive} \\ c_{ij} & \text{if } i\in S,\ j\notin S, \text{ and } (i,j) \text{ is active} \\ -c_{ij} & \text{if } i\notin S,\ j\in S, \text{ and } (i,j) \text{ is active or balanced} \\ 0 & \text{otherwise} \end{cases} \tag{20}$$

In words C(S,t) is the difference of outflow and inflow across S when the flows of inactive and active arcs are set at their lower and upper bounds respectively, while the flow of each balanced arc incident to S is set to its lower or upper bound depending on whether the arc is going out of S or coming into S respectively.

The following proposition shows that -C(S,t) is the directional derivative of the dual cost along the elementary vector corresponding to S [cf. (18)].†

Proposition 1: For every nonempty strict subset S of $N$ and every tension vector $t\in C^{\perp}$ there holds

$$w(t+\gamma v) = w(t) - \gamma C(S,t), \quad \forall\, \gamma\in[0,\delta) \tag{21}$$

†The vector v of (12) will not necessarily be an elementary vector of $C$ cause the cutset corresponding to S may separate the network into more than two connected components. However this difficulty can be bypassed by working with an augmented network involving a "supernode" s and an arc (s,i) for each $i\in N$ having cost coefficient and upper and lower capacity bounds equal to zero. Then every connected subset S of defines an elementary vector for the augmented network via (18).

where $w(\cdot)$ is the dual cost function

$$w(t) = \sum_{(i,j)\in A} g^*_{ij}(t_{ij}), \tag{22}$$

v is the elementary vector corresponding to S

$$v_{ij} = \begin{cases} 1 & \text{if } i\notin S,\ j\in S \\ -1 & \text{if } i\in S,\ j\notin S \\ 0 & \text{otherwise} \end{cases} \tag{23}$$

and $\delta$ is given by

$$\delta = \inf\{\{t_{im}-a_{im} \mid i\in S,\ m\notin S,\ (i,m)\text{: active}\}, \tag{24}$$

$$\{a_{mi}-t_{mi} \mid i\in S,\ m\notin S,\ (m,i)\text{: inactive}\}\}.$$

(We use the convention $\delta = +\infty$ if the set over which the infimum above is taken is empty.)

Proof: It is easily verified using (16) that the rate of change of each function $g^*_{ij}(t_{ij})$ as t is changed along the direction v of (23) is $-e_{ij}(S,t)$ as given by (20). Therefore using (22) we see that the corresponding rate of change of the dual cost w is $-C(S,t)$. Since w is piecewise linear the actual change of w along the direction v is linear in the stepsize $\gamma$ up to the point where $\gamma$ becomes large enough so that the pair $[w(t+\gamma v),\ t+\gamma v]$ meets a new face of the graph of w. This value of $\gamma$ is the one for which a new arc incident to S becomes balanced and equals the scalar $\delta$ of (24). Q.E.D.

We now describe the following classical primal-dual algorithm which as we will see solves (D) by making price changes along elementary descent directions. We first recall the definition of an augmenting path (e.g. [4], [7]-[9]).

Definition 1: Given a pair (f,t) satisfying the complementary slackness conditions (10)-(12) we say that a sequence of nodes $\{n_1,n_2,\ldots,n_k\}$ is an augmenting path if $d_{n_1} < 0$, $d_{n_k} > 0$ and, for m = 1,2,...,k-1, either there exists a balanced arc $(n_m,n_{m+1})$ with $f_{n_m n_{m+1}} < c_{n_m n_{m+1}}$, or there exists a balanced arc $(n_{m+1},n_m)$ with $f_{n_{m+1} n_m} > \ell_{n_{m+1} n_m}$. The scalar

$$\varepsilon = \min \{-d_{n_1}, d_{n_k}, \varepsilon_1,\ldots,\varepsilon_{k-1}\} \tag{25}$$

where

$$\varepsilon_m = \begin{cases} c_{n_m n_{m+1}} - f_{n_m n_{m+1}} & \text{if } (n_m,n_{m+1}) \text{ is the arc of the path} \\ f_{n_{m+1} n_m} - \ell_{n_{m+1} n_m} & \text{if } (n_{m+1},n_m) \text{ is the arc of the path} \end{cases} \tag{26}$$

will be called the capacity of the path.

In the following algorithm at any time each node can be in one of the three possible states unlabeled, labeled and unscanned, or labeled and scanned. This is a standard device that is used here in the same manner as in many other sources [4], [7]-[9].

Classical Primal-Dual Algorithm:

Step 0: Select a pair (f,t) satisfying the complementary slackness

conditions (10)-(12).

Step 1: Discard all existing labels. Give the label "0" to all nodes i with $d_i > 0$. If $d_i = 0$ for all $i \in N$, terminate.

Step 2: Choose a labeled but unscanned node i and go to step 3. If no such node can be found go to step 5.

Step 3: Scan the label of the node i by giving the label "i" to all unlabeled nodes m such that (m,i) is balanced and $f_{mi} < c_{mi}$, and to all unlabeled nodes m such that (i,m) is balanced and $\ell_{im} < f_{im}$. If for any of those nodes m we have $d_m < 0$ go to step 4. Else go to step 2.

Step 4 (Flow Augmentation): An augmenting path has been found which starts at the node with $d_m < 0$ identified in step 3 and ends at a node i with $d_i > 0$. The path can be constructed by tracing labels starting from m. Let $\varepsilon > 0$ be the capacity of the path. Increase by $\varepsilon$ the flow of all arcs on the path that are oriented in the direction from m to i, reduce by $\varepsilon$ the flow of all other arcs on the path and go to step 1.

Step 5 (Price Adjustment): Let L be the set of all labeled nodes. (Because all nodes in L have nonnegative deficit, and at least one of these has positive deficit, there must exist a node with negative deficit that is not in L and as a result L is a strict subset of N). Let

$$\delta = \min\{\{t_{km}-a_{km} \mid k\in L,\ m\notin L,\ (k,m):\ \text{active}\},$$
$$\{a_{mk}-t_{mk} \mid k\in L,\ m\notin L,\ (m,k):\ \text{inactive}\}\}. \qquad (27)$$

Set

$$t_{ij} \leftarrow \begin{cases} t_{ij}+\delta & \text{if } i\notin L,\ j\in L \\ t_{ij}-\delta & \text{if } i\in L,\ j\notin L \\ t_{ij} & \text{otherwise,} \end{cases}$$

give the label "k" to each node $m \notin L$ such that the arc (k,m) or (m,k) attains the minimum in (27) and go to step 2.

The computation required in the method above between augmentations is bounded by $O(|N||A|)$ where $|N|$ and $|A|$ are the cardinalities of $N$ and $A$ respectively. (Step 5 requires $O(|A|)$ computation and will be carried out at most $|N|-1$ times before an augmenting path is found in Step 3.) It is possible to implement the method so that the computation between augmentations is bounded by $O(|N|^2)$. This implementation is discussed in the appendix, and if all arc costs are nonnegative the method can also be implemented as a sequence of shortest path computations followed by solution of a max-flow problem (see e.g. [9], p. 141).

The properties of the primal-dual algorithm are well known and will only be summarized. The flow and price changes in steps 4 and 5 respectively are such that at all times the pair (f,t) maintained by the algorithm satisfies the complementary slackness conditions (10)-(12). Each time a flow augmentation occurs in step 4 the total absolute deficit $\sum_{i \in N} |d_i|$ is reduced by $2\varepsilon$ and all prices remain unaffected. Price changes occur at step 5 along the elementary vector corresponding to the set L. The amount of dual cost reduction is (Proposition 1)

$$\delta C(L,t).$$

It is easily verified that $\delta$ is well defined as a positive number.

Notice that, by construction of the set L, each balanced arc connecting L and its complement carries flow at its lower bound or upper bound depending on whether the arc is oriented out from L or into L

respectively. Therefore we have

$$C(L,t) = \sum_{i \in L} d_i > 0.$$

It follows that the cost reduction $\delta C(L,t)$ occurring at step 5 is strictly positive. Now it is easily seen from (19) and (20) that for any t and any strict subset S of $N$ we have

$$C(S,t) \leq \sum_{i \in S} d_i.$$

Therefore, since L contains all the nodes with positive deficit and no nodes with negative deficit, we see that C(L,t) maximizes C(S,t) over all sets S. In other words, among all possible cutsets, the one used for price adjustment in step 5 of the primal-dual method is associated with maximal rate of descent.

The fact that the primal-dual descent direction is optimal in the sense described above by no means implies that the primal-dual method is best among methods that make price changes along elementary vectors. What is really important is the ratio of dual cost reduction over the amount of computation needed for this reduction.

This ratio is roughly proportional to

$$\frac{\delta C(L,t)}{\eta_L}$$

where $\eta_L$ is the number of nodes labeled. Experimental evidence suggests that the methods to be presented in the next section have for many classes of problems a more favorable ratio primarily because they compute elementary

descent directions by labeling fewer nodes than the primal-dual method. This more than offsets the fact that the rate of decrease of dual cost is not as large as that associated with the primal-dual method.

Finally we note that the dual simplex method can also be viewed as a process of iterative price adjustment along elementary descent directions. A proof of this may be found in [11] which also provides yet another implementation of the primal-dual method.

## 3. Relaxation Methods

The following algorithm bears some similarity with the primal-dual method. It maintains complementary slackness at all times. At each iteration it starts from a single node with nonzero deficit and checks if by changing its price it is possible to reduce the value of the dual cost. If not it gradually builds up either an augmenting path or a cutset associated with an elementary direction of descent. The main difference from the primal-dual method is that instead of continuing the process until an elementary direction of maximal rate of descent is found, we stop at the first possible elementary direction of descent--possibly the direction associated with the starting node.

At the beginning of each iteration we have a pair (f,t) satisfying complementary slackness. The iteration determines a new pair (f,t) satisfying complementary slackness by means of the following process.

Typical Relaxation Iteration:

Step 1: Choose a node i with $d_i > 0$. If no such node can be found terminate the algorithm. Else give the label "0" to i, set $S = \emptyset$, and go to step 2.

Step 2: Choose a labeled but unscanned node k, set $S = S \cup \{k\}$, and go to step 3.

Step 3: Scan the label of the node k as follows: Give the label "k" to all unlabeled nodes m such that (m,k) is balanced and $f_{mk} < c_{mk}$, and to all unlabeled m such that (k,m) is balanced and $\ell_{km} < f_{km}$. If

$$C(S,t) > 0 \tag{28}$$

go to step 5. Else if for any of the nodes m labeled from k we have $d_m < 0$ go to step 4. Else go to step 2.

Step 4 (Flow Augmentation): An augmenting path has been found which starts at the node m with $d_m < 0$ identified in step 3 and ends at the node i. The path can be constructed by tracing labels starting from m. Let $\varepsilon > 0$ be the capacity of the path. Increase by $\varepsilon$ the flow of all arcs on the path that are oriented in the direction from m to i, reduce by $\varepsilon$ the flow of all other arcs on the path. Go to the next iteration.

Step 5 (Price Adjustment): Let

$$\delta = \min\{\{t_{km}-a_{km} \mid k\in S,\ m\notin S,\ (k,m):\ \text{active}\},\ \{a_{mk}-t_{mk} \mid k\in S,\ m\notin S,\ (m,k):\ \text{inactive}\}\}. \tag{29}$$

where S is the set of scanned nodes constructed earlier. Set

$$f_{km} \leftarrow \ell_{km}\ ,\ \forall \text{ balanced arcs } (k,m) \text{ with } k\in S,\ m\in L,\ m\notin S$$

$$f_{mk} \leftarrow c_{mk}\ ,\ \forall \text{ balanced arcs } (m,k) \text{ with } k\in S,\ m\in L,\ m\notin S$$

where L is the set of labeled nodes. Set

$$t_{km} \leftarrow \begin{cases} t_{km}+\delta & \text{if} \quad k\notin S,\ m\in S \\ t_{km}-\delta & \text{if} \quad k\in S,\ m\notin S \\ 0 & \text{otherwise} \end{cases}$$

Go to the next iteration.

Note that the change of the tension vector in step 5 is equivalent

to reducing the prices of the nodes in S by an amount $\delta$ while leaving all other prices unchanged.

The relaxation iteration terminates with either a flow augmentation (via step 4) or with a descent of the dual cost (via step 5). In order for the procedure to be well defined, however, we must show that whenever we return to step 2 from step 3 there is still left a labeled node with an unscanned label. Indeed when all node labels are scanned (i.e. the set S coincides with the labeled set) there is no balanced arc (m,k) such that $m \notin S$, $k \in S$ and $f_{mk} < c_{mk}$ or a balanced arc (k,m) such that $k \in S$, $m \notin S$ and $f_{km} > \ell_{km}$. It follows from the definition (19), (20) that

$$C(S,t) = \sum_{k \in S} d_k .$$

Under the circumstances above all nodes in S have nonnegative deficit and at least one node in S (the starting node i) has strictly positive deficit. Therefore $C(S,t) > 0$ and it follows that the procedure switches from step 3 to step 5 rather than to step 2.

If the starting t is integer, then $\delta$ will also be a positive integer and the dual cost is reduced by an integer amount each time step 5 is executed. Each time an augmentation takes place via step 4 the dual cost remains unchanged. If the starting f is integer all successive f will be integer so the amount of augmentation $\varepsilon$ in step 4 will be a positive integer. Therefore there can be only a finite number of augmentations between successive reductions of the dual cost. It follows that the algorithm will finitely terminate at an integer optimal pair (f,t) if the starting pair (f,t) is integer.

It can be seen that the relaxation iteration involves a comparable amount of computation per node scanned as the primal-dual method of the

previous section once it is realized that the quantity C(S,t) in step 3 can be computed recursively rather than recomputed each time the set S is enlarged in step 2. For this purpose it is most efficient to use the formula

$$C(S,t) = \sum_{i \in S} d_i - \sum_{\substack{(i,j):\text{balanced} \\ i \in S,\ j \notin S}} (x_{ij} - \ell_{ij}) - \sum_{\substack{(i,j):\text{balanced} \\ i \notin S,\ j \in S}} (c_{ij} - x_{ij}).$$

We note that a similar iteration can be constructed starting from a node with negative deficit. Here the set S is initially set to $N$ (instead of $\emptyset$) and each time a node k is scanned the node k is subtracted from S (instead of added to S). The straightforward details are left to the reader. Computational experience suggests that it is typically beneficial to initiate the descent iteration from nodes with both positive and negative deficit.

Line Search

Each time step 5 of the relaxation iteration is executed a descent is made along the elementary vector associated with the cutset corresponding to S. The stepsize $\delta$ of (29) corresponds to the first break point of the (piecewise linear) dual functional along the descent direction. It is possible to use instead an optimal stepsize that minimizes the dual functional along the descent direction as in nonlinear programming algorithms. It turns out that such a stepsize can be calculated quite efficiently by testing the sign of the directional derivative of the dual cost at successive breakpoints along the descent direction. Computational experimentation showed that this type of line search is beneficial, and was efficiently implemented in the RELAX code (see Section 6).

Single Node Iterations

The case where the relaxation iteration scans a single node (the starting node i), finds that

$$C(\{i\},t) > 0,$$

reduces its price (perhaps repeatedly via the line search mentioned earlier), and terminates is particularly important for the conceptual understanding of the algorithm. We believe that much of the success of the algorithm is owed to the relatively large number of single node iterations for many classes of problems.

Basically when only the price of a single node i is changed, the absolute value of the deficit of i is decreased at the expense of possibly increasing the absolute value of the deficit of its neighboring nodes. This is reminiscent of relaxation methods where a change of a single variable is effected with the purpose of satisfying a single constraint at the expense of violating others.

A dual viewpoint, reminiscent of coordinate descent methods, is that a single (the ith) coordinate direction is chosen and a line search is performed along this direction resulting in a cost reduction. Figure 2 shows the form of the dual cost function along the direction of the coordinate $p_i$ for a node with

$$d_i > 0.$$

The left slope of the function at $p_i$ is

$$C(\{i\},t)$$

while the right slope is

$$\overline{C}(\{i\},t) = \sum_{\substack{(i,m)\in A \\ (i,m):\text{active} \\ \text{or balanced}}} c_{im} + \sum_{\substack{(i,m)\in A \\ (i,m):\text{inactive}}} \ell_{im}$$

$$- \sum_{\substack{(m,i)\in A \\ (m,i):\text{active}}} c_{mi} - \sum_{\substack{(m,i)\in A \\ (m,i):\text{inactive} \\ \text{or balanced}}} \ell_{mi} .$$

We have

$$C(\{i\},t) \leq d_i \leq \overline{C}(\{i\},t) \tag{30}$$

so $d_i$ is a subgradient of the dual cost at p in the ith coordinate direction.

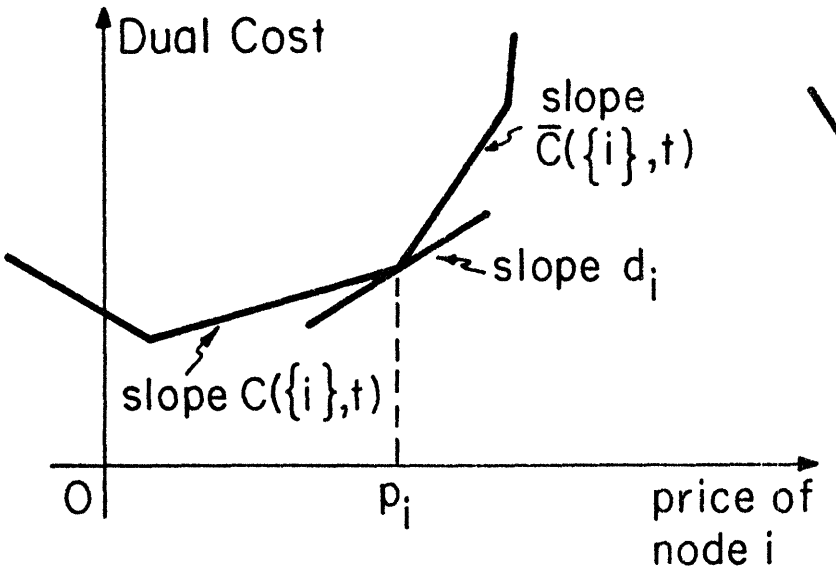
A single node iteration will be possible if and only if the left slope is positive or equivalently

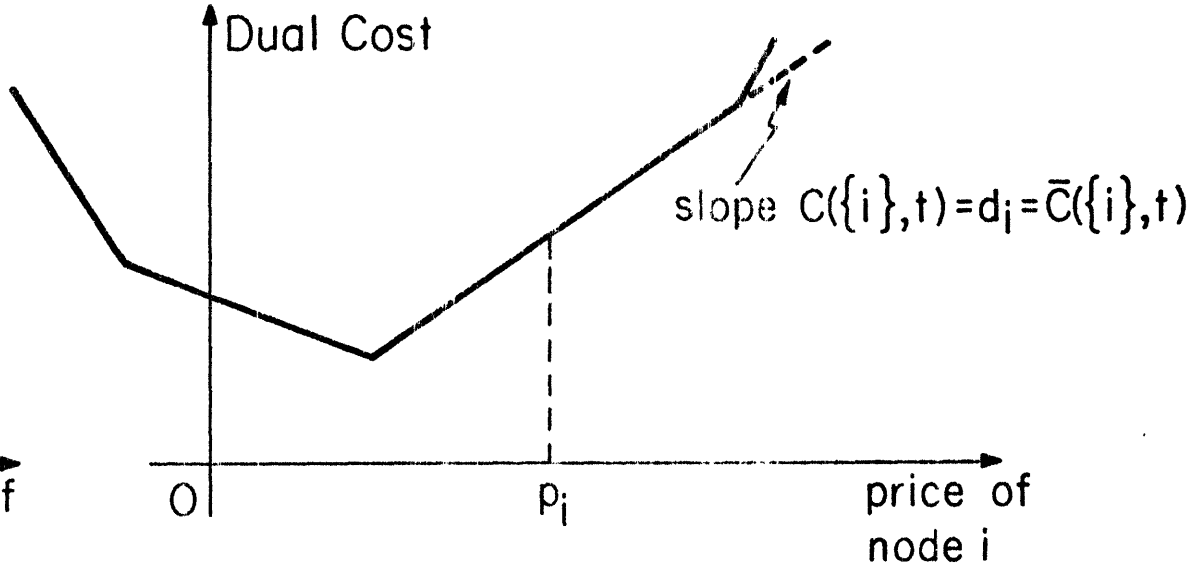
$$C(\{i\},t) > 0.$$

This will always be true if we are not at a corner and equality holds throughout in (30). However if the dual cost is nondifferentiable at $p_i$ along the ith coordinate, it may happen that (see Figure 2)

$$C(\{i\},t) \leq 0 < d_i \leq \overline{C}(\{i\},t)$$

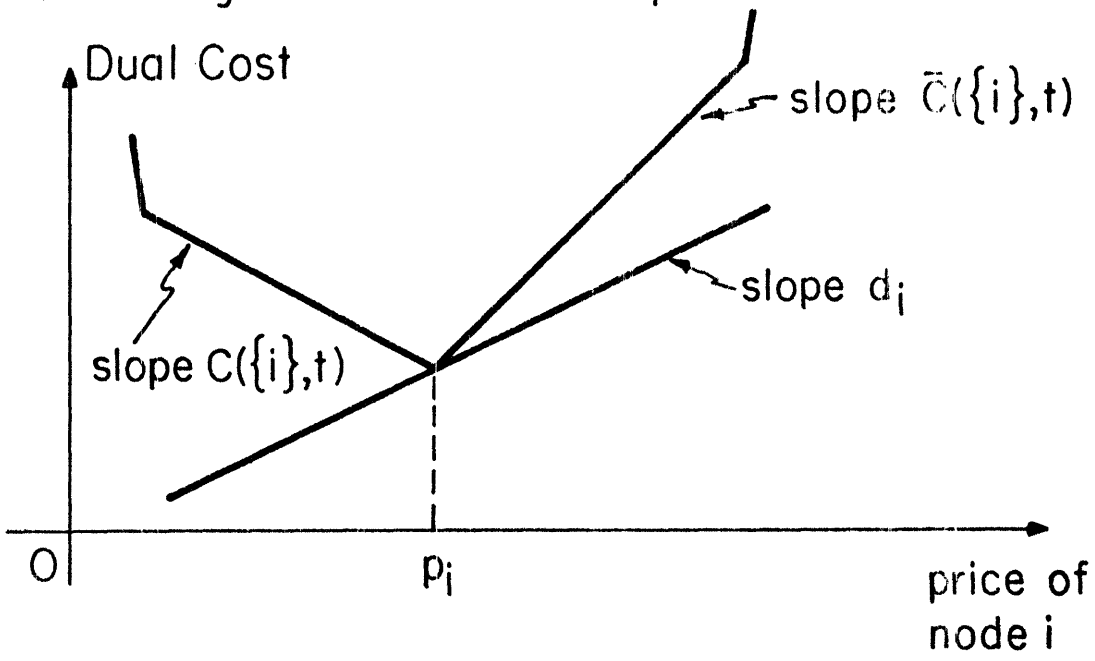
in which case we must resort to scanning more than one nodes. This point will be further clarified in the next section when we discuss the case of




Cases where a single node iteration is possible






Case where a single node iteration is not possible

Figure 2

strictly convex, differentiable arc costs. There the dual functional is differentiable and the analog of the relaxation method consists exclusively of single node iterations.

Degenerate Descent Iterations

If for a given t a strict subset S of $N$ can be found such that

$$C(S,t) = 0$$

then from Proposition 1 we see that the dual cost remains constant as we start moving along the elementary vector corresponding to t, i.e.

$$w(t + \gamma v) = w(t) , \qquad \forall\ \gamma \in [0,\delta)$$

where w, v, and $\delta$ are given by (22)-(24). We refer to such incremental changes in t as degenerate descent iterations. If the descent condition $C(S,t) > 0$ in the descent iteration [cf. (28)] is replaced by $C(S,t) \geq 0$ then we obtain an algorithm that produces at each iteration either a flow augmentation, or a strict dual cost reduction or a degenerate descent step. This algorithm can be guaranteed to terminate at an optimal solution under the following condition:

(C) For each degenerate descent iteration the starting node i has positive deficit $d_i$ and all nodes in the scanned set S have nonnegative deficit at the end of the iteration.

We refer the reader to [6] for a proof of this fact. It can be easily seen that condition (C) always holds when the set S consists of just the starting node i. For this reason if the descent iteration is modified so that a price adjustment at step 5 is made not only when $C(S,t) > 0$ but

also when $d_i > 0$, $S = \{i\}$ and $c(\{i\}, t) = 0$ the algorithm maintains its termination properties. The modification was implemented in the RELAX code (see Section 6) and can have an important beneficial effect for special classes of problems such as assignment and transportation problems. We don't quite understand the reasons for this but it apparently has to do with the orientation and shape of the isocost surfaces of the dual cost function relative to the elementary price vectors (see Fig. 3). For the assignment problem condition (C) is guaranteed to hold even if S contains more than one node. The assignment algorithm of [5] and the ASSIGN code (see Section 6) make extensive use of degenerate descent steps.

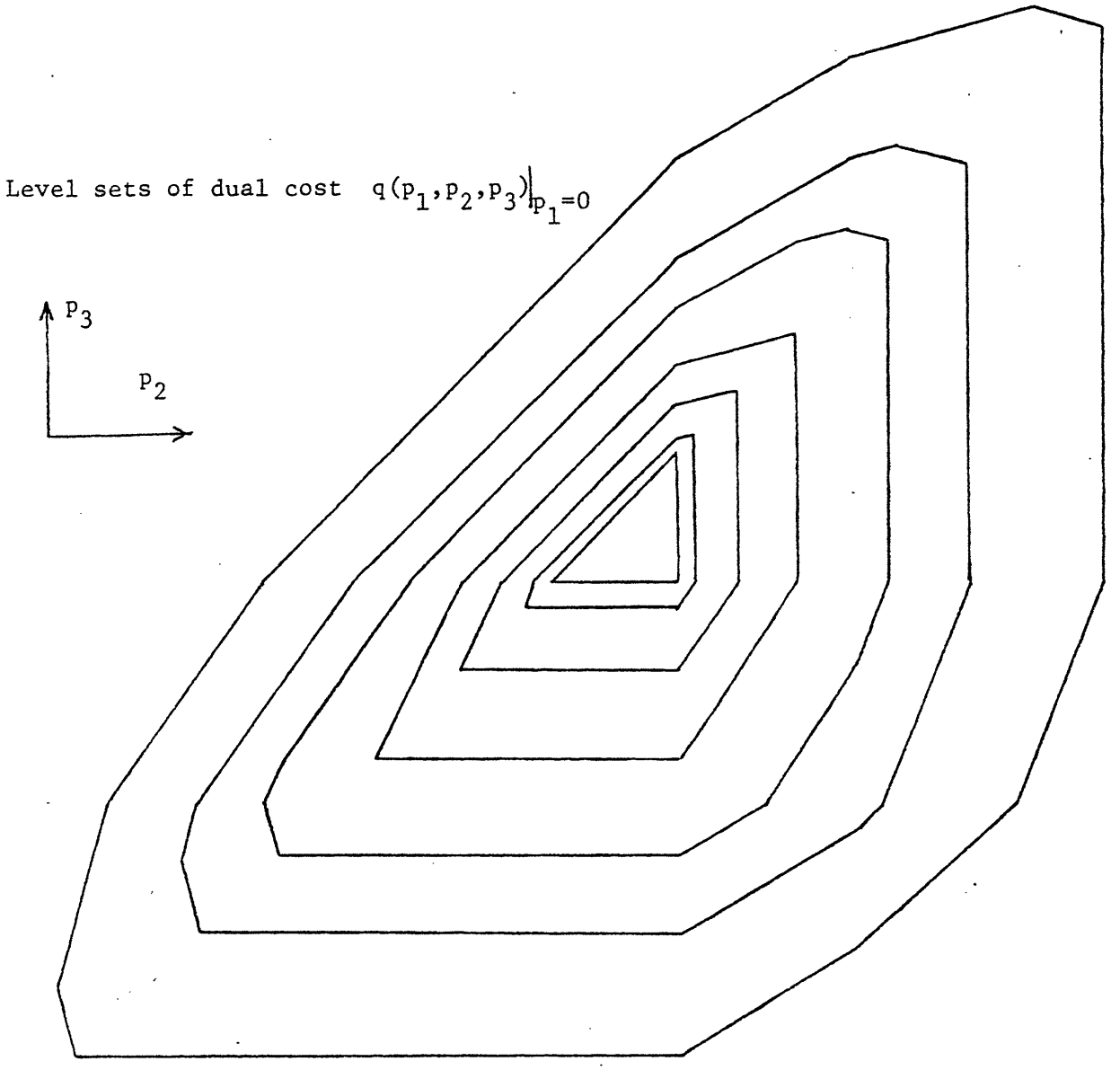



Figure 3: Level sets of a dual cost function for a three node problem. Notice the orientation of the level sets along the elementary vector (1,1).

## 4. Extension to Strictly Convex, Differentiable Arc Costs

Consider the following extension of problem (P) of Section 1:

$$\text{minimize} \quad \sum_{(i,j)\in A} g_{ij}(f_{ij}) \tag{CP}$$

$$\text{subject to } f\in C$$

where $g_{ij}: R \to (-\infty,+\infty]$ is given by

$$g_{ij}(f_{ij}) = \begin{cases} a_{ij}(f_{ij}) & \text{if } \ell_{ij} \le f_{ij} \le c_{ij} \\ +\infty & \text{otherwise} \end{cases} \tag{31}$$

and $a_{ij}: R \to R$ is a strictly convex function, $\ell_{ij}$, $c_{ij}$ are given scalars, and $C$ is the circulation subspace of the network given by (15).

By repeating the development of Section 1 (or see Rockafellar [2]) we obtain the dual problem

$$\text{minimize} \quad \sum_{(i,j)\in A} g^*_{ij}(t_{ij}) \tag{CD}$$

$$\text{subject to } t\in C^{\perp}$$

where $g^*_{ij}: R \to R$ is the conjugate convex function of $g_{ij}$ given by

$$g^*_{ij}(t_{ij}) = \sup_{f_{ij}} \{t_{ij}f_{ij} - g_{ij}(f_{ij})\}, \quad \forall\ (i,j)\in A \tag{32}$$

Figure 4 demonstrates an example of a pair $(g_{ij}, g^*_{ij})$. Because the (effective) domain of $g_{ij}$ is bounded, $g^*_{ij}$ is real valued. Also the strict

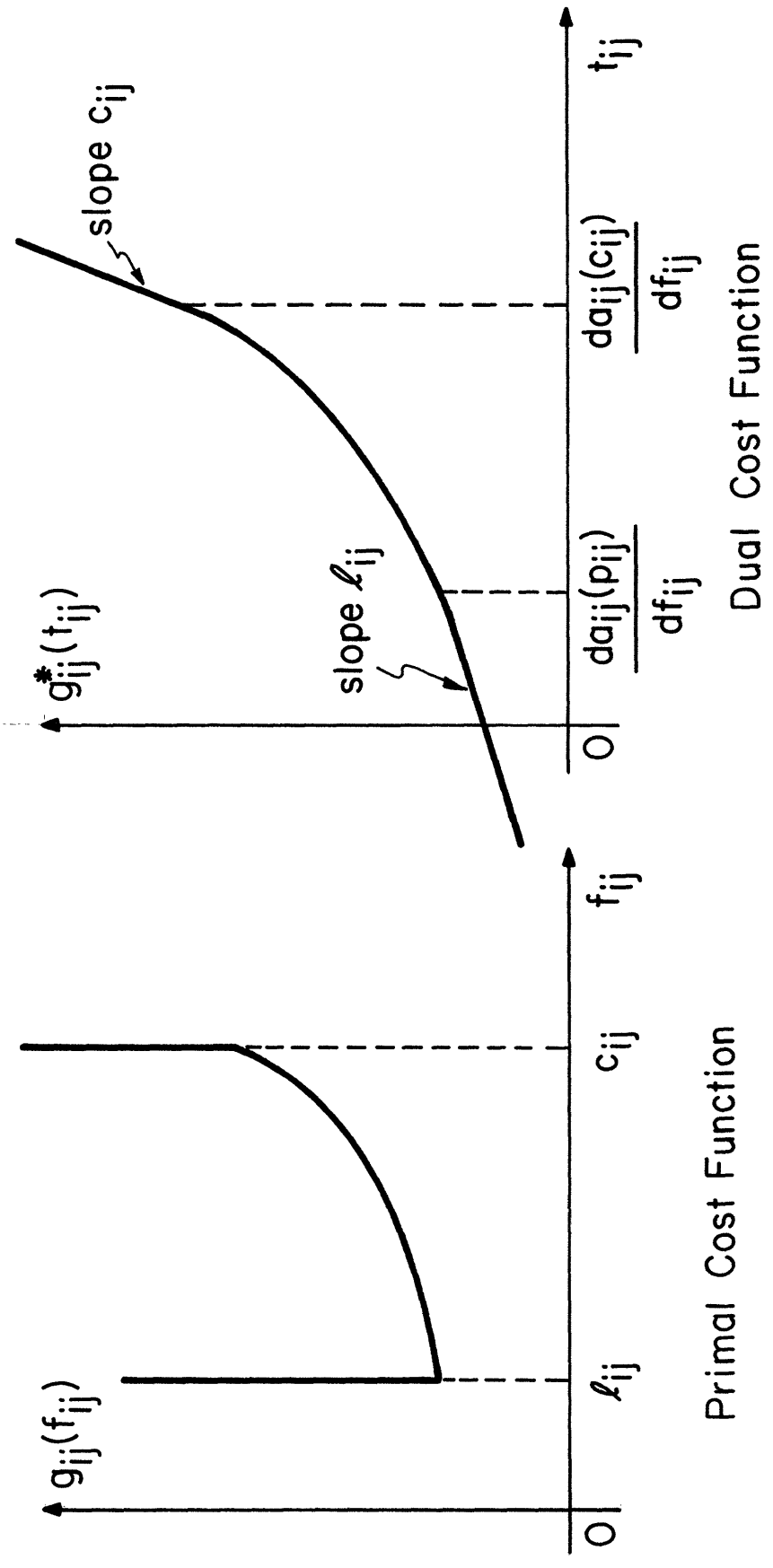

$g_{ij}(f_{ij})$
O
$\ell_{ij}$
$c_{ij}$
$f_{ij}$
Primal Cost Function
$g^*_{ij}(t_{ij})$
slope $\ell_{ij}$
slope $c_{ij}$
O
$\frac{da_{ij}(p_{ij})}{df_{ij}}$
$\frac{da_{ij}(c_{ij})}{df_{ij}}$
$t_{ij}$
Dual Cost Function

Figure 4

convexity of $g_{ij}$ implies that $g^*_{ij}$ is continuously differentiable ([1], p.253).

The necessary and sufficient conditions for optimality of a pair (f,t) can be verified to be (see also [2])

$$f \in C \quad , \quad t \in C^{\perp} \tag{33}$$

$$f_{ij} = c_{ij} \quad \text{if} \quad t_{ij} \geq \text{some subgradient of } a_{ij} \text{ at } c_{ij} \tag{34a}$$

$$f_{ij} = \ell_{ij} \quad \text{if} \quad t_{ij} \leq \text{some subgradient of } a_{ij} \text{ at } \ell_{ij} \tag{34b}$$

$$f_{ij} \text{ is such that } t_{ij} = \text{some subgradient of } a_{ij} \text{ at } f_{ij} \text{ otherwise.} \tag{34c}$$

The conditions (34) are the analogs of the complementary slackness conditions (10)-(12). It is easily shown that, for a given t, the vector f satisfying complementary slackness is uniquely specified via (34). This is an important difference with problems where the arc costs are not strictly convex (a fortiori linear). The condition $t \in C^{\perp}$ is equivalent to t being of the form

$$t_{ij} = p_i - p_j \quad , \quad \forall\ (i,j) \in A \tag{35}$$

where p is some price vector. The condition $f \in C$ is equivalent to all node deficits being zero [cf. (13)].

$$d_i = 0 \quad , \qquad \forall\ i \in N. \tag{36}$$

By representing the tension vector in terms of a price vector [cf. (35)] we can alternately write the dual problem (CD) as

$$\text{minimize} \quad \xi(p) \triangleq \sum_{(i,j)\in A} g^*_{ij}(p_i - p_j) \tag{37}$$

subject to no constraints on p

which is an unconstrained differentiable problem. The following proposition provides the form of the gradient of $\xi$.

Proposition 2: The first partial derivatives of $\xi$ are given by

$$\frac{\partial \xi(p)}{\partial p_i} = d_i , \quad i = 1,\ldots, |N| \tag{38}$$

where $d_i$ is the deficit of the ith node [cf. (9)] corresponding to the unique vector f satisfying the complementary slackness conditions (34).

Proof: The first derivative of $g^*_{ij}$ of (32) is (Lasdon [12], p. 426)

$$g^{*\prime}_{ij}(t_{ij}) = f_{ij} \tag{39}$$

where $f_{ij}$ is the unique maximizing scalar in (32). Furthermore in view of the form (31) of $g_{ij}$ it is easily seen that $f_{ij}$ satisfies the complementary slackness conditions (34). Using (9), (37) and (39) we obtain

$$\frac{\partial \xi(p)}{\partial p_i} = \sum_{(i,m)\in A} g^{*\prime}_{im}(t_{im}) - \sum_{(m,i)\in A} g^{*\prime}_{mi}(t_{mi})$$

$$= \sum_{(i,m)\in A} f_{im} - \sum_{(m,i)\in A} f_{mi} = d_i$$

Q.E.D.

Proposition 2 suggests an algorithm which, given a pair (f,t) satisfying complementary slackness, generates another such pair (f,t) with reduced value of dual cost.

Typical Relaxation Iteration:

Step 1: Choose a node i with nonzero deficit $d_i$. If no such node can be found terminate the algorithm. Else go to step 2.

Step 2: If $d_i > 0$ ($d_i < 0$) reduce (increase) $p_i$ to the level required so that the deficit of i corresponding to the flows given by (34) becomes zero. Terminate the iteration.

The iteration in effect changes p along the ith coordinate vector and minimizes the dual functional $\xi$ along that coordinate (deficit of i = zero means [cf. (38)] that the ith partial derivative of $\xi$ is zero or equivalently that $\xi$ is at a minimum along the ith coordinate). Of course the ith coordinate vector is also the elementary vector corresponding to the cutset associated with the single node i, so the algorithm falls within the framework of descent along elementary vectors.

It is also possible to view the iteration in the context of classical relaxation methods for solving systems of equations (cf. [13], p.219) since a single problem variable (the price $p_i$) is adjusted so as to satisfy a single constraint (the conservation of flow equation $d_i = 0$) at the expense of violating some other constraints (the conservation of flow equations at the neighboring nodes of i).

The expression

$$\frac{\partial \xi(p)}{\partial p_i} = d_i \tag{40}$$

for the ith partial derivative of $\xi$ should be compared with $-C(\{i\},t)$ [cf. (19),(20)] which, by Proposition 1, is the directional derivative of $\xi$ along the negative ith coordinate direction when arc costs are linear.

In the latter case we have in general

$$C(\{i\},t) \leq d_i$$

with equality if and only if there are no balanced arcs incident to i along which augmentation can be performed. It is therefore possible that

$$C(\{i\},t) < 0 < d_i$$

in which case reducing the price of a node with positive deficit will tend to increase the value of the dual cost (see Figure 2). For this reason it was necessary to consider price reductions involving more than one node in the case of linear arc costs. By contrast the equality (40) obviates this need for the problem of this section.

## 5. Basis for Computational Experimentation

Historically computational experimentation has been the primary method for comparative evaluation of network flow algorithms. During the sixties it was generally believed that primal-dual methods held an advantage over simplex methods. However during the seventies major improvements in implementation [14]-[20] using sophisticated data structures have propelled simplex algorithms to a position of prominence as far as general minimum cost flow problems are concerned. The situation is less clear for special classes of problems such as assignment where some computational comparisons [21], [22] suggest that primal-dual methods perform at least as well as simplex methods.

Analytical results aiding substantively the comparison of different methods are in scarce supply. An interesting observation was made by Zadeh [23] who showed that, for problems with nonnegative arc costs, primal-dual, dual simplex, primal simplex (with "big-M" starting method and most negative pivot rule), and the parametric method implement an essentially identical process--a sequence of augmentations along shortest paths between a supersource and a supersink node. The essential similarity between parametric and primal-dual methods actually extends to general linear programs with positive cost coefficients as shown to us recently by Gallager [24]. This is significant in view of recent average complexity results for the parametric method (Haimovich [25]). The "big-M" method is known to be more effective for network problems than the PhaseI-PhaseII method (Mulvey [26]). However there are pivot rules that are empirically more effective than the most negative rule and much research has been directed along this

direction ([26]-[28]). Zadeh [23] concludes that the "big M" method with empirically best pivot rule should be a better method than primal-dual for general minimum cost flow problems with nonnegative arc costs. This conclusion agrees with empirical observations of others (e.g. [16]) as well as our own (see Section 7).

There are several difficulties in comparing empirically computational methods the most important of which are:

1) Differences in computer hardware, software, timing routines, and test conditions make computation time reports difficult to compare.

2) One can only compare codes which are themselves particular implementations of methods. Therefore one cannot be sure whether an advantage of one code over another is due to inherent strength of the underlying method or to superior data structures and coding technique.

3) One can only compare codes on a restricted class of problems. Therefore certain performance features that manifest themselves on problems outside the class tested will be missed.

We were fortunate to obtain two state-of-the-art computer codes for general minimum cost network problems; KILTER (a primal-dual code due to Aashtiani and Magnanti [29]) and RNET (a primal simplex code due to Grigoriadis and Hsu [30]). We compared these codes with our implementation of the relaxation method of the previous section (called RELAX). We also implemented a specialized relaxation version for the assignment problem (called ASSIGN). In addition we implemented two versions of the primal-dual method (called PDUAL1 and PDUAL2) and a version of the Hungarian method for the assignment problem (called HUNGARIAN). A description of each of these

methods is given in the next section. We describe below our experimental approach:

Test Conditions

All methods were tested under identical conditions, same computer (VAX 11/750), same language (FORTRAN IV), same compiler (standard FORTRAN of the VMS system in the OPTIMIZE mode), same timing routine, and same system conditions (empty system at night). The CPU times reported were obtained using the system routines LIB$INIT_TIMER and LIB$SHOW_TIMER. These times do not include problem input and output but include algorithm initialization and testing for problem infeasibility. We also run a few problems with some of these codes on an IBM 370/168 with the Fortran compiler in the OPT=2 mode. The computation time on the IBM 370/168 is roughly 9 times faster than on the VAX 11/750 but this varies considerably from problem to problem (by as much as 20%). The VAX 11/750 is a relatively small machine on which problems of large size can produce an excessive number of page faults thereby severely distorting the computation time. The size of problems used in our experiments and the system configuration were such that page faults were never a serious difficulty.

The methods tested include parameters that must be set by the user. A single default set of parameters was chosen for each method and was kept unchanged throughout the experimentation. For RNET these parameters are in the range suggested by its authors with the parameter FRQ set at 7.0

Efficiency of Implementation

RNET is a mature primal simplex code developed and refined over a period of twelve years at Rutgers University. Computational results reported in [19] and reproduced in Table 7

suggest that RNET is faster on standard NETGEN benchmark problems [31] (see Table 1) than PNET [31] and GNET [17] both of which are sophisticated simplex codes that represented an advance in the state of the art at the time they were introduced. Kennington and Helgason have compared RNET with their own primal simplex code NETFLO on the first 35 NETGEN benchmark problems and conclude in their 1980 book ([20], p. 255) that "RNET... produced the shortest times that we have seen on these 35 test problems". Our own experimentation generally supports these findings and suggests that for general minimum cost flow problems RNET is at least as fast and probably faster than any other simplex code for which computation times on benchmark problems are available to us ([17], [20], [22], [29], [31]).

KILTER is an implementation of the primal-dual method that uses a sophisticated labeling scheme described in [29]. The version we tested is the fastest of nine versions tested in [29]. It is called KILTER9 in [29]. On the basis of the somewhat limited computational results of [29], KILTER outperforms by a wide margin earlier primal-dual codes such as SUPERK, SHARE and BOEING [31], and is comparable to the simplex code PNET [31]. KILTER is also generally faster than our own primal-dual codes PDUAL1 and PDUAL2 (see Section 7). However an extensive computational study by Mulvey and Dembo [33] shows that KILTER is outperformed on assignment problems under identical test conditions by LPNET (a primal simplex code due to Mulvey [18]). Our own experimentation also shows that KILTER is consistently outperformed by RNET and agrees with the generally held opinion that the most efficient primal-dual codes are slower than primal simplex codes on general minimum cost flow problems.

The preceding discussion was intended to show that the implementations of both RNET and KILTER seem very efficient. Therefore it appears valid to consider these codes as representative of the best that has been achieved through the enormous collective efforts of many people over many years to date with the primal simplex and primal-dual methods respectively.

The implementation of our own relaxation codes are of course the first of their type and it is probable that substantial improvements will be possible in the future. In particular no attempt was made to implement a labeling structure whereby labels are saved from one iteration to the next.

Test Problems and Random Generators

The types of problems on which we conducted our tests are:

1) Assignment problems
2) Transportation problems
3) Uncapacitated or lightly capacitated general minimum cost flow problems with nonnegative arc costs.
4) Heavily capacitated minimum cost problems with nonnegative arc costs.
5) Problems with both positive and negative arc costs.
6) Problems with piecewise linear arc cost functions.

Most of these problems were generated using the publicly available and widely used NETGEN program [31]. Standard benchmark problems from [31] as well as others were solved. Since asymmetric assignment and piecewise linear cost problems cannot be generated by NETGEN we wrote our own random problem generator for this purpose. This generator was also used to generate some problems of the NETGEN type to provide an additional comparative dimension.

## 6. Code Descriptions

The relaxation code RELAX and the primal-dual codes PDUAL1 and PDUAL2 we implemented solve the problem

$$\text{mimimize} \quad \sum_{(i,j)\in A} a_{ij} f_{ij}$$

subject to

$$\sum_{(m,i)\in A} f_{mi} - \sum_{(i,m)\in A} f_{im} = b_i, \quad \forall\, i \in N$$

$$0 \leq f_{ij} \leq c_{ij}, \quad \forall\, (i,j)\in A.$$

This form has become standard in network codes as it does not require storage and use of the array of lower bounds $\{\ell_{ij}\}$. Instead the smaller size array $\{b_i\}$ is stored and used. The problem (MCF) of Section 1 can be reduced to the form above by making the transformation of variables $f_{ij} \leftarrow f_{ij} - \ell_{ij}$. The method for representing the problem is the linked list structure suggested by Aashtiani and Magnanti [29] and used in their KILTER code (see also Magnanti [34]). Briefly, during solution of the problem we store for each arc its head and tail node, its capacity, its reduced cost $(a_{ij}-t_{ij})$, its flow $f_{ij}$, the next arc with the same start node, and the next arc with the same end node. An additional array of length equal to half the number of arcs is used for internal calculations. This array could be eliminated at the expense of a modest increase in computation time. The total storage for arc length arrays is 7.5 $|A|$. There is additional storage needed for node length arrays but this is relatively insignificant for all but extremely sparse problems. This compares unfavorably with primal simplex codes which can be implemented

with four arc length arrays.

The RELAX code implements with minor variations the relaxation algorithm of Section 3. Line search and degenerate descent steps are implemented as discussed in Section 3. The PDUAL1 and PDUAL2 codes implement the primal-dual algorithms of Section 2 and the appendix respectively. All three algorithms are implemented with very similar coding techniques and in fact share a considerable amount of code. In each of the three codes all labels are discarded after each iteration and this is an area where potential improvements in implementation shouldbe possible.

The codes assume no prior knowledge about the structure of the problem or the nature of the solution. Initial prices are set to zero and initial arc flows are set to their lower or upper bound depending on whether the arc cost is nonnegative or negative respectively. The RELAX code includes a preprocessing phase (included in the CPU time reported) whereby it tries to modify the arc capacities to as small a value as possible without changing the problem. Thus for transportation problems the capacity of each arc is set at the minimum of the supply and demand at the head and tail nodes of the arc. We found experimentally that this preprocessing can improve markedly the performance of RELAX particularly for transportation problems. We do not fully understand the nature of this phenomenon, but it is apparently related to the orientation and shape of the isocost surfaces of the dual cost. Generally speaking, tight arc capacity bounds increase the frequency of single node iterations. This behavior is in sharp contrast with that of primal simplex which benefits from loose arc capacity bounds (fewer extreme points to potentially search over), and appears to be one of the main reasons for the experimentally observed superiority of relaxation over primal simplex for heavily capacitated problems.

The specialized codes ASSIGN and HUNGARIAN solve the assignment problem

$$\text{minimize} \sum_{(i,j)\in A} a_{ij} f_{ij}$$

subject to

$$\sum_{j\in T} f_{ij} = 1 \quad , \quad \forall\, i\in S$$

$$\sum_{i\in S} f_{ij} \leq 1 \quad , \quad \forall\, j\in T$$

$$0 \leq f_{ij} \leq 1$$

where S and T are sets of sources and sinks respectively and $|S| \leq |T|$. ASSIGN is a relaxation algorithm, basically the one in Bertsekas [5]. It uses to the maximum degree possible degenerate descent steps and we found that this is an important factor contributing to computational efficiency. The reason is most likely related to the orientation of the isocost surfaces of the dual cost. HUNGARIAN is an implementation of Kuhn's Hungarian method [35]. It is based on the algorithm of the appendix and it is an improved version of the implementation given in [5]. This latter implementation was tested by McGinnis ([22], p. 238) and found to be superior to implementations based on the algorithm of Section 2. Both ASSIGN and HUNGARIAN require storage of three arc length arrays during problem solution.

It is possible to reduce the memory requirements of the codes by ordering the arc list of the network by head node, i.e., the outgoing arcs of the first node are listed first followed by the outgoing arcs of the second node etc. If this is done one arc length array becomes unnecessary thereby reducing the memory requirements of RELAX, PDUAL1, and PDUAL2 to 6.5 arc length arrays, and the memory requirements of ASSIGN and HUNGARIAN to two arc length arrays. The problem solution time remains essentially unaffected by this device, but the time needed to prepare (or alter) the

problem data will be increased. The same technique can also be used to reduce the memory requirements of the primal simplex method to three arc length arrays.

## 7. Computational Results

Our computational results are organized in eight tables. The first six tables involve comparisons under identical test conditions on a VAX 11/750. The last two tables involve our own experimentation on an IBM 370/168 and experimentation of others on different computers. Therefore the last two tables cannot be firmly relied upon for comparative purposes. They are however informative in that they corroborate the results of the experimentation on the VAX 11/750. Most of the problems shown were generated using the NETGEN program. The random number seed used is the same as the one in [31]. All additional information needed to replicate these problems is given in the corresponding tables.

### Table 1: (Standard NETGEN Benchmarks on VAX 11/750)

This table shows the results for the 40 problems described in detail in [31] and generated by the NETGEN program. The results show the substantial superiority of RELAX over the other codes for assignment and transportation problems. This finding was corroborated on a large number of additional assignment and transportation problems of broadly varying size. For lightly capacitated and uncapacitated problems RELAX and RNET are competitive and much superior to the other codes.

### Table 2: (Assignment Problems on VAX 11/750)

The first 15 problems of this table were generated by the NETGEN program. ASSIGN being a specialized assignment algorithm, outperforms RELAX by about a factor of 2. RELAX outperforms RNET by a factor that

is over 2.5. It is worth noting that the solution times of RNET for the standard benchmark problems 11-15 of [31] are the fastest we have seen--faster than those obtained by specialized assignment codes (see Table 8). It can therefore be concluded that relaxation methods provide a substantial improvement in the state of the art for solution of the assignment problem.

The last 5 problems were obtained using a special routine we wrote for generating asymmetric assignment problems. Note that the superiority of ASSIGN over HUNGARIAN increases markedly for asymmetric problems. This is probably due to the fact that HUNGARIAN relies to a large extent on the standard price and flow initialization procedure for efficiency ([7], p. 405). When the problem is asymmetric this initialization is much less effective. Additional experiments, not given here, showed also that the factor of superiority of ASSIGN over HUNGARIAN increases further when the arc cost range is larger (e.g. [1, 1000] rather than [1, 100]). Generally speaking the Hungarian method is sensitive to the arc cost range as shown elsewhere ([5], [22]). This type of sensitivity seems to be exhibited also by KILTER as evidenced by our limited experimentation. ASSIGN does not seem to be nearly as sensitive in this regard.

Table 3: (Transportation Problems on VAX 11/750)

These results are in general agreement with those of Table 1. RELAX gave solution times that are faster than over a factor of 2 over those of RNET. It should be noted here that the value of the default parameter FRQ used in all experiments with RNET (this parameter controls the pivoting strategy) is the one recommended for heavily capacitated transhipment

problems. Limited experimentation showed that when this parameter is set at the value recommended for transportation problems the margin of superiority of RELAX over RNET became smaller yet it was still substantial.

Table 4: (Heavily Capacitated Transhipment Problems on VAX 11/750)

Our experience with heavily capacitated transhipment problems with positive arc costs is similar to that for transportation problems. Generally speaking stringent capacity constraints favor RELAX over both RNET and KILTER. By contrast it appears that an extremely sparse network, as in problems 5 and 15 of this table (see also problems 16-19, 36-40 of Table 1) tends to favor RNET over RELAX.

Table 5: (Transhipment Problems with Both Positive and Negative Arc Costs on VAX 11/750)

When there are both positive and negative arc costs the performances of RNET and (particularly) KILTER deteriorate substantially. By contrast a comparison with Table 4 shows that RELAX is essentially unaffected by the presence of both positive and negative arc costs. The performance deterioration of RNET can be partially explained by the fact that negative arc costs result in many more arc flows at their upperbounds. (In the runs with RNET in this table all arc flows were initialized at zero). The sharp performance deterioration of KILTER can be partially explained by the fact that negative arc costs result in a much larger total flow augmentation required to obtain the solution.

Table 6: (Piecewise Linear Arc Cost Transhipment Problems)

These problems were obtained using our our random problem generator in order to test the effect of multiple arcs connecting the same pairs of nodes. The relative performance of RELAX and RNET is essentially the same as that shown in Tables 4 and 5. However KILTER is not designed for the case where there are multiple arcs connecting the same pair of nodes and handles this type of situation inefficiently. As a result its performance is much worse than expected.

Table 7: (Standard NETGEN Benchmarks on IBM 370/168)

This table represents an attempt to compare our own computational results with those of others. The times for RELAX and KILTER were obtained by ourselves on an IBM 370/168 at M.I.T. The times for RNET are given by Grigoriadis in [19] and were obtained on a different IBM 370/168. (We had no access to a version of RNET that would run on an IBM 370/168.) These times show RNET in more favorable light versus RELAX than those of Table 1. This can be partially explained by the fact that in the runs of [19] an optimized set of RNET parameters was used for each class of problems while we used the same set of parameters for all problems.

The times given for GNET (due to Bradley et al [17]), PNET and PNET-I (e.g. [31]) are collected from various sources and can be compared only indirectly with our times. McGinnis [22] gives a 10 to 7 speed advantage for the CDC 6600 over the CYBER 70/74, while Bradley et al [17] give a 6 to 5 speed advantage for the IBM 370/168 over the CDC6600. Even if these figures represent coarse approximations to reality, RELAX appears to be much faster than GNET, PNET, and PNET-I for the problems of Table 7.

Table 8: (NETGEN Assignment Benchmarks on IBM 370/168)

Again here it is not possible to make an accurate comparison of our times with the times for the specialized assignment codes AP-AB and PDAC, however on the basis of the speed comparisons given earlier between the IBM 370/168, the CDC6600 and the CYBER 70/74 our times appear to be far superior.

| Problem Type | Problem # | # of Nodes | # of Arcs | RELAX | RNET | KILTER | PDUAL1 | PDUAL2 |
|---|---|---|---|---|---|---|---|---|
| Transportation | 1 | 200 | 1300 | 2.29 | 3.11 | 8.81 | 16.05 | 17.78 |
| | 2 | 200 | 1500 | 2.52 | 3.68 | 9.04 | 15.98 | 17.24 |
| | 3 | 200 | 2000 | 2.45 | 4.27 | 9.22 | 20.35 | 25.42 |
| | 4 | 200 | 2200 | 3.21 | 4.95 | 10.45 | 19.39 | 23.30 |
| | 5 | 200 | 2900 | 3.21 | 7.12 | 16.48 | 22.88 | 30.70 |
| | 6 | 300 | 3150 | 5.13 | 9.16 | 25.08 | 43.99 | 55.18 |
| | 7 | 300 | 4500 | 7.35 | 12.61 | 35.55 | 55.01 | 81.85 |
| | 8 | 300 | 5155 | 5.04 | 14.73 | 46.30 | 53.77 | 69.30 |
| | 9 | 300 | 6075 | 7.87 | 18.57 | 43.12 | 62.32 | 110.88 |
| | 10 | 300 | 6300 | 6.14 | 16.10 | 47.80 | 57.97 | 94.46 |
| Total (Problems 1-10) | | | | 45.22 | 94.30 | 251.85 | 367.71 | 526.11 |
| Assignment | 11 | 400 | 1500 | 1.75 | 4.79 | 8.09 | 11.20 | 9.30 |
| | 12 | 400 | 2250 | 1.90 | 6.54 | 10.76 | 14.49 | 12.17 |
| | 13 | 400 | 3000 | 2.60 | 8.50 | 8.99 | 15.77 | 13.69 |
| | 14 | 400 | 3750 | 3.04 | 9.56 | 14.52 | 13.92 | 13.77 |
| | 15 | 400 | 4500 | 4.73 | 9.82 | 14.53 | 16.22 | 16.09 |
| Total (Problems 11-15) | | | | 14.02 | 39.21 | 56.89 | 71.60 | 65.02 |
| Uncapacitated & Lightly Capacitated Problems | 16 | 400 | 1306 | 4.36 | 2.72 | 13.57 | 16.71 | 9.53 |
| | 17 | 400 | 2443 | 3.53 | 3.38 | 16.89 | 23.02 | 13.13 |
| | 18 | 400 | 1306 | 3.95 | 2.59 | 13.05 | 16.50 | 8.93 |
| | 19 | 400 | 2443 | 3.66 | 3.55 | 17.21 | 21.97 | 11.93 |
| | 20 | 400 | 1416 | 5.06 | 2.97 | 11.88 | 22.68 | 12.85 |
| | 21 | 400 | 2836 | 5.17 | 4.38 | 19.06 | 33.65 | 19.93 |
| | 22 | 400 | 1416 | 5.09 | 2.84 | 12.14 | 19.42 | 10.46 |
| | 23 | 400 | 2836 | 5.95 | 4.50 | 19.65 | 30.32 | 15.94 |
| | 24 | 400 | 1382 | 2.27 | 2.66 | 13.07 | 14.68 | 6.25 |
| | 25 | 400 | 2676 | 3.24 | 5.76 | 26.17 | 25.06 | 11.03 |
| | 26 | 400 | 1382 | 2.14 | 2.39 | 11.31 | 10.78 | 3.94 |
| | 27 | 400 | 2676 | 2.85 | 3.47 | 18.88 | 15.39 | 4.77 |

TABLE 1 (continued on next page)

| Problem Type | Problem # | # of Nodes | # of Arcs | RELAX | RNET | KILTER | PDUAL1 | PDUAL2 |
|---|---|---|---|---|---|---|---|---|
| Uncapacitated and Lightly Capacitated Problems | 28 | 1000 | 2900 | 6.00 | 8.39 | 29.77 | 47.66 | 33.55 |
| | 29 | 1000 | 3400 | 6.97 | 11.87 | 32.36 | 50.36 | 37.06 |
| | 30 | 1000 | 4400 | 13.39 | 11.08 | 42.21 | 49.89 | 39.56 |
| | 31 | 1000 | 4800 | 11.57 | 10.33 | 39.11 | 48.94 | 40.49 |
| | 32 | 1500 | 4342 | 11.47 | 18.22 | 69.28 | 81.65 | 70.71 |
| | 33 | 1500 | 4385 | 17.71 | 17.12 | 63.59 | 91.91 | 78.12 |
| | 34 | 1500 | 5107 | 12.74 | 20.29 | 72.51 | 94.49 | 81.46 |
| | 35 | 1500 | 5730 | 11.38 | 18.15 | 67.49 | 104.42 | 93.00 |
| Total (Problems 16-35) | | | | 138.50 | 156.66 | 609.20 | 819.50 | 602.64 |
| Large Uncapacitated Problems | 36 | 8000 | 15000 | 397.57 | 270.77 | 1,074.76 | | |
| | 37 | 5000 | 23000 | 294.68 | 280.79 | 681.94 | | |
| | 38 | 3000 | 35000 | 170.48 | 269.85 | 607.89 | | |
| | 30 | 5000 | 15000 | 180.48 | 149.51 | 558.60 | | |
| | 40 | 3000 | 23000 | 81.75 | 171.02 | 369.40 | | |
| Total (Problems 36-40) | | | | 1,124.96 | 1,141.94 | 3,292.59 | | |

TABLE 1: Standard Benchmark Problems 1-40 of [31] obtained using NETGEN. All times are in secs on a VAX 11/750.

| Problem # | # of Sources | # of Sinks | # of Arcs | Cost Range | ASSIGN | HUNGARIAN | RELAX | RNET |
|---|---|---|---|---|---|---|---|---|
| 1 | 200 | 200 | 1,500 | 1-100 | 0.56 | 1.37 | 1.75 | 4.79 |
| 2 | " | " | 2,250 | " | 0.74 | 1.85 | 1.90 | 6.54 |
| 3 | " | " | 3,000 | " | 0.65 | 2.19 | 2.60 | 8.50 |
| 4 | " | " | 3,750 | " | 0.89 | 2.34 | 3.04 | 9.56 |
| 5 | " | " | 4,500 | " | 1.35 | 2.84 | 4.73 | 9.82 |
| 6 | 200 | 200 | 7,000 | 1-1,000 | 2.57 | 5.08 | 7.18 | 17.24 |
| 7 | 400 | 400 | " | " | 6.63 | 11.17 | 9.12 | 30.19 |
| 8 | 600 | 600 | " | " | 10.11 | 24.18 | 13.15 | 53.12 |
| 9 | 800 | 800 | " | " | 12.61 | 37.01 | 30.71 | 71.84 |
| 10 | 1,000 | 1,000 | " | " | 22.56 | 56.42 | 22.16 | 94.69 |
| 11 | 400 | 400 | 4,000 | 1-1,000 | 3.30 | 8.89 | 7.98 | 21.63 |
| 12 | " | " | 6,000 | " | 4.05 | 9.76 | 19.68 | 27.05 |
| 13 | " | " | 8,000 | " | 5.62 | 12.47 | 16.05 | 32.88 |
| 14 | " | " | 10,000 | " | 7.17 | 13.76 | 11.84 | 40.08 |
| 15 | " | " | 15,000 | " | 8.20 | 14.94 | 18.89 | 60.46 |
| Total (Problems 1-15) | | | | | 87.01 | 204.27 | 170.78 | 488.39 |
| 16 | 100 | 150 | 1,999 | 1-100 | 0.14 | 0.59 | | |
| 17 | 200 | 300 | 5,981 | " | 0.40 | 2.87 | | |
| 18 | 300 | 450 | 12,005 | " | 0.79 | 8.21 | | |
| 19 | 400 | 600 | 20,019 | " | 2.44 | 22.18 | | |
| 20 | 500 | 750 | 29,965 | " | 4.70 | 40.88 | | |
| Total (Problems 16-20) | | | | | 8.47 | 74.73 | | |

TABLE 2: Assignment Problems. Times in Secs on VAX 11/750.
Problems 1-5 are Identical with Problems 11-15 of Table 1.
Problems 6-15 Obtained Using NETGEN with 0% High Cost Arcs.

| Problem # | # of Sources | Sinks | # of Arcs | Cost Range | RELAX | RNET | KILTER |
|---|---|---|---|---|---|---|---|
| 1 | 200 | 200 | 7,000 | 1-100 | 9.26 | 25.90 | 54.20 |
| 2 | 400 | 400 | " | " | 23.35 | 52.82 | 99.36 |
| 3 | 600 | 600 | " | " | 35.28 | 78.00 | 155.27 |
| 4 | 800 | 800 | " | " | 48.33 | 119.56 | 248.66 |
| 5 | 1,000 | 1,000 | " | " | 71.54 | 122.68 | 292.74 |
| 6 | 200 | 200 | 6,000 | 1-100 | 7.93 | 23.16 | 48.04 |
| 7 | " | " | 8,000 | " | 13.18 | 27.96 | 63.78 |
| 8 | " | " | 10,000 | " | 13.74 | 31.86 | 86.72 |
| 9 | " | " | 12,000 | " | 20.24 | 35.65 | 113.27 |
| 10 | " | " | 15,000 | " | 20.48 | 35.37 | 150.68 |
| Total (Problems 1-10) | | | | | 263.33 | 552.96 | 1,312.72 |
| 11 | 100 | 300 | 7,000 | 1-100 | 12.14 | 20.55 | 68.22 |
| 12 | 200 | 600 | " | " | 17.66 | 48.32 | 116.32 |
| 13 | 300 | 900 | " | " | 25.81 | 66.10 | 206.66 |
| 14 | 350 | 1,050 | " | " | 40.45 | 86.42 | 232.50 |
| 15 | 400 | 1,200 | " | " | 46.91 | 106.67 | 252.93 |
| 16 | 100 | 300 | 6,000 | 1-100 | 7.80 | 17.96 | 54.39 |
| 17 | " | " | 8,000 | " | 12.94 | 21.51 | 77.39 |
| 18 | " | " | 10,000 | " | 14.27 | 25.25 | 115.22 |
| 19 | " | " | 12,000 | " | 14.39 | 26.08 | 121.71 |
| 20 | " | " | 15,000 | " | 23.64 | 38.27 | 156.51 |
| Total (Problems 11-20) | | | | | 216.01 | 457.13 | 1,401.85 |

TABLE 3: Transportation Problems. Times in Secs on VAX 11/750. All Problems Obtained Using NETGEN with Total Supply 200,000 and 0% High Cost Arcs.

| Problem # | # of Sources | # of Sinks | # of Arcs | Cost Range | Capacity Range | RELAX | RNET | KILTER |
|---|---|---|---|---|---|---|---|---|
| 1 | 200 | 200 | 7,000 | 1-100 | 100-500 | 15.22 | 43.92 | 124.17 |
| 2 | 400 | 400 | " | " | " | 31.99 | 71.75 | 165.14 |
| 3 | 600 | 600 | " | " | " | 51.06 | 95.96 | 179.80 |
| 4 | 800 | 800 | " | " | " | 67.40 | 167.00 | 251.94 |
| 5 | 1,000 | 1,000 | " | " | " | 94.15 | 101.31 | 280.68 |
| 6 | 200 | 200 | 6,000 | 1-100 | 100-1,000 | 14.05 | 38.60 | 91.21 |
| 7 | " | " | 8,000 | " | " | 19.27 | 46.45 | 108.97 |
| 8 | " | " | 10,000 | " | " | 20.04 | 50.22 | 111.83 |
| 9 | " | " | 12,000 | " | " | 22.61 | 66.22 | 122.33 |
| 10 | " | " | 15,000 | " | " | 28.85 | 72.67 | 210.91 |
| Total (Problems 1-10) | | | | | | 364.62 | 694.10 | 1,743.01 |
| 11 | 100 | 300 | 7,000 | 1-100 | 100-500 | 18.24 | 42.93 | 118.69 |
| 12 | 200 | 600 | " | " | " | 33.67 | 67.88 | 146.10 |
| 13 | 300 | 900 | " | " | " | 61.97 | 88.79 | 206.85 |
| 14 | 400 | 1,200 | " | " | " | 89.74 | 108.75 | 238.24 |
| 15 | 500 | 1,500 | " | " | " | 113.67 | 106.39 | 306.03 |
| 16 | 100 | 300 | 6,000 | 1-100 | 100-1,000 | 18.69 | 32.96 | 84.29 |
| 17 | " | " | 8,000 | " | " | 17.10 | 38.65 | 94.28 |
| 18 | " | " | 10,000 | " | " | 23.32 | 48.24 | 118.68 |
| 19 | " | " | 12,000 | " | " | 21.58 | 57.46 | 153.95 |
| 20 | " | " | 15,000 | " | " | 35.87 | 71.40 | 190.34 |
| Total (Problems 11-20) | | | | | | 433.85 | 663.45 | 1,657.45 |

TABLE 4: Capacitated Transhipment Problems. Times in Secs on VAX 11/750. All Problems Obtained Using NETGEN with Total Supply 200,000, 100% of Sources and Sinks Being Transhipment Nodes, 0% High Cost Arcs, and 100% of Arcs Capacitated. Each node is either a source or a sink.

| Problem # | # of Sources | # of Sinks | # of Arcs | Cost Range | Capacity Range | RELAX | RNET | KILTER |
|---|---|---|---|---|---|---|---|---|
| 1 | 200 | 200 | 7,000 | -50-50 | 100-1000 | 18.63 | 135.61 | 1,144.02 |
| 2 | 400 | 400 | " | " | " | 29.76 | 216.43 | 1,972.35 |
| 3 | 600 | 600 | " | " | " | 57.86 | 287.46 | 2,735.99 |
| 4 | 800 | 800 | " | " | " | 74.52 | 318.63 | 3,567.50 |
| 5 | 1,000 | 1,000 | " | " | " | 105.92 | 339.26 | 4,295.42 |
| 6 | 200 | 200 | 6,000 | -50-50 | 100-1,000 | 14.92 | 111.51 | 1,137.72 |
| 7 | " | " | 8,000 | " | " | 18.14 | 166.57 | 1,346.74 |
| 8 | " | " | 10,000 | " | " | 25.45 | 199.01 | 1,895.97 |
| 9 | " | " | 12,000 | " | " | 27.89 | 242.90 | 2,520.55 |
| 10 | " | " | 15,000 | " | " | 36.82 | 299.37 | 3,487.57 |
| Total (Problems 1-10) | | | | | | 409.91 | 2,316.75 | 24,103.80 |
| 11 | 100 | 300 | 7,000 | -50-50 | 100-1,000 | 17.94 | 124.55 | 1,130.51 |
| 12 | 200 | 600 | " | " | " | 35.13 | 219.54 | 1,932.00 |
| 13 | 300 | 900 | " | " | " | 54.50 | 284.27 | 2,710.03 |
| 14 | 400 | 1,200 | " | " | " | 80.53 | 330.92 | 3,275.79 |
| 15 | 500 | 1,500 | " | " | " | 109.63 | 330.59 | |
| 16 | 100 | 300 | 6,000 | -50-50 | 100-1,000 | 15.13 | 107.34 | |
| 17 | " | " | 8,000 | " | " | 17.76 | 148.69 | |
| 18 | " | " | 10,000 | " | " | 24.88 | 197.79 | |
| 19 | " | " | 12,000 | " | " | 28.13 | 240.88 | |
| 20 | " | " | 15,000 | " | " | 33.54 | 313.11 | |
| Total (Problems 11-20) | | | | | | 417.17 | 2,297.68 | |

TABLE 5: Capacitated Transhipment Problems with Both Negative and Positive Arc Costs. Same Problems as in Table 4 Except that the Cost Range is [-50,50].

| Problem # | # of Nodes | # of Arcs | # of Pieces per Arc | Cost Range | RELAX | RNET | KILTER |
|---|---|---|---|---|---|---|---|
| 1 | 50 | 600 | 1 | 1-100 | 0.66 | .80 | 2.20 |
| 2 | " | " | 4 | " | 2.35 | 3.19 | 15.84 |
| 3 | " | " | 10 | " | 5.11 | 10.01 | 12.78 |
| 4 | " | " | 15 | " | 8.88 | 15.61 | 160.89 |
| 5 | 50 | 600 | 1 | -50-50 | 0.84 | 2.64 | 13.07 |
| 6 | " | " | 4 | " | 2.41 | 11.48 | 175.54 |
| 7 | " | " | 10 | " | 5.95 | 32.58 | 958.68 |
| 8 | " | " | 15 | " | 10.40 | 54.21 | 2,464.50 |
| 9 | 100 | 600 | 1 | 1-100 | 1.72 | 1.74 | 5.49 |
| 10 | " | " | 4 | " | 4.64 | 7.23 | 56.14 |
| 11 | " | " | 10 | " | 9.98 | 14.63 | 182.76 |
| 12 | " | " | 15 | " | 15.59 | 24.04 | 543.56 |
| 13 | 100 | 600 | 1 | -50-50 | 1.28 | 3.59 | 20.82 |
| 14 | " | " | 4 | " | 3.86 | 15.57 | 310.68 |
| 15 | " | " | 10 | " | 7.62 | 39.30 | |
| 16 | " | " | 15 | " | 13.67 | 52.91 | |
| Total | | | | | 94.96 | 289.63 | |

TABLE 6: Transhipment problems with piecewise linear arc costs. Times in secs on VAX 11/750. Problems obtained using our own random generator. All arcs are capacitated in the range [100,1000]. Supply of each node is chosen from the range [-1000,1000].

| Problem Type | Probl. # | RELAX | RNET | KILTER | GNET | PNET | PNET-I |
|---|---|---|---|---|---|---|---|
| Transportation Problems | 1 | 0.28 | 0.39 | 0.91 | 1.06 | 1.30 | 0.92 |
| | 2 | 0.29 | 0.46 | 0.95 | 1.08 | 1.49 | 0.98 |
| | 3 | 0.30 | 0.53 | 0.96 | 1.45 | 1.94 | 1.20 |
| | 4 | 0.39 | 0.54 | 1.10 | 1.44 | 1.64 | 1.07 |
| | 5 | 0.39 | 0.57 | 1.82 | 1.76 | 1.88 | 1.61 |
| | 6 | 0.61 | 0.85 | 2.69 | 2.45 | 3.55 | 2.28 |
| | 7 | 0.90 | 0.97 | 3.88 | 3.39 | 4.06 | 2.79 |
| | 8 | 0.82 | 1.22 | 5.05 | 4.06 | 4.72 | 3.11 |
| | 9 | 0.98 | 1.50 | 4.84 | 4.12 | 4.80 | 3.29 |
| | 10 | 0.75 | 1.14 | 5.34 | 4.68 | 5.88 | 4.08 |
| Total (Prob. 1-10) | | 5.71 | 8.17 | 27.54 | 25.49 | 31.26 | 21.33 |
| Assignment Problems | 11 | 0.20 | 0.38 | 0.92 | 3.75 | 3.52 | |
| | 12 | 0.22 | 0.69 | 1.15 | 4.86 | 4.87 | |
| | 13 | 0.32 | 0.96 | 1.03 | 6.95 | 5.52 | |
| | 14 | 0.37 | 1.02 | 1.57 | 7.06 | 6.02 | |
| | 15 | 0.59 | 0.93 | 1.62 | 8.24 | 6.50 | |
| Total (Prob.11-15) | | 1.70 | 3.98 | 6.29 | 30.86 | 26.43 | |
| Uncapacitated & Lightly Capacitated Problems | 16 | 0.48 | 0.34 | 1.40 | | 2.02 | |
| | 17 | 0.39 | 0.37 | 1.89 | | 3.23 | |
| | 18 | 0.53 | 0.31 | 1.36 | | 2.38 | |
| | 19 | 0.47 | 0.41 | 1.87 | | 3.17 | |
| | 20 | 0.55 | 0.38 | 1.28 | | 2.36 | |
| | 21 | 0.58 | 0.50 | 2.20 | | 3.71 | |
| | 22 | 0.56 | 0.37 | 1.31 | | 1.97 | |

TABLE 7: (Continued on next page)

| Problem Type | Probl. # | RELAX | RNET | KILTER | GNET | PNET |
|---|---|---|---|---|---|---|
| Uncapacitated & Lightly Capacitated Problems | 23 | 0.68 | 0.47 | 2.17 | | 3.20 |
| | 24 | 0.23 | 0.40 | 1.49 | | 2.68 |
| | 25 | 0.35 | 0.64 | 2.79 | | 3.26 |
| | 26 | 0.23 | 0.32 | 1.16 | | 2.33 |
| | 27 | 0.32 | 0.40 | 2.01 | 0.50 | 3.30 |
| | 28 | 0.72 | 0.93 | 3.38 | | 6.35 |
| | 29 | 0.81 | 1.16 | 3.62 | | 7.39 |
| | 30 | 1.71 | 1.16 | 4.68 | | 9.08 |
| | 31 | 1.42 | 1.23 | 4.28 | | 9.59 |
| | 32 | 1.34 | 1.25 | 7.29 | | 15.70 |
| | 33 | 2.19 | 1.63 | 7.10 | | 20.20 |
| | 34 | 1.50 | 1.60 | 7.74 | | 17.10 |
| | 35 | 1.29 | 2.36 | 6.98 | | 19.39 |
| Total (Prob.16-35) | | 16.35 | 16.23 | 66.00 | | 138.41 |

TABLE 7: Benchmark Problems 1-35 obtained by NETGEN in [31].

All times in secs as follows:

RELAX: Our time on IBM 370/168, FORTRAN, OPT=2

RNET : IBM 370/168, FORTRAN, OPT=2, times from [19] (default parameters)

KILTER: Our time on IBM 370/168, FORTRAN, OPT=2

GNET : Problems 1-10, CDC 6600, FTN, OPT=2, times from [32]

Problems 11-15, CYBER 70/74, FTN, times from [22]

Problem 27, IBM 370/168, time from [17]

PNET : CCC 6600, RUN, times from [31]

PNET-1: CDC 6600, FTN, OPT=2, times from [32].

| Probl. # | ASSIGN | HUNGARIAN | AP-AB | PDAC |
|---|---|---|---|---|
| 1 | 0.05 | 0.13 | 0.97 | 1.37 |
| 2 | 0.07 | 0.18 | 1.14 | 1.42 |
| 3 | 0.06 | 0.22 | 1.48 | 2.60 |
| 4 | 0.08 | 0.24 | 1.61 | 2.79 |
| 5 | 0.13 | 0.29 | 1.68 | 3.98 |
| Total | 0.29 | 1.06 | 6.88 | 12.16 |

TABLE 8: Benchmark assignment problems genrated by NETGEN. Same as problems 11-15 of Tables 1 and 7. All times in secs as follows:

ASSIGN: Our times on IBM 370/168, FORTRAN, OPT=2.

HUNGARIAN: Our times on IBM 370/168, FORTRAN, OPT=2.

AP-AB: Specialized assignment code of [36]. Times on CDC 6600, RUN from [36].

PDAC: Specialized assignment code of [22]. Times on CYBER 70/74, FTN from [22].

## 8. Conclusions

Relaxation methods adapt nonlinear programming ideas to solve linear network flow problems. They are apparently superior to the classical methods in terms of speed of solution for important classes of problems including assignment, transportation and heavily capacitated transhipment problems. Their main disadvantage relative to primal simplex is that they require more computer memory. However technological trends are such that this disadvantage should become less significant in the future.

Much remains to be done to effect improvements in implementation of relaxation methods, particularly in the area of preserving labeling information from one iteration to the next. Furthermore relaxation ideas should be applicable to problems beyond the class considered in this paper. Our computational results did not provide any clear indication as to whether any of the three methods tested (relaxation, primal simplex, and primal-dual) has a superior average computational complexity over the others. Experimentation with larger problems may provide some evidence in this regard.

## References


[1] Rockafellar, R. T., Convex Analysis, Princeton Univ. Press, 1970.

[2] Rockafellar, R. T., "Monotropic Programming: Descent Algorithms and Duality", in Nonlinear Programming 4, by O. L. Mangasarian, R. Meyer, and S. Robinson (eds.), Academic Press, 1981, pp. 327-366.

[3] Rockafellar, R. T., "The Elementary Vectors of a Subspace of $R^N$", in Combinatorial Mathematics and Its Applications, by R. C. Bose and T. A. Dowling (eds.), The Univ. of N. Carolina Press, Chapel Hill, N.C., 1969, pp. 104-127.

[4] Ford, L. R., Jr., and Fulkerson, D. R., Flows in Networks, Princeton Univ. Press, Princeton, N. J., 1962.

[5] Bertsekas, D. P., "A New Algorithm for the Assignment Problem", Math. Programming, Vol. 21, 1981, pp. 152-171.

[6] Bertsekas, D. P., "A Unified Framework for Minimum Cost Network Flow Problems", LIDS Report LIDS-P-1245-A, M.I.T., October 1982.

[7] Dantzig, G. B., Linear Programming and Extensions, Princeton Univ. Press, Princeton, N. J., 1963.

[8] Lawler, E. L., Combinatorial Optimization: Networks and Matriods, Holt, Rinehart & Winston, New York, 1976.

[9] Papadimitriou, C. H., and Steiglitz, K., Combinatorial Optimization: Algorithms and Complexity, Prentice Hall, Englewood Cliffs, N. J., 1982.

[10] Stern, T. E., "A Class of Decentralized Routing Algorithms Using Relaxation", IEEE Trans. on Communications, Vol. COM-25, 1977, pp. 1092-1102.

[11] Hassin, R., "The Minimum Cost Flow Problem: A Unifying Approach to Dual Algorithms and a New Tree-Search Algorithm", Math. Programming, Vol. 25, 1983, pp. 228-239.

[12] Lasdon, L. S., Optimization Theory for Large Systems, McMillan, N.Y., 1970.

[13] Ortega, J. M., and Rheinboldt, W. C., Iterative Solution of Nonlinear Equations in Several Variables, Academic Press, N.Y., 1970.

[14] Srinivasan, V. and Thompson, G. L., "Benefit-Cost Analysis of Coding Techniques for the Primal Transportation Algorithm", JACM, Vol. 20, pp. 194-213.

[15] Glover, F., Karney, D., Klingman, D., and Napier, A., "A Computation Study on Start Procedures, Basis Change Criteria and Solution Algorithms for Transportation Problems", Management Science, Vol. 20, 1974, pp. 793-819.

[16] Glover, F., Karney, D., and Klingman, D., "Implementation and Computational Comparisons of Primal, Dual and Primal-Dual Computer Codes for Minimum Cost Network Flow Problems", Networks, Vol. 4, 1974, pp. 191-212.

[17] Bradley, G. H., Brown, G. G., and Graves, G. W., "Design and Implementation of Large Scale Transhipment Algorithms", Management Science, Vol. 24, 1977, pp. 1-34.

[18] Johnson E., "Networks and Basic Solutions", Operations Research, Vol. 14, 1966, pp. 619-623.

[19] Grigoriadis, M. D., "Algorithms for the Minimum Cost Single and Multicommodity Network Flow Problems", Notes for Summer School in Combinatorial Optimization SOGESTA, Urbino, Italy, July 1978.

[20] Kennington, J., and Helgason, R., Algorithms for Network Programming, Wiley, N.Y., 1980.

[21] Hatch, R. S., "Bench Marks Comparing Transportation Codes Based on Primal Simplex and Primal-Dual Algorithms", Operations Research, Vol. 23, 1975, pp. 1167-1172.

[22] McGinnis, L. F., "Implementation and Testing of a Primal-Dual Algorithm for the Assignment Problem", Operations Research, Vol. 31, pp. 277-291, 1983.

[23] Zadeh, N., "Near-Equivalence of Network Flow Algorithms", Tech. Rep. No. 26, Department of Operations Research, Stanford University, Dec. 1979.

[24] Gallager, R. G., "Parametric Optimization and the Primal-Dual Method", Lab. for Information and Decision Systems Report, Mass. Institute of Technology, June 1983.

[25] Haimovich, M., "The Simplex Algorithm is Very Good! -- On the Expected Number of Pivot Steps and Related Properties of Random Linear Programs", Columbia Univ. Report, April 1983.

[26] Mulvey, J. M., "Pivot Strategies for Primal-Simplex Network Codes", JACM, Vol. 25, 1978, pp. 266-270.

[27] Mulvey, J. M., "Testing of a Large-Scale Network Optimization Program", Math. Programming, Vol. 15, 1978, pp. 291-314.

[28] Gavish, B., Schweitzer, P., and Shlifer, E., "The Zero Pivot Phenomenon in Transportation Problems and Its Computational Implications", Math. Programming, Vol. 12, 1977, pp. 226-240.

[29] Aashtiani, H. A., and Magnanti, T. L., "Implementing Primal-Dual Network Flow Algorithms", Operations Research Center Report 055-76, Mass. Institute of Technology, June 1976.

[30] Grigoriadis, M. D., and Hsu, T., "The Rutgers Minimum Cost Network Flow Subroutines", (RNET documentation), Dept. of Computer Science, Rutgers University, Nov. 1980.

[31] 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, pp. 814-822., 1974.

[32] Barr, R. S., and Turner, J. S., "Microdata File Merging Through Large-Scale Network Technology", Math. Programming Study 15, 1981, pp. 1-22.

[33] Dembo, R. S., and Mulvey, J. M., "On the Analysis and Comparison of Mathematical Programming Algorithms and Software", Harvard Business School Report 76-19, 1976.

[34] Magnanti, T., "Optimization for Sparse Systems", in Sparse Matrix Computations (J. R. Bunch and D. J. Rose, eds.), Academic Press, N.Y., 1976, pp. 147-176.

[35] Kuhn, H. W., "The Hungarian Method for the Assignment Problem", Naval Research Logistics Quarterly, Vol. 2, 1955, pp. 83-97.

[36] Barr, R. S., Glover, F., and Klingman, D., "The Alternating Basis Algorithm for Assignment Problems", Math. Progr., Vol. 13, p. 1, 1977.

Appendix: An Alternate Implementation of the Primal-Dual Algorithm

The following algorithm implements the primal-dual method with at most $O(|N|^2)$ computation between augmentations. The algorithm is the basis for the PDUAL2 and HUNGARIAN codes (see Section 6). For a price vector $p = \{p_i | i \in N\}$ and a flow vector $f = \{f_{ij} | (i,j) \in A\}$ we say that the pair (f,p) satisfies complementary slackness if the tension vector t corresponding to p together with f satisfy (10)-(12).

Step 0: Select a pair (f,p) satisfying complementary slackness.

Step 1: Discard all existing labels. Give the label "0" to all nodes i with $d_i > 0$. Let $\pi_i = 0$ for all i with $d_i > 0$ and $\pi_i = \infty$ for all i with $d_i \leq 0$. If $d_i = 0$ for all i terminate, else go to step 2.

Step 2: Choose a labeled but unscanned node i with $\pi_i = 0$ and go to step 3. If no such node can be found go to step 5.

Step 3: Scan the label of the node i as follows. For each $(i,m) \in A$ for which $0 \leq p_i - p_m - a_{im} < \pi_m$ and $\ell_{im} < f_{im}$ give node m the label "i" (replacing any existing label) and set $\pi_m = p_i - p_m - a_{im}$. For each $(m,i) \in S$ for which $0 \leq p_i - p_m + a_{mi} < \pi_m$ and $f_{mi} < c_{mi}$ give node m the label "i" (replacing any existing label) and set $\pi_m = p_i - p_m + a_{mi}$. If for any of the nodes m just labeled from i we have $\pi_m = 0$ and $d_m < 0$ go to step 4. Else go to step 2.

Step 4 (Flow Augmentation): An augmenting path has been found which starts at the node m with $d_m < 0$ identified in step 3 and ends at a node i with $d_i > 0$. The path can be constructed by tracing labels starting from m. Let $\varepsilon > 0$ be the capacity of the path. Increase by $\varepsilon$ the flow of all

arcs on the path that are oriented in the direction from m to i, reduce by $\varepsilon$ the flow of all other arcs on the path and go to step 1.

Step 5 (Price Adjustment): Let

$$\delta = \min \{\pi_i \mid i \in N, \pi_i > 0\}$$

Let

$$p_i \leftarrow p_i - \delta \qquad \forall\ i \text{ with } \pi_i = 0$$

$$\pi_i \leftarrow \pi_i - \delta \qquad \forall\ i \text{ with } \pi_i > 0$$

and go to step 2.

The implementation above is equivalent to the one of Section 2 but calculates the increment of price adjustment $\delta$ with $O(|N|)$ computation as opposed to $O(|A|)$ computation [cf. (27)]. Since step 5 will be executed at most $|N|-1$ times between augmentations and all other computation between augmentations is bounded by $O(|N|^2)$, the total amount of computation between augmentations in the implementation above is bounded by $O(|N|^2)$.