Mathematical Programming 32 (1985) 125-145
North-Holland

A UNIFIED FRAMEWORK FOR PRIMAL-DUAL METHODS IN MINIMUM COST NETWORK FLOW PROBLEMS

Dimitri P. BERTSEKAS

Department of Electrical Engineering and Computer Science, Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, MA 02139, USA

Received 13 October 1982
Revised manuscript received 21 February 1984

We introduce a broad class of algorithms for finding a minimum cost flow in a capacitated network. The algorithms are of the primal-dual type. They maintain primal feasibility with respect to capacity constraints, while trying to satisfy the conservation of flow equation at each node by means of a wide variety of procedures based on flow augmentation, price adjustment, and ascent of a dual functional. The manner in which these procedures are combined is flexible thereby allowing the construction of algorithms that can be tailored to the problem at hand for maximum effectiveness. Particular attention is given to methods that incorporate features from classical relaxation procedures. Experimental codes based on these methods outperform by a substantial margin the fastest available primal-dual and primal simplex codes on standard benchmark problems.

Key words: Primal-Dual, Out-of-Kilter, Relaxation, Network Flow.

1. Introduction

Consider a directed graph with set of nodes $\mathcal{N}$ and set of arcs $\mathcal{A}$. 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 problem

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

subject to

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

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

where l_{ij} and c_{ij} are given integers. We assume throughout that there exists at least one feasible solution of (MCF).

For simplicity in what follows we assume that there is at most one arc associated with each ordered pair of nodes (i,j). However this is not an essential restriction and the algorithms and results of the paper can be trivially modified to account for the possibility of multiple arcs joining a pair of nodes.

This work was supported by the National Science Foundation under Contract NSF/ECS 8217668.

There is a large variety of methods for solving (MCF) the most popular of which are primal simplex methods, primal-dual methods, and the out-of-kilter method (see [5, 6, 11, 13]). Our purpose in this paper is to propose a broad and flexible class of algorithms embedded in a primal-dual framework. It does not appear possible to relate these methods on a one-to-one basis with any of the existing methods, although particular implementations for specific problems yield known algorithms—for example the Hungarian method for the assignment problem [10], and versions of other known primal-dual or out-of-kilter algorithms. However, even for assignment and transportation problems, other implementations yield algorithms that differ substantially from classical primal-dual methods in both their form and their performance. For example a particular implementation of the class of algorithms of this paper yields the assignment algorithm recently proposed by the author and shown via computational experiment to be greatly superior to the Hungarian method [3]. In fact this algorithm served as the starting point for the developments of this paper. The single node procedures described in Section 3 are reminiscent of *relaxation methods* whereby the problem variables are adjusted with the aim of satisfying a single unsatisfied constraint (the conservation of flow equation at a single node in our case) at the expense of violating some others. Indeed if these procedures are applied to a problem with convex piecewise linear costs with many break points, the resulting algorithm asymptotically approaches a relaxation method due to Stern [17] as the number of break points tends to infinity.

In the next section we introduce a class of algorithms as consisting of a sequence, in any order, of three basic and flexible steps called *flow augmentation*, *price adjustment*, and *dual ascent*. We show that any algorithm in the class terminates at an optimal solution of (MCF) in a finite number of operations. Various alternatives for implementating these steps are described in Section 3. Some specific algorithms and relations with classical methods are discussed in Section 4. Computational results with the new algorithms involving the single node procedures are described in detail in a separate report [4] and are also summarized in Section 4. These results indicate impressive advantages in terms of computation time over classical primal-dual methods, and substantial advantages over the fastest primal simplex codes available at present for most types of problems of interest.

2. A class of primal–dual algorithms

Let us associate a Lagrange multiplier p_i with the ith conservation of flow constraint (1). By denoting by f and p the vectors with elements f_{ij}, $(i,j) \in \mathcal{A}$, and p_i, $i \in \mathcal{N}$ respectively, we can write the corresponding Lagrangian function

$$L(f,p) = \sum_{(i,j)\in\mathcal{A}} (a_{ij} + p_j - p_i) f_{ij}. \tag{3}$$

Consider also the dual problem

maximize

subject to no constraints on p, (4)

where the dual functional q is given by

$$q(p) = \min_{l_{ij} \le f_{ij} \le c_{ij}} L(f, p) = \sum_{(i,j) \in \mathcal{A}} q_{ij}(p_i - p_j), \tag{5a}$$

$$q_{ij}(p_i - p_j) = \min_{l_{ij} \le f_{ij} \le c_{ij}} \{(a_{ij} + p_j - p_i) f_{ij}\}. \tag{5b}$$

The function q_{ij} is shown in Fig. 1. This formulation of the dual problem is consistent with classical duality frameworks [14–16] but can also be obtained via standard linear programming duality [11, p. 144]. The scalar p_i will be referred to as the *price of node i.*

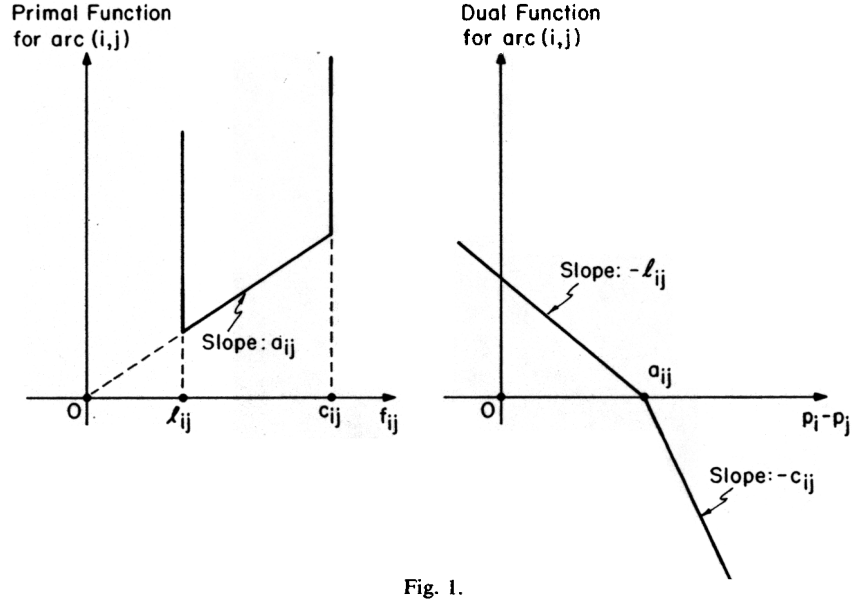


Fig. 1.

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

Inactive if $p_i < a_{ij} + p_j$. (6)

Balanced if $p_i = a_{ij} + p_j$. (7)

Active if $p_i > a_{ij} + p_j$. (8)

For any flow vector f the scalar

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

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 saddle point conditions for optimality in connection with (MCF) and its dual given by (3)-(5) state that (f, p) is a primal and dual optimal solution pair if and only if

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

$$l_{ij} \leq f_{ij} \leq c_{ij} \quad \text{for all balanced arcs } (i,j), \tag{11}$$

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

$$d_i = 0 \quad \text{for all nodes } i.$$

Relations (10)-(12) are the well known *complementary slackness conditions.*

In the general algorithm that we consider *we have at the beginning of each iteration, a pair* (f, p) *satisfying complementary slackness* ((10)-(12)). If there are no node deficits (i.e., $d_i = 0$ for all $i \in \mathcal{N}$) the algorithm stops—we are at an optimum. Otherwise there must exist at least one node i with positive deficit ($d_i > 0$) and at least one node i' with negative deficit ($d_{i'} < 0$) since, in view of (9), we always have $\sum_{i \in \mathcal{N}} d_i = 0$. *The general purpose of the iteration is to work towards eliminating those deficits either directly or indirectly through improvement of the value of the dual functional.* The general procedures by means of which this can be done will now be formalized.

Definition 1. Given a pair (f, p) satisfying complementary slackness, an *adjustment step* determines a new pair $(\bar{f}, \bar{p})$ satisfying complementary slackness and such that the new deficit $\bar{d}_i$ and price $\bar{p}_i$ of each node i satisfy

$$d_i \leq \bar{d}_i, \quad \bar{p}_i \geq p_i \quad \text{if } d_i < 0, \tag{14}$$

$$0 \leq \bar{d}_i, \quad \bar{p}_i \leq p_i \quad \text{if } d_i \geq 0. \tag{15}$$

We say that the adjustment step is a *flow adjustment step* (FAS for short) if

$$d_i < \bar{d}_i \text{ for at least one node } i \text{ with } d_i < 0, \tag{16}$$

and a *price adjustment step* (PAS for short) if

$$\bar{p}_i < p_i \text{ for at least one node } i \text{ with } d_i \geq 0. \tag{17}$$

An important fact about an adjustment step is that *it will not increase the total absolute deficit*, i.e.

$$\sum_{i \in \mathcal{N}} |\bar{d}_i| \leq \sum_{i \in \mathcal{N}} |d_i|. \tag{18}$$

To see this note that from (14) and (15) and the fact $\sum_{i \in \mathcal{N}} d_i = 0$ we have

$$\tfrac{1}{2} \sum_{i \in \mathcal{N}} |d_i| = \sum_{\substack{i \in \mathcal{N} \\ d_i < 0}} d_i \leq \sum_{\substack{i \in \mathcal{N} \\ \bar{d}_i < 0}} d_i \leq \sum_{\substack{i \in \mathcal{N} \\ \bar{d}_i < 0}} \bar{d}_i = -\tfrac{1}{2} \sum_{i \in \mathcal{N}} |\bar{d}_i|. \tag{19}$$

Equality holds throughout in (19) if and only if $d_i = \bar{d}_i$ for all i with $d_i < 0$. It follows from (16) and (19) that *a FAS will strictly decrease the total absolute deficit*, i.e.

$$\sum_{i \in \mathcal{N}} |\bar{d}_i| < \sum_{i \in \mathcal{N}} |d_i|. \tag{20}$$

It is not necessarily true that the value of the dual functional (5) will be improved by a PAS or FAS. Steps that effect such an improvement are formalized as follows.

Definition 2. Given a pair (f, p) satisfying complementary slackness, an *ascent step* determines a new pair $(\bar{f}, \bar{p})$ satisfying complementary slackness and such that the value of the dual functional is increased, i.e.

$$q(p) < q(\bar{p}).$$

It is not necessarily true that an ascent step is also an adjustment step. Indeed we will show in the next section that a certain type of ascent step, which has been found very effective computationally may increase strictly the total absolute deficit.

We consider the following class of algorithms:

Prototype Primal–Dual Algorithm

Step 0: Choose a starting integer vector pair (f, p) satisfying complementary slackness.

Step 1: Test whether there is a node with positive deficit. If so go to step 2, else terminate.

Step 2: Obtain a new integer vector pair by carrying out either a PAS or a FAS or an ascent step and go to step 1.

Various implementations of adjustment steps and ascent steps that are useful in the context of specific algorithms will be described in the next section. By applying a known primal-dual method we will show in the next proposition that it is always possible to carry out an adjustment step in step 2 of the algorithm, and therefore the algorithm is well defined. We then show in Proposition 2 that under certain assumptions the algorithm always terminates at an optimal pair (f, p) after a finite number of steps. We first recall the definition of an augmenting path:

Definition 3. Given a pair (f, p) satisfying complementary slackness, 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 $m = 1, 2, \ldots, k-1$, 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 $f_{n_{m+1} n_m} > l_{n_{m+1} n_m}$. The scalar

$$\delta = \min\{-d_{n_1}, d_{n_k}, \delta_1, \ldots, \delta_{k-1}\}$$

where

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

will be called the *capacity* of the path.

Given an (f, p) satisfying complementary slackness and an augmenting path $\{n_1, n_2, \ldots, n_k\}$ we can carry out a FAS by increasing (decreasing) the flow by the path capacity δ on each arc of the path of the form (n_m, n_{m+1}) $[(n_{m+1}, n_m)]$. If f is an integer vector then δ will be integer and the new flow vector will also be integer.

Proposition 1. *If an integer vector pair (f, p) satisfies complementary slackness and there exists at least one node with nonzero deficit, then it is possible to obtain a new integer vector pair by carrying out either a FAS or a PAS.*

Proof. The proof is constructive and provides the basic step of an algorithm for solving (MCF) which is in effect a variation of a known primal-dual algorithm given in [6, p. 113]. In the procedure that follows each node is allowed to be in three possible states: (a) *Unlabeled*, (b) *Labeled but Unscanned*, (c) *Labeled and Scanned*. This terminology has been used extensively in Ford and Fulkerson [6] and many other sources (e.g. [11]) and is used in the same manner here. Initially all nodes are unlabeled.

Give the label '0' to a node i with positive deficit.

Until all labels are scanned or a labeled node with negative deficit is found scan the label of a labeled node k as follows:

Give the label 'k' to all unlabeled nodes m such that (m, k) is a balanced arc with $f_{mk} < c_{mk}$, or (k, m) is a balanced arc with $f_{km} > l_{km}$.

Since this algorithm will clearly terminate eventually, there are two possibilities. The first is that a node m with negative deficit will be found in which case it can be seen that by tracing labels an augmenting path can be constructed which starts at m and terminates at i. Therefore a FAS can be carried out by flow augmentation along this path so that the resulting flow vector will be integer.

The second possibility is that each labeled node has a nonnegative deficit. Let L be the set of labeled nodes and $\bar{L}$ be its complement in $\mathcal{N}$ (i.e., $\bar{L} = \mathcal{N} - L$). Since there must exist a node with negative deficit the set $\bar{L}$ is nonempty. It is easily shown that there must exist either an active arc (k, m) with $k \in L$, $m \in \bar{L}$ or an inactive arc (m, k) with $m \in \bar{L}$, $k \in L$. Therefore the scalar δ given by

$$\delta = \min\{\{p_k - a_{km} - p_m \mid k \in L,\ m \in \bar{L},\ (k, m)\text{: active}\},\ \{p_k + a_{mk} - p_m \mid k \in L,\ m \in \bar{L},\ (m, k)\text{: inactive}\}\}$$

is well defined as a positive integer.

Define the new price vector $\bar{p}$ by

$$\bar{p}_k = \begin{cases} p_k - \delta & \text{if } k \in \bar{L}, \\ p_k & \text{if } k \in \bar{L}. \end{cases}$$

It is easily seen that $(f, \bar{p})$ is an integer vector pair that satisfies complementary slackness. Therefore by changing (f, p) to $(f, \bar{p})$ we are carrying out a PAS. □

Progress towards solving (MCF) by the prototype primal-dual algorithm can be measured using two criteria:

(a) The total absolute deficit $\sum_{i\in\mathcal{N}}|d_i|$. If this is reduced to zero the algorithm terminates with an optimal pair (f, p).

(b) The value of the dual functional $q(p)$. If this is increased to its maximum value we will have an optimal p.

An adjustment step will not necessarily improve strictly either criterion—it will not increase the value of criterion (a), while it may degrade criterion (b). An ascent step will improve strictly criterion (b) but may degrade criterion (a). Since the prototype primal-dual algorithm mixes adjustment and ascent steps in arbitrary fashion the type of adjustment steps delineated in the following definition are of interest.

Definition 4. An adjustment step is called *harmless* if the resulting price vector $\bar{p}$ satisfies

$$q(p)\leq q(\bar{p}).$$

Termination of the algorithm at an optimal solution will be shown under the following assumption:

Assumption 1. In the prototype primal-dual algorithm either (a) all steps are adjustment steps, or else (b) all adjustment steps are harmless.

If (a) ((b)) holds in Assumption 1 we are guaranteed that the total absolute deficit (respectively the value of the dual function) will not be degraded at each iteration. However there is no guarantee of strict improvement of either criterion at any iteration. Therefore the assertion of the following proposition is nontrivial.

Proposition 2. *Under Assumption 1 the prototype primal-dual algorithm terminates at an optimal pair (f, p) after a finite number of steps.*

Proof. Each time an ascent step is carried out the dual value is improved by an integer amount. Therefore, if (b) of Assumption 1 holds, it is not possible to carry out an infinite number of ascent steps so after some iteration all steps will be adjustment steps or else the algorithm will terminate finitely. It is therefore sufficient to show the result under the assumption that all steps are adjustment steps.

Each time a FAS is carried out (or more generally when $d_i<\bar{d}_i$ for some i with $d_i<0$) the total absolute deficit $\sum_{i\in\mathcal{N}}|d_i|$ is reduced (cf. (20)) by an integer amount (since the algorithm generates integer vector pairs), while each time a PAS is carried out $\sum_{i\in\mathcal{N}}|d_i|$ is not increased. Therefore it is not possible to carry out an infinite number of steps where $d_i<\bar{d}_i$ for some i with $d_i<0$ during any single execution of the algorithm.

Assume that the algorithm does not terminate in a finite number of steps and generates an infinite sequence of integer vector pairs $\{f^k, p^k\}$. Then it follows that after a finite number of iterations the algorithm will be executing PAS exclusively and, in view of (14)-(17), the deficits of nodes with negative deficit will be constant,

while the deficits of nodes with nonnegative deficit will remain nonnegative. Also by (15) the price of each node with nonnegative deficit will not increase while, by (17), it is impossible for the price of all nodes with positive deficit to remain unchanged. It follows that the set

$$N^\infty = \{i \mid p_k^k \to -\infty\}$$

is nonempty and that for all k sufficiently large we have $0 \leq d_i^k$ for all $i \in N^\infty$, and $0 < d_i^k$ for at least one $i \in N^\infty$. Therefore for some index $\bar{k}$ we have

$$0 < \sum_{i \in N^\infty} d_i^k \quad \forall k > \bar{k}. \tag{21}$$

On the other hand since each (f^k, p^k) satisfies complementary slackness we must have, for all k greater than some index $\tilde{k} \geq \bar{k}$,

$$(m, i) \text{ is active} \quad \text{if } (m, i) \in \mathcal{A},\ m \notin N^\infty,\ i \in N^\infty,$$

$$(i, m) \text{ is inactive} \quad \text{if } (i, m) \in \mathcal{A},\ i \in N^\infty,\ m \notin N^c$$

Therefore we have

$$f_{mi}^k = c_{mi} \quad \text{if } (m, i) \in \mathcal{A},\ m \notin N^\infty,\ i \in N^\infty,$$

$$f_{im}^k = l_{im} \quad \text{if } (i, m) \in \mathcal{A},\ i \in N^\infty,\ m \notin N^\infty,$$

and (21) can be written as

$$0 < \sum_{i \in N^\infty} \left(\sum_{\substack{m \notin N^\infty \\ (i, m) \in \mathcal{A}}} l_{im} - \sum_{\substack{m \notin N^\infty \\ (m, i) \in \mathcal{A}}} c_{mi} \right)$$

This contradicts the fact that there exists a feasible solution for (MCF) and therefore also contradicts our earlier assumption that the algorithm does not terminate in a finite number of steps. □

3. Implementation of adjustment and ascent steps

There is a large variety of procedures for carrying out a FAS, a PAS, or an ascent step and some of these can be combined in a computationally efficient manner to form longer procedures consisting of sequences of steps. We first consider the simplest possible procedures that involve a single node together with its immediate neighbors. The idea here is to select a node i with positive deficit and try to reduce its deficit to zero by changing the flow of its incident arcs and possibly its price p_i. The process is reminiscent of coordinate descent and relaxation methods as explained more fully in [4].

Let (f, p) satisfy complementary slackness. For any node i consider the set of neighbor nodes that can exchange flow with i in a way that d_i is reduced and

complementary slackness is maintained. These are

$$B_i^- = \{k \mid (k, i) \text{ is balanced}, f_{ki} < c_{ki}\}, \tag{22}$$

$$B_i^+ = \{k \mid (i, k) \text{ is balanced}, f_{ik} > l_{ik}\}. \tag{23}$$

If $d_i > 0$ and $d_k < 0$ for any $k \in B_i^- \cup B_i^+$ then it is possible to carry out a FAS by simply increasing (decreasing) the flow on the arc (k, i) $((i, k))$ if $k \in B_i^-$ $(k \in B_i^+)$. This can be done for each node $k \in B_i^- \cup B_i^+$ with $d_k < 0$ until either we exhaust these nodes or we reduce the deficit d_i to zero. We thus arrive at the following procedure:

Single node FAS

Step 0: Choose a node i with $d_i > 0$.

Step 1: Choose a node $k \in B_i^-$ such that $d_k < 0$ and go to step 2, or choose a node $k \in B_i^+$ such that $d_k < 0$ and go to step 3. If no such node can be found terminate.

Step 2: Let

$$\delta = \min\{-d_k, d_i, c_{ki} - f_{ki}\}.$$

Let

$$d_i \leftarrow d_i - \delta, \qquad d_k \leftarrow d_k + \delta, \qquad f_{ki} \leftarrow f_{ki} + \delta.$$

If $d_i = 0$ terminate, else go to step

Step 3: Let

$$\delta = \min\{-d_k, d_i, f_{ik} - l_{ik}\}$$

Let

$$d_i \leftarrow d_i - \delta, \qquad d_k \leftarrow d_k + \delta, \qquad f_{ik} \leftarrow f_{ik} - \delta.$$

If $d_i = 0$ terminate, else go to step

Note that *the single node FAS described above is harmless since it leaves the price vector unchanged.* We next consider the possibility of a PAS involving a price reduction of a single node. For any node i with $d_i \geq 0$ the smallest value that the price p_i can be changed to without violating complementary slackness is[1]

$$\bar{p}_i = \max\{\{p_k - a_{ki} \mid (k, i) \in \mathcal{A}, f_{ki} < c_{ki}\}, \{p_k + a_{ik} \mid (i, k) \in \mathcal{A}, f_{ik} > l_{ik}\}\}. \tag{24}$$

If $d_i > 0$ and $p_i > \bar{p}_i$ then by simply reducing p_i to the level $\bar{p}_i$ we can carry out a PAS. Even if $p_i = \bar{p}_i$ it is still possible to carry out a PAS provided

$$\sum_{k \in B_i^-} (c_{ki} - f_{ki}) + \sum_{k \in B_i^+} (f_{ik} - l_{ik}) \leq d_i. \tag{25}$$

[1] Under unusual circumstances it is possible that for a node i with $d_i \geq 0$ the sets $\{k \mid (k, i) \in \mathcal{A}, f_{ki} < c_{ki}\}$ and $\{k \mid (i, k) \in \mathcal{A}, f_{ik} > l_{ik}\}$ appearing in (24) are both empty in which case the current flows of arcs incident to i are $f_{ki} = c_{ki}$ for all $(k, i) \in \mathcal{A}$ and $f_{ik} = l_{ik}$ for all $(i, k) \in \mathcal{A}$. Then there are two possibilities. Either $d_i > 0$ in which case the problem is infeasible or else $d_i = 0$ and the current flows incident to i are the only ones that are feasible. The first case has been excluded by assumption. In the second case the scalar $\bar{p}_i$ of (24) is defined to be $-\infty$.

This can be done by first increasing f_{ki} to the level c_{ki} for all $k \in B_i^-$, and decreasing f_{ik} to the level l_{ik} for all $k \in B_i^+$. Then in view of (25), we will still have $d_i \geq 0$ and the new deficits will satisfy (14) and (15) in the PAS definition. Furthermore $\bar{p}_i$ as given by (24) will now be reduced to a lower level. By setting p_i to the new value of $\bar{p}_i$ we can carry out a PAS. This process can be repeated as many times as desired as long as (25) holds. Formally we have the following procedure:

Single node PAS

Step 0: Choose a node with $d_i > 0$.

Step 1: Compute $\bar{p}_i$, as given by (24). Set

$$p_i \leftarrow \bar{p}_i.$$

If

$$\sum_{k \in B_i^-} (c_{ki} - f_{ki}) + \sum_{k \in B_i^+} f_{ik} - l_{ik}) \leq d_i \tag{26}$$

go to step 2, else terminate.

Step 2: Let

$$d_i \leftarrow d_i - \sum_{k \in B_i^-} (c_{ki} - f_{ki}) - \sum_{k \in B_i^+} f_{ik} - l_{ik}),$$

$$f_{ki} = c_{ki} \quad \forall k \in B_i^-, \qquad f_{ik} = l_{ik} \quad \forall k \in B_i^+$$

and go to step 1.

It can be easily shown [compare with (5) and Fig.] that *the directional derivative of the dual function $q(p)$ along the direction $\{v \mid v_i = -\ , v_m = 0$ if $m \neq i\}$ corresponding to decreasing p_i is given by*

$$\sum_{\substack{k \\ (i,k)\text{: active}}} c_{ik} + \sum_{\substack{k \\ (i,k)\text{: inactive} \\ \text{or balanced}}} l_{ik} - \sum_{\substack{k \\ (k,i)\text{: active} \\ \text{or balanced}}} c_{ki} - \sum_{\substack{k \\ (k,i)\text{ inactive}}} l_{ki}.$$

This directional derivative can also be written as

$$d_i - \sum_{k \in B_i^-} (c_{ki} - f_{ki}) - \sum_{k \in B_i^+} (f_{ik} - l_{ik}).$$

It follows that *a single node PAS will improve the dual value if and only if strict inequality holds in* (25). *If* (25) *holds as an equation the dual value will be left unchanged while the deficit of i will be reduced to zero.* Also *a single node PAS will decrease the total absolute deficit if and only if some node in $B_i^+ \cup B_i^-$ has negative deficit. Otherwise the total absolute deficit will remain unchanged.* It is therefore possible that an algorithm employing the single node PAS may perform a large number of iterations without changing neither the dual value nor the total absolute deficit. It can be shown by example (compare also with the proof of Proposition 2) that the number of successive such iterations can exceed the number of nodes in the network and indeed can only be bounded by numbers that depend on other problem data such as arc costs and capacities. It is important to note however that extensive computational experimentation has shown that allowing a single node

PAS even if (25) holds as an equation is (at least on the average) computationally beneficial, particularly for certain types of problems such as assignment.

It appears that the only known algorithm for solving (MCF) or a special case thereof that uses the single node PAS procedure is the assignment algorithm proposed by the author in [3]. As shown there experimentally this procedure can be a very powerful device for solving (MCF). Computational results some of which are given in the next section, show also that by simply combining the single node PAS with the multiple node adjustment step of the primal-dual method given in the proof of Proposition 1 one can tremendously improve the performance of the primal-dual algorithm.

If we attempt to carry out a single node PAS at a node i with $d_i > 0$ there are three possibilities:

(a) The PAS is carried out and as a result the new deficit of i is zero in which case we can make no further progress with node i.

(b) The PAS is carried out and as a result the new deficit d_i of i is positive and the new pair (f, p) satisfies (cf. (26))

$$\sum_{k \in B_i^-} (c_{ki} - f_{ki}) + \sum_{k \in B_i^+} (f_{ik} - l_{ik}) > d_i. \tag{27}$$

(c) The PAS cannot be carried out because (27) holds.

If either (b) or (c) occurs we may attempt to follow the single node PAS attempt with a single node FAS. Whether this can be carried out or not we will end up with two possibilities:

(I) $d_i = 0$ in which case no further progress with node i can be made.

(II) $d_i > 0$ and $d_k \geq 0$ for all $k \in B_i^- \cup B_i^+$.

If (II) occurs we can make no further progress on the node i with either a single node FAS or a single node PAS. We thus need a way to perform a FAS or a PAS in the case of a node i such that

$$d_i > 0, \tag{28}$$

$$d_k \geq 0 \quad \forall k \in B_i^- \cup B_i^+, \tag{29}$$

$$\sum_{k \in B_i^-} (c_{ki} - f_{ki}) + \sum_{k \in B_i^+} (f_{ik} - l_{ik}) > d_i.$$

This requires a PAS or FAS involving multiple nodes (i.e., more nodes than just i and its immediate neighbors). One possibility is to use the procedure described in the proof of Proposition 1. However this procedure can be generalized in a direction which is consistent with the philosophy of striving for large price reductions that underlies the idea of a single node PAS.

For a node i satisfying (28)-(30) let T be either the empty set or a subset of $B_i^- \cup B_i^+$ such that

$$\sum_{k \in B_i^- \cap T} (c_{ki} - f_{ki}) + \sum_{k \in B_i^+ \cap T} (f_{ik} - l_{ik}) \leq d_i.$$

Let $\bar{T}$ be the complement of T in $B_i^- \cup B_i^+$, i.e.,

$$\bar{T} = \{k \in B_i^- \cup B_i^+ \mid k \notin T\}. \tag{32}$$

In view of (30) and (31), the set $\bar{T}$ is nonempty. The procedure we describe involves labeling which is similar to the one given in the proof of Proposition 1. However the initial label 'i' will be given only to the subset $\bar{T}$ rather than the entire set $B_i^- \cup B_i^+$.

Multiple node PAS or FAS

Step 0: Choose a node i satisfying (28)–(30) and a subset $T \subset B_i^- \cup B_i^+$ satisfying (31). Give the label '0' to i, and the label 'i' to each node in the set $\bar{T}$ of (32).

Step 1: Choose a node $k \neq i$ with an unscanned label and go to step 2. If no such node can be found go to step 4.

Step 2 (Labeling): Scan the label on the node k by giving the label 'k' to all nodes m that are unlabeled and belong to the set $B_k^- \cup B_k^+$ where

$$B_k^- = \{m \mid (m, k) \text{ is balanced}, f_{mk} < c_{mk}\},$$

$$B_k^+ = \{m \mid (k, m) \text{ is balanced}, f_{km} > l_{km}\}.$$

If for any one of these nodes m we have $d_m < 0$ go to step 3, else go to step 1.

Step 3 (Flow augmentation): An augmenting path has been found which starts at the node m with $d_m < 0$ identified in step 2 and ends at the node i. The path can be constructed by tracing labels starting from m. Let $\delta > 0$ be the capacity of the path. Increase by δ the flow of all arcs on the path that are oriented in the direction from k to i, reduce by δ the flow of all other arcs on the path and terminate.

Step 4 (Price adjustment): Let L be the set of all labeled nodes and $\bar{L}$ be its complement, i.e., $\bar{L} = \mathcal{N} - L$. (Because all nodes in L have nonnegative deficit and $d_i > 0$, there must exist a node with negative deficit and as a result the set $\bar{L}$ is nonempty.) Let

$$\delta = \min\{\{p_k - a_{km} - p_m \mid k \in L, m \in \bar{L}, (k, m)\text{: active}\},$$
$$\{p_k + a_{mk} - p_m \mid k \in L, m \in \bar{L}, (m, k)\text{: inactive}\}\}. \tag{33}$$

(It can be shown easily that the scalar δ is well defined as a positive integer.) Set

$$p_k \leftarrow \bar{p}_k \quad \forall k \in \mathcal{N}$$

where

$$\bar{p}_k = \begin{cases} p_k - \delta & \text{if } k \in L, \\ p_k & \text{if } k \in \bar{L}. \end{cases}$$

Set

$$f_{ki} \leftarrow c_{ki} \quad \forall k \in B_i^- \cap T,$$

$$f_{ik} \leftarrow l_{ik} \quad \forall k \in B_i^+ \cap T$$

and terminate.

Note that in the above procedure termination will occur *either through step* 3 *in which case a FAS will have been carried out, or through step* 4 *in which case* we claim that *a PAS will have been carried out.* Indeed the integer δ of (33) is positive so the new price vector as given by (34) is such that the prices of all nodes with negative deficit will be unchanged while the prices of all nodes in L will have been reduced by a positive amount. Furthermore the deficit of each node $k \in T$ will be increased [in view of (35), (36)], the deficit of d_i will still be nonnegative in view of (31), while the deficit of every other node will remain unchanged. All of this is in agreement with the requirements for a PAS (cf. (14)-(17)). In either case *the multiple node procedure described above is harmless.*

Note also that *if T is empty then the procedure above coincides with the procedure described in the proof of Proposition* 1. It seems however that it is advantageous in some cases to take T nonempty when possible since then there is a tendency for larger price reductions to occur. In the assignment algorithm of [3] T is taken as large as possible.

When strict inequality holds in (25) we showed earlier that the corresponding single node PAS will also be an ascent step. The same is true for a multinode PAS if strict inequality holds in (31) (a fortiori if the set T is taken empty). However it is possible to construct ascent steps of computational interest that are not PAS and may in fact increase the total absolute deficit. In what follows in this section we describe such an ascent step procedure.

Constructing ascent steps

Given a price vector p and a subset of nodes $\mathcal{S} \subset \mathcal{N}$ with $\mathcal{S} \neq \mathcal{N}$ denote

$$C(\mathcal{S}, p) = \sum_{(i,j) \in \mathcal{A}} e_{ij}(\mathcal{S}, p),$$

where, for all (i, j),

$$e_{ij}(\mathcal{S}, p) = \begin{cases} l_{ij} & \text{if } i \in \mathcal{S}, j \notin \mathcal{S}, \text{ and } (i,j) \text{ is inactive or balanced,} \\ -l_{ij} & \text{if } i \notin \mathcal{S}, j \in \mathcal{S}, \text{ and } (i,j) \text{ is inactive,} \\ c_{ij} & \text{if } i \in \mathcal{S}, j \notin \mathcal{S} \text{ and } (i,j) \text{ is active,} \\ -c_{ij} & \text{if } i \notin \mathcal{S}, j \in \mathcal{S}, \text{ and } (i,j) \text{ is active or balanced,} \\ 0 & \text{otherwise.} \end{cases}$$

In words $C(\mathcal{S}, p)$ is the difference of outflow and inflow across $\mathcal{S}$ 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 $\mathcal{S}$ is set to its lower or upper bound depending on whether the arc is going out of $\mathcal{S}$ or coming into $\mathcal{S}$ respectively. For a subset $\mathcal{S}$ consider the vector $v = \{v_i \mid i \in \mathcal{N}\}$ defined by

$$v_i = \begin{cases} 1 & \text{if } i \in \mathcal{S}, \\ 0 & \text{if } i \notin \mathcal{S}. \end{cases}$$

It is easily verified using (5) and Fig. 1 that $C(\mathcal{S}, p)$ *is the directional derivative of the dual functional* q *at* p *along the direction* v. So if

$$C(\mathcal{S}, p) > 0$$

an ascent of the dual functional can be effected by moving from p along the direction v, i.e. by reducing the prices of the nodes in $\mathcal{S}$ by equal amounts while keeping all other prices constant. The following ascent step procedure is based on this fact. The starting point for the procedure is an integer vector pair (f, p) satisfying complementary slackness.

Ascent step procedure

Step 0: Choose a node i with $d_i > 0$. Give to i the label '0'. Set $\mathcal{S} = \emptyset$.

Step 1: Choose a labeled but unscanned node k. Set

$$\mathcal{S} \leftarrow \mathcal{S} \cup \{k\}.$$

Scan the label of the node k by giving the label 'k' to all nodes m that are unlabeled and belong to the set $B_k^- \cup B_k^+$ where

$$B_k^- = \{m \mid (m, k) \text{ is balanced}, f_{mk} < c_{mk}\},$$

$$B_k^+ = \{m \mid (k, m) \text{ is balanced}, f_{km} > l_{km}\}.$$

If $d_m < 0$ for any one of these nodes m and $C(\mathcal{S}, p) \leq 0$ go to step 3, else go to step 2.

Step 2: If

$$C(\mathcal{S}, p) > 0$$

go to step 4, else go to step 1.

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

Step 4 (Ascent step): Let $\bar{\mathcal{S}}$ be the complement of $\mathcal{S}$, i.e. $\bar{\mathcal{S}} = \mathcal{N} - \mathcal{S}$. (Because $d_i > 0$ and all nodes in $\mathcal{S}$ have nonnegative deficit, there must exist a node with negative deficit in $\bar{\mathcal{S}}$ and as a result the set $\bar{\mathcal{S}}$ is nonempty.) Let

$$\delta = \min\{\{p_k - a_{km} - p_m \mid h \in \mathcal{S}, m \in \bar{\mathcal{S}}, (k, m)\text{: active}\},$$

$$\{p_k + a_{mk} - p_m \mid k \in \mathcal{S}, m \in \bar{\mathcal{S}}, (m, k)\text{: inactive}\}\}.$$

Set

$$f_{mk} \leftarrow c_{mk} \quad \text{if } k \in \mathcal{S}, m \in \bar{\mathcal{S}}, m \text{ is labeled and } (m, k) \text{ is balanced},$$

$$f_{km} \leftarrow l_{km} \quad \text{if } k \in \mathcal{S}, m \in \bar{\mathcal{S}}, m \text{ is labeled and } (k, m) \text{ is balanced}.$$

Set

$$p_k \leftarrow \bar{p}_k \quad \forall k \in \mathcal{N}$$

where

$$\bar{p}_k = \begin{cases} p_k - \delta & \text{if } k \in \mathcal{S}, \\ p_k & \text{if } k \in \bar{\mathcal{S}}, \end{cases}$$

and terminate.

The procedure above terminates either via step 3 in which case it generates a harmless FAS, or via step 4 in which case it generates an ascent step. In order for the procedure to be well defined, however, we must show that whenever we return to step 1 from step 2 there is still left a labeled node with an unscanned label. Indeed when all node labels are scanned (i.e. the set $\mathcal{S}$ coincides with the labeled set L) there is no balanced arc (m, k) such that $m \notin \mathcal{S}$, $k \in \mathcal{S}$ and $f_{mk} < c_{mk}$ or a balanced arc (k, m) such that $k \in \mathcal{S}$, $m \notin \mathcal{S}$ and $f_{km} > l_{km}$. It follows from the definition (37), (38) that

$$C(\mathcal{S}, p) = \sum_{k \in \mathcal{S}} d_k > 0.$$

Therefore in this case the procedure identifies an ascent direction and switches from step 2 to step 4 rather than switch back to step 1.

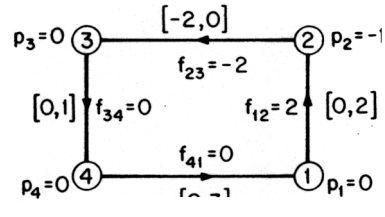
It can be seen that the ascent step procedure involves a comparable amount of computation per node labeled as the multinode PAS-FAS once it is realized that the quantity $C(\mathcal{S}, p)$ in step 2 can be computed recursively rather than recomputed each time the set $\mathcal{S}$ is enlarged in step 1. However this procedure tends to terminate earlier since the final set of labeled nodes may be considerably smaller than the corresponding set L of the multinode PAS-FAS procedure. According to our experience this fact typically results in substantial computational savings.

We note that a similar ascent step procedure can be constructed starting from a node with negative deficit. The straightforward details are left to the reader. Computational experience has shown that it is beneficial to initiate the ascent procedure from nodes with both positive and negative deficit.

An important difference from the adjustment steps described earlier as well as operations of the primal-dual and out-of-kilter methods (see the next section) is that *when the ascent procedure terminates via step* 4 *the total absolute deficit may increase strictly* as we now show by example:

Example. Consider the four node, four arc network shown in Fig. 2. All arc costs are zero and the upper and lower bound constraints are

$$0 \leq f_{12} \leq 2, \quad -2 \leq f_{23} \leq 0, \quad 0 \leq f_{34} \leq 1, \quad 0 \leq f_{41} \leq 3.$$



Consider the pair (f, p)

$$p_1 = p_3 = p_4 = 0, \quad p_2 = -1, \quad f_{12} = 2, \quad f_{23} = -2, \quad f_{34} = f_{41} = 0$$

satisfying complementary slackness, and the corresponding node deficits

$$d_1 = 2, \quad d_2 = -4, \quad d_3 = 2, \quad d_4 = 0.$$

Apply the ascent step procedure starting from node 1. The initial set $\mathcal{S}$ is $\{1\}$ but $C(\{1\}, p) = -1$ (cf. (37)) so an ascent step is not possible by reducing p_1. Node 4 will be the only one labeled from node 1, the set $\mathcal{S}$ will be enlarged in Step 2 and become $\mathcal{S} = \{1, 4\}$. We have $C(\{1, 4\}, p) = 1 > 0$ so an ascent step will be performed in Step 4 by reducing the prices of nodes 1 and 4 from 0 to -1. The resulting flows will be

$$f_{12} = 2, \quad f_{23} = -2, \quad f_{34} = 1, \quad f_{41} = 0$$

as shown in Fig. 3. The corresponding node deficits are

$$d_1 = 2, \quad d_2 = -4, \quad d_3 = 3, \quad d_4 = -1$$

so the total absolute deficit has degraded from 8 to 10, while a straight-forward calculation shows that the dual value has improved from -4 to -3.

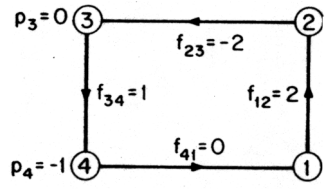


Fig. 3.

4. Algorithms and computational experience

Several procedures for carrying out adjustment steps and ascent steps were described in the previous section and they can be combined in different ways to give a variety of algorithms some of which are known and some of which are new and apparently cannot be embedded within the framework of the most general known primal-dual method—the out-of-kilter method. The purpose of this section is to consider these new algorithms and compare them experimentally with classical primal-dual, variants of out-of-kilter and primal simplex methods. We refer to [4] for a detailed description of the codes involved and our experimental conditions. The test problems used are the standard 40 benchmarks generated by the publicly available NETGEN program [9]. In addition to our own FORTRAN codes (written in collaboration with Paul Tseng [4] and briefly described below) we have made comparisons under identical test conditions with the primal-dual code KILTER

(Aashtiani and Magnanti [1]) and the primal simplex code RNET (Grigoriadis and Hsu [7]). Among presently available network codes written in FORTRAN, it appears that RNET has produced the fastest computation times for the NETGEN benchmarks. This is based on the computation times given by Grigoriadis [7] and on detailed comparisons with other network codes given in our own report [4]. As additional evidence we note that Kennington and Helgason in their 1980 book [8] compare RNET with their own primal simplex code NETFLO on the first 35 NETGEN benchmarks and conclude that 'RNET... produced the shortest times that we have seen on these 35 test problems" ([8, p. 255]). Our computational results are given in Tables 1 and 2.

(a) *Relaxation method (code RELAX)*. This method and corresponding code is described in detail in [4], where its conceptual similarity with classical coordinate descent and relaxation methods is discussed in detail. At each iteration a node with nonzero deficit is selected. A single node PAS or FAS as described in Section 3 is attempted. If one of the two can be carried out the iteration terminates. Otherwise an ascent step procedure as described in Section 3 is carried out. This method (which is new and apparently cannot be embedded in any way within the framework of the out-of-kilter method) seems to be the fastest general purpose method for network problems according to the results of Table 1. (*Note added in proof*: While the paper was under review an enhanced version of RELAX was developed which solves the problems of Table 1 in time that is roughly 20% faster than the one reported in Table 1. Also the times reported in Table 1 refer to the FORTRAN compiler of the VMS operating system version 3.7. The compiler of version 4.0 (released while the paper was under review) produces code that runs substantially faster than the one corresponding to version 3.7.)

(b) *Assignment-Relaxation Method (code ASSIGN)*. This method solves the assignment problem and is basically the one given in [3]. At each iteration an unassigned source is selected and a single node PAS or FAS as described in Section 3 (see also [3]) is attempted. If one of the two can be carried out the iteration terminates. Otherwise the multiple node PAS or FAS described in Section 3 is carried out with the set T consisting of a single node. As evidenced by Table 2, the solution time of the ASSIGN code is faster by a factor of over 10 than the fastest times reported to our knowledge so far on the five NETGEN assignment benchmark problems (Problems 11 to 15 in Table 1).

(c) *Primal-Dual Method (code PDUAL)*. This method is the classical primal-dual method described in the proof of Proposition 1 except that at each iteration the label '0' is given to all nodes with positive deficit rather than a single node. Furthermore if a PAS results the iteration is continued labeling further nodes until (perhaps after several PAS) a node with negative deficit is found and a FAS is carried out. The code PDUAL implements this method using very similar data structures and coding techniques as the RELAX code. The results of Table 1 show that this code is substantially inferior to both RELAX and KILTER. (Actually KILTER implements a very similar method but uses a sophisticated labeling scheme

Table 1

Standard benchmark problems 1-40 of [9] obtained using NETGEN. All times are in secs on a VAX 11/750 obtained under identical test conditions. All codes are in standard FORTRAN compiled under VMS version 3.7 in the OPTIMIZE mode.

Problem type	Problem No.	No of nodes	No. of Arcs	RELAX	RNET	KILTER	PDUAL	VKILTER
Transportation	1	200	1300	2.29	3.11	8.81	16.05	2.77
	2	200	1500	2.52	3.68	9.04	15.98	3.18
		200	2000	2.45	4.27	9.22	20.35	4.98
		200	2200	3.21	4.95	10.45	19.39	5.07
		200	2900	3.21	7.12	16.48	22.88	5.25
		300	3150	5.13	9.16	25.08	43.99	8.54
		300	4500	7.35	12.61	35.55	55.01	10.17
	8	300	5155	5.04	14.73	46.30	53.77	13.04
	9	300	6075	7.87	18.57	43.12	62.32	11.30
	10	300	6300	6.14	16.10	47.80	57.97	10.89
Total (Problems 1-10)				45.22	94.30	251.85	367.71	75.19
Assignment	11	400	1500	1.75	4.79	8.09	11.20	5.27
		400	2250	1.90	6.54	10.76	14.49	7.03
		400	3000	2.60	8.50	8.99	15.77	5.82
	14	400	3750	3.04	9.56	14.52	13.92	8.28
	15	400	4500	4.73	9.82	14.53	16.22	8.30
Total (Problems 11-15)				14.02	39.21	56.89	71.60	34.70
Uncapacitated	16	400	1306	4.36	2.72	13.57	16.71	6.05
and lightly	17	400	2443	3.53	3.38	16.89	23.02	5.18
capacitated	18	400	1306	3.95	2.59	13.05	16.50	6.01
problems	19	400	2443	3.66	3.55	17.21	21.97	4.38
	20	400	1416	5.06	2.97	11.88	22.68	6.10
	21	400	2836	5.17	4.38	19.06	33.65	13.61
	22	400	1416	5.09	2.84	12.14	19.42	3.87
	23	400	2836	5.95	4.50	19.65	30.32	10.57
	24	400	1382	2.27	2.66	13.07	14.68	2.08
	25	400	2676	3.24	5.76	26.17	25.06	4.72
	26	400	1382	2.14	2.39	11.31	10.78	3.38
	27	400	2676	2.85	3.47	18.88	15.39	5.35
	28	1000	2900	6.00	8.39	29.77	47.66	8.27
	29	1000	3400	6.97	11.87	32.36	50.36	11.85
	30	1000	4400	13.39	11.08	42.21	49.89	13.23
	31	1000	4800	11.57	10.33	39.11	48.94	11.55
	32	1500	4342	11.47	18.22	69.28	81.65	20.79
	33	1500	4385	17.71	17.12	63.59	91.91	15.30
	34	1500	5107	12.74	20.29	72.51	94.49	22.53
	35	1500	5730	11.38	18.15	67.49	104.42	23.78
Total (Problems 16-35)				138.50	156.66	609.20	819.50	209.08
Large	36	8000	15,000	397.57	270.77	1074.76		
uncapacitated	37	5000	23,000	294.68	280.79	681.94		
problems	38	3000	35,000	170.48	269.85	607.89		
	30	5000	15,000	180.48	149.51	558.60		
	40	3000	23,000	81.75	171.02	369.40		
Total (Problems 36-40)				1,124.96	1,141.94	3,292.59		

Table 2

Benchmark assignment problems generated by NETGEN. Same as problems 11-15 of Table 1. All times in secs as follows:
ASSIGN: Our times on IBM 370/168, FORTRAN, OPT = 2.
RNET: Times on IBM 370/168, FORTRAN, OPT = 2 from [7].
AP-AB: Specialized primal simplex assignment code of [2]. Times on CDC 6600, RUN from [2].
PDAC: Specialized primal-dual assignment code of [12]. Times on CYBER 70/74, FTN from [12]

Problem No.	ASSIGN	RNET	AP-AB	PDAC
	0.05	0.38	0.97	1.37
	0.07	0.69	1.14	1.42
	0.06	0.96	1.48	2.60
	0.08	1.02	1.61	2.79
	0.13	0.93	1.68	3.98
Total	0.39	3.98	6.88	12.16

whereby labels are preserved from one iteration to the next. This accounts for the superiority of KILTER over PDUAL). We note also that another implementation of the primal-dual method described in [4] (it is called there PDUAL2) gave comparable computational results. It is evident from the results of Table 1 and those of [4] that the relaxation method is far superior to classical primal-dual methods such as the one described in the proof of Proposition 1.

(d) *A Variant of the Out-of-Kilter Method (code VKILTER).* In the standard implementation of the out-of-kilter method (see Lawler [11, p. 142]) the deficits of all nodes are kept at zero and price and flow adjustments are effected so as to reduce deviations of arcs from complementary slackness. In our prototype primal-dual algorithm complementary slackness is maintained so the adjustment steps described in the previous section cannot be embedded within the framework of the out-of-kilter method mentioned above. However it is possible to apply the out-of-kilter method on a modified but equivalent problem whereby the single node PAS and FAS, and the multiple node PAS or FAS can be almost (but not quite) embedded within the framework of operations of the out-of-kilter method. This possibility, which was suggested by an anonymous referee, will now be described.

Suppose that the network is enlarged by addition of a dummy node '0' and a set of dummy arcs (0, i) one for every node $i \in \mathcal{N}$. The cost and upper and lower bounds of each dummy arc are zero. If complementary slackness is maintained on the nondummy arcs, the only arcs that can be out-of-kilter are dummy arcs and the deficit d_i of a node i is equal to the flow of the dummy arc $(0, i)$. The kilter number of dummy arc $(0, i)$ is the absolute deficit $|d_i|$ while the total kilter number is $\sum_{i \in \mathcal{N}} |d_i|$. A single node FAS reduces the total kilter number and can be viewed as a standard flow adjustment operation of the out-of-kilter method applied to the enlarged

network. A single node PAS at a node i can be interpreted as a flow adjustment operation followed by a price adjustment operation of the out-of-kilter algorithm in the case where after the flow adjustment takes place the deficit of node i is nonzero and no further nodes can be labeled from i. This will happen if and only if (25) holds with strict inequality. However if equality holds in (25) the single node PAS will reduce the deficit of the starting node to zero while possibly leaving $\sum_{i\in\mathcal{N}} |d_i|$ unchanged, so it cannot be viewed as a standard operation of the out-of-kilter method applied to the enlarged network. A similar situation occurs if equality holds in (31); a multiple node PAS may leave the total absolute deficit $\sum_{i\in\mathcal{N}} |d_i|$ unchanged and set the deficit of the starting node to zero. Note also that as shown in the previous section *the ascent step procedure* can actually increase the total absolute deficit so it *is totally incompatible with the out-of-kilter method operations.*

The question arises whether the seemingly minor differences in the adjustment steps described above are computationally significant. To answer this we implemented a modification of the single node PAS and multiple node PAS-FAS procedures so that a single node PAS is carried out as long as (26) holds with strict inequality while the set T in (31) is taken empty (this guarantees that strict inequality holds in (31)). We implemented an algorithm which is operated as follows: At each iteration a node i with nonzero deficit is chosen. If a single node FAS or PAS (modified as described above) is carried out the iteration terminates. Otherwise a multiple node PAS or FAS (with the set T in (31) taken empty) is carried out starting from i. If as a result we obtain a PAS the labeling process is continued until (after perhaps several additional PAS) an FAS is obtained similarly as in the classical primal-dual and out-of-kilter methods.

The corresponding code, called VKILTER, differs by only a few FORTRAN lines from the RELAX code. It may be viewed as an (apparently unreported thus far) implementation of the out-of-kilter method applied to the enlarged network described earlier. The results of Table 1 show that VKILTER is substantially outperformed by RELAX. However VKILTER is much faster than both KILTER and PDUAL, and indeed outperforms RNET on transportation and assignment problems. This is surprising in view of the widely held opinion that the best primal simplex codes outperform the best primal-dual and out-of-kilter codes on network flow problems. Apparently the full potential of the out-of-kilter method has not been realized as yet. The only explanation that we can give for the superior performance of VKILTER over KILTER and PDUAL is the use of the single node adjustment steps which are apparently very effective while requiring very low computational overhead.

References

[1] H.A. Aashtiani and T.L. Magnanti, "Implementing primal-dual network flow algorithms' Operations Research Center Report 055-76, Massachusetts Institute of Technology, June 1976.

[2] R.S. Barr, F. Glover and D. Klingman, "The alternative basis algorithm for assignment problems", *Mathematical Programming* 13 (1977) 1.

[3] D.P. Bertsekas, "A new algorithm for the assignment problem", *Mathematical Programming* 21 (1981) 153-171.

[4] D.P. Bertsekas and P. Tseng, "Relaxation methods for minimum cost network flow problems" Laboratory for Information & Decision Systems Report LIDS-P-1339, Massachusetts Institute of Technology, October 1983.

[5] G.B. Dantzig, *Linear programming and extensions* (Princeton University Press, Princeton, New Jersey, 1963).

[6] L.R. Ford, Jr. and D.R. Fulkerson, *Flows in networks* (Princeton University Press, Princeton, New Jersey, 1962).

[7] M.D. Grigoriadis, "Algorithms for the minimum cost single and multi-commodity network flow problems", Notes for Summer School in Combinatorial Optimization SOGESTA, Urbino, Italy, July 1978.

[8] J. Kennington and R. Helgason, *Algorithms for network programming*, (Wiley, New York, 1980).

[9] D. Klingman, A Napier, and J. Stutz, "NETGEN—a program for generating large scale (un)capacitated assignment, transportation and minimum cost flow network problems", *Management Science* 20 (1974) 814-822.

[10] H.W. Kuhn, "The Hungarian method for the assignment problem", *Naval Research Logistics Quarterly* 2 (1955) 83-97.

[11] E.L. Lawler, *Combinatorial optimization: Networks and matroids* (Holt, Rinehart & Winston, New York, 1976).

[12] L.F. McGinnis, "Implementation and testing of a primal-dual algorithm for the assignment problem", *Operations Research* 31 (1983) 277-291.

[13] C.H. Papadimitriou and K. Steiglitz, *Combinatorial optimization: Algorithms and complexity* (Prentice Hall, Englewood Cliffs, New Jersey, 1982).

[14] R.T. Rockafellar, *Convex analysis* (Princeton University Press, Princeton, New Jersey, 1970).

[15] R.T. Rockafellar, *Network flows and monotropic programming* (Wiley, New York, 1984).

[16] R.T. Rockafellar, "Monotropic programming: Descent algorithms and duality", O.L. Mangasarian, R. Meyer and S. Robinson, eds., *Nonlinear programming 4* (Academic Press, New York, 1981, 327-366).

[17] T.E. Stern, "A class of decentralized routing algorithms using relaxation", *IEEE Transactions on Communications* COM-25 (1977) 1092-1102.