Networks with Side Constraints: An LU Factorization Update

Richard S. Barr, Keyvan Farhangian, Jeffery L. Kennington

An important class of mathematical programming models which are frequently used in logistics studies is the model of a network problem having additional linear constraints. A specialization of the primal simplex algorithm which exploits the network structure can be applied to this problem class. This specialization maintains the basis as a rooted spanning tree and a general matrix called the working basis. This paper presents the algorithms which may be used to maintain the inverse of this working basis as an LU factorization, which is the industry standard for general linear programming software. Our specialized code exploits not only the network structure but also the sparsity characteristics of the working basis. Computational experimentation indicates that our LU implementation results in a 50 percent savings in the non-zero elements in the eta file, and our computer codes are approximately twice as fast as MINOS and XMP on a set of randomly generated multicommodity network flow problems.

ACKNOWLEDGEMENT

This research was supported in part by the Department of Defense under Contract Number MDA903-82-C-0440 and the Air Force Office of Scientific Research under Contract Number AFOSR 83-0278.

Good software for solving linear programming models is one of the most important tools available to the logistics engineer. For logistics studies, these linear programs frequently involve a very large network of nodes and arcs, which may be duplicated by time period. For example, nodes may represent given cities at a particular point in time while arcs represent roads, railways, and legs of flights connecting these cities. Some nodes are designated as supply nodes, others demand nodes, while some may simply represent points of transshipment. The mathematical model characterizes a solution such that the supply is shipped to the demand nodes at least cost while not violating either the upper or lower bounds on the flow over an arc.

If the main structure of a logistics problem can be captured in a network model, then the size of solvable problems becomes enormous. Hence, more realistic situations can be modelled that would otherwise lie outside the domain of general linear programming techniques. For example, one current logistics planning model involves 200 nodes and (365 days/yr) (30 years) = 10,950 time periods to give over 2,000,000 constraints. Network problems having 20,000 constraints and 20,000,000 variables are solved routinely at the U. S. Treasury Department.

Unfortunately, the pure network structure may require simplification of the problem to the point that key policy restrictions must be omitted. The work presented in this study builds upon existing large-scale network solution technology to allow for the inclusion of arbitrary additional constraints. Typical constraints include capacities on vehicles carrying different types of goods, restrictions on the total number of vehicles available for assignment, and budget restrictions. The addition of even a few non-network constraints can greatly enhance the realism and usability of these models. Our approach exploits—to as great an extent as possible—the traditional network portion of the problem while simultaneously enforcing any additional restrictions imposed by the practitioner.

For general linear programming systems, the most important component is the algorithm used to update the basis inverse. Due to the excellent sparcity and numerical stability characteristics, an LU factorization with either a Bartels-Golub or Forrest-Tomlin update has been adopted for modern linear programming systems. For pure network problems, the basis is always triangular and corresponds to a rooted spanning tree. The modern network codes which exploit this structure have been found to be from one to two orders of magnitude faster than the general linear programming systems. In this paper, we have combined these two powerful techniques into an algorithm for solving network models having additional side constraints.

Let A be an $\bar{m} \times \bar{n}$ matrix, let **c** and **u** be $\bar{n}$-component vectors, and let **b** be an $\bar{m}$-component vector. Without loss of generality, the **linear program** may be stated mathematically as follows:

$$\text{minimize} \qquad \mathbf{cx} \tag{1}$$

$$\text{subject to:} \qquad A\mathbf{x} = \mathbf{b} \tag{2}$$

$$\mathbf{0} \le \mathbf{x} \le \mathbf{u}. \tag{3}$$

The **network with side constraint model** is a special case of (1)–(3) in which A takes the form

$$A = \left[\begin{array}{c|c} M & \\ \hline S & P \end{array}\right] \begin{array}{l} \}\, n \\ \}\, m \end{array}$$

where M is a node-arc incidence matrix.
If $m = 0$, then (1) – (3) is a pure network problem.

1.1 Applications

There are numerous applications of the network with side constraint model. Professor Glover and his colleagues have solved a large passenger-mix model for Frontier Airlines and a large land management model for the Bureau of Land Management (see [7, 8]). A world grain export model has been solved to help analyze the port capacity of U. S. ports during the next decade (see [2]). A cargo routing model is being used by the Air Force Logistics Command to assist in routing cargo planes for the distribution of serviceable spares (see [1]). Lt. Col. Dennis McLain, has developed a large model to assist in the development of a casualty evacuation plan in the event of a European conflict (see [14]). A National Forest Management Model has been developed to aid forest managers in long term planning for national forests (see [10]). In addition, work is currently underway which attempts to convert general linear programs into the network with side constraint model (see [4, 16]).

1.2 Objective of Investigation

Due to both storage and time considerations, the basis inverse is maintained as an LU factorization in modern LP software (see [3, 5, 15]). The objective of this investigation is to extend these ideas to the primal partitioning algorithm when applied to the network with side constraints model.

1.3 Notation

The i^{th}component of the vector **a** will be denoted by a_i. The $(i,j)^{th}$ element of the matrix A is denoted by A_{ij}. A(i) and A[i] denotes the i^{th} column and i^{th} row of the matrix A, respectively. **0** denotes a vector of zeroes, **1** denotes a vector of ones, and $\mathbf{e}^k$ denotes a vector with a 1 in the k^{th} position and zeroes elsewhere. Sigma is used to denote the scalar signum function defined by

$$\sigma(y) = \begin{cases} 1, & \text{if } y > 0 \\ 0, & \text{if } y = 0 \\ -1, & \text{if } y < 0. \end{cases}$$

The identity matrix is given by "I".

II. THE PRIMAL SIMPLEX ALGORITHM

We assume that A has full row rank and that there exist a feasible solution for (1)–(3). Given a basic feasible solution, we may partition A, $\mathbf{c}$, $\mathbf{x}$, and $\mathbf{u}$ into basic and nonbasic components, that is, $A = [B\,|\,N]$, $c = [\mathbf{c}^B|\mathbf{c}^N]$, $\mathbf{x} = [\mathbf{x}^B|\mathbf{x}^N]$, and $\mathbf{u} = [\mathbf{u}^B|\mathbf{u}^N]$. Using the above partitioning, the primal simplex algorithm may be stated as follows:

PRIMAL SIMPLEX ALGORITHM

0. *Initialization*. Let $[\mathbf{x}^B|\mathbf{x}^N]$ be a basic feasible solution.
1. *Pricing*. Let $\boldsymbol{\pi} = \mathbf{c}^B B^{-1}$. Define

   $\psi_1 = \{i : x_i^N = 0 \text{ and } \boldsymbol{\pi} N(i) > c_i^N\}$,
   $\psi_2 = \{i : x_i^N = u_i^N \text{ and } \boldsymbol{\pi} N(i) < c_i^N\}$.

   If $\psi_1 \cup \psi_2 = \emptyset$, terminate with $[\mathbf{x}^B|\mathbf{x}^N]$ optimal; otherwise, select $k \in \psi_1 \cup \psi_1$ and set $\delta \leftarrow 1$ if $k \in \psi_1$ and $\delta \leftarrow -1$, otherwise.
2. *Ratio Test*. Set $\mathbf{y} \leftarrow B^{-1}N(k)$. Set

   $$\Delta_1 \leftarrow \underset{\sigma(y_j) = \delta}{\min}\left\{\frac{x_j^B}{|y_j|}, \infty\right\}$$

   $$\Delta_2 \leftarrow \underset{-\sigma(y_j) = \delta}{\min}\left\{\frac{u_j^B - x_j^B}{|y_j|}, \infty\right\}$$

   Set $\Delta \leftarrow \min\{\Delta_1, \Delta_2, u_k^N\}$.
   If $\Delta \neq \infty$, then go to 3; otherwise, terminate with the conclusion that the problem is unbounded.
3. *Update Values*. Set $x_k^N \leftarrow x_k^N + \Delta\delta$ and $\mathbf{x}^B \leftarrow \mathbf{x}^B - \Delta\delta\mathbf{y}$. If $\Delta = u_k^N$, return to step 1.
4. *Update Basis Inverse*. Let

   $\psi_3 = \{j : x_j^B = 0 \text{ and } \sigma(y_j) = \delta\}$
   $\psi_4 = \{j : x_j^B = u_j^B \text{ and } -\sigma(y_j) = \delta\}$.

   Select any $\ell \in \psi_3 \cup \psi_4$. In the basis, replace $B(\ell)$ with $N(k)$, update the inverse of the new basis, and return to step 1.

III. THE PARTITIONED BASIS

The network with side constraint model may be stated as follows:

minimize	$\mathbf{c}^1\mathbf{x}^1 + \mathbf{c}^2\mathbf{x}^2$	(4)
subject to:	$M\mathbf{x}^1 = \mathbf{b}^1$	(5)
	$S\mathbf{x}^1 + P\mathbf{x}^2 = \mathbf{b}^2$	(6)
	$\mathbf{0} \le \mathbf{x}^1 \le \mathbf{u}^1$	(7)
	$\mathbf{0} \le \mathbf{x}^2 \le \mathbf{u}^2.$	(8)

We may assume without loss of generality that,

(i) The graph associated with M has n nodes and is connected (i.e., there exists an undirected path between every pair of nodes).

(ii) $[S \mid P]$ has full row rank (i.e., rank $[S \mid P] = m$).

(iii) Total supply equals total demand (i.e., $\mathbf{1b}^1 = 0$).

Since the rank of system (5) is one less than the number of rows, we add what has been called the root arc to (5) to obtain

$$M\mathbf{x}^1 + \mathbf{e}^p a = \mathbf{b}^1$$

where $0 \le a \le 0$ and $1 \le p \le n$.

Then the constraint matrix for the network with side constraints model becomes

$$A = \left[\begin{array}{c|c|c} M & & \mathbf{e}^p \\ \hline S & P & \end{array}\right]$$

and A has full row rank

It is well-known that every basis for A may be placed in the form

$$B = \left[\begin{array}{c|c} T & C \\ \hline D & F \end{array}\right] \tag{9}$$

where T corresponds to a rooted spanning tree and

$$B^{-1} = \left[\begin{array}{c|c} T^{-1} + T^{-1}CQ^{-1}DT^{-1} & -T^{-1}CQ^{-1} \\ \hline -Q^{-1}DT^{-1} & Q^{-1} \end{array}\right] \tag{10}$$

where $Q = F - DT^{-1}C$. The objective of this paper is to give algorithms which maintain Q^{-1} as an LU factorization.

IV. THE INVERSE UPDATE

Recall that the partitioned basis takes the form

$$B = \left[\begin{array}{c|c} \overbrace{T}^{\text{key}} & \overbrace{C}^{\text{nonkey}} \\ \hline D & F \end{array}\right].$$

Let

$$L = \left[\begin{array}{c|c} T^{-1} & -T^{-1}C \\ \hline & I \end{array}\right]$$

and let

$$\overline{B} = BL = \left[\begin{array}{c|c} I & \\ \hline DT^{-1} & Q \end{array}\right].$$

The inverse update requires a technique for obtaining a new Q^{-1} after a basis exchange. Let $\overline{B}_i$, L_i, B_i, and Q_i denote the above matrices at iteration i. Then we want an expression for Q_{i+1}^{-1} in terms of Q_i^{-1}. The transformation takes the form

$$B_{i+1}^{-1} = EB_i^{-1} \tag{11}$$

where E is either an elementary column matrix or a permutation matrix. Let E be partitioned to be compatible with B. That is,

$$E = \begin{array}{c} \left[\begin{array}{c|c} E_1 & E_2 \\ \hline \underbrace{E_3}_{n} & \underbrace{E_4}_{m} \end{array}\right] \end{array} \begin{array}{l} \}n \\ .\}m \end{array}$$

By examining the (2,2) partition of $\overline{B}_{i+1}^{-1}$, we obtain

$$Q_{i+1}^{-1} = (E_4 - E_3T^{-1}C)Q_i^{-1} \tag{12}$$

In determining the updating formulae, we must examine two major cases with subcases.

Case 1. The leaving column is nonkey. For this case, E takes the form

$$\left[\begin{array}{c|c} 1 & E_2 \\ \hline & E_4 \end{array}\right].$$

and (12) reduces to $Q_{i+1}^{-1} = E_4Q_i^{-1}$.

Case 2. The leaving column is key.

Let $\boldsymbol{\gamma} = \mathbf{e}^j\, T^{-1}\, C$. If $\gamma_k \neq 0$, then the k^{th} column of C can be interchanged with the j^{th} column of T and the new T will be nonsingular.

Subcase 2a. $\boldsymbol{\gamma} \neq \mathbf{0}$. Suppose $\gamma_k \neq 0$.

Then $E_4 - E_3\, T^{-1}\, C$ reduces to

$$R = \left[\begin{array}{ccc} I & & \\ \hline & -\mathbf{e}^jT^{-1}C & \\ \hline & & I \end{array}\right] \begin{array}{l} \\ \leftarrow \text{row } j \\ \text{, and} \end{array} \tag{13}$$

$Q_{i+1}^{-1} = RQ_i^{-1}$. Case 1 is applied to complete the update.

Subcase 2b. $\boldsymbol{\gamma} = \mathbf{0}$. For this case no interchange is possible, the entering column becomes key, and $Q_{i+1}^{-1} = Q_i^{-1}$.

V. AN LU UPDATE

Let

$$
U^i = \left[\begin{array}{c|c|c}
I & \begin{array}{c}\mu_1 \\ \vdots \\ \mu_{i-1}\end{array} & 0 \\
\hline
0 & 1 & \\
\cline{2-3}
 & & I
\end{array}\right],
$$

and

$$
L^i = \left[\begin{array}{c|c|c}
I & & 0 \\
\hline
 & \ell_i & \\
\cline{2-3}
0 & \begin{array}{c}\ell_{i+1} \\ \vdots \\ \ell_m\end{array} & I
\end{array}\right].
$$

Matrices of the form given by U^i and L^i are called upper etas and lower etas, respectively. Suppose we have a factorization of Q^{-1} in the form

$$
Q^{-1} = U^1U^2 \dots U^mF^sF^{s-1} \dots F^1, \tag{14}
$$

where $F^1, \dots, F^5$ are a combination of row and column etas. The right side of (14) is referred to as the eta file where only the non-identity rows and columns are stored. Suppose that the k^{th} column of Q is replaced by $\widehat{Q}(k)$ to form the new m by m working basis $\widehat{Q}$. This section presents algorithms which may be used to update (14) to produce $\widehat{Q}^{-1}$ in the same form.

5.1 Nonkey Column Leaves The Basis

If $k = m$, then let $\boldsymbol{\beta} = F^s \dots F^1\widehat{Q}(k)$, let

$$
\tilde{L}^m = \left[\begin{array}{c|c}
I & \\
\hline
 & 1/\beta_m
\end{array}\right],
$$

and let

$$
\tilde{U}^m = \left[\begin{array}{c|c}
I & \begin{array}{c}-\beta_1 \\ \vdots \\ -\beta_{m-1}\end{array} \\
\hline
 & 1
\end{array}\right].
$$

We will show that $\widehat{Q}^{-1} = U^1 \ldots U^{m-1}\widetilde{U}^m\widetilde{L}^m F^s \ldots F^1$.

If $k < m$, then let $R^k = I$ and

$$Q^{-1} = U^1 \ldots U^k R^k U^{k+1} \ldots U^m F^s \ldots F^1. \tag{15}$$

We next define a new upper eta, $\widetilde{U}^k$, and a new row eta, R^{k+1}, such that

$$R^k U^{k+1} = \widetilde{U}^k R^{k+1}. \tag{16}$$

Substituting (16) into (15) yields

$$Q^{-1} = U^1 \ldots U^k \widetilde{U}^k R^{k+1} U^{k+2} \ldots U^m F^s \ldots F^1. \tag{17}$$

We again define two new eta's, $\widetilde{U}^{k+1}$ and R^{k+2}, such that

$$R^{k+1} U^{k+2} = \widetilde{U}^{k+1} R^{k+2}. \tag{18}$$

Substituting (18) into (17) yields

$$Q^{-1} = U^1 \ldots U^k \widetilde{U}^k \widetilde{U}^{k+1} R^{k+2} U^{k+3} \ldots U^m F^s \ldots F^1.$$

Repeating this process eventually yields

$$Q^{-1} = U^1 \ldots U^k \widetilde{U}^k \ldots \widetilde{U}^{m-1} R^m F^s \ldots F^1. \tag{19}$$

Let $\boldsymbol{\gamma} = R^m F^s \ldots F^1 \widehat{Q}(k)$, let

$$\widetilde{L}^m = \left[\begin{array}{c|c|c} I & & \\ \hline & 1/\gamma_k & \\ \hline & -\gamma_{k+1}/\gamma_k & \\ & \vdots & I \\ & -\gamma_m/\gamma_k & \end{array}\right],$$

and let

$$\widetilde{U}^m = \left[\begin{array}{c|c|c} & -\gamma_1 & \\ I & \vdots & \\ & -\gamma_{k-1} & \\ \hline & 1 & \\ \hline & & I \end{array}\right].$$

Then $\widetilde{U}^m \widetilde{L}^m \boldsymbol{\gamma} = \mathbf{e}^k$ and we will show that $\widehat{Q}^{-1} = U^1 \ldots U^{k-1} \widetilde{U}^k \ldots \widetilde{U}^m \widetilde{L}^m R^m F^s \ldots F^1$.

We now present the algorithm which updates the LU representation of Q^{-1} when the leaving column is nonkey. Assume that $\widehat{Q}(k)$ is replacing $Q(k)$ in the working basis.

ALG 1: LU UPDATE FOR NONKEY LEAVING COLUMN

1. Set $\boldsymbol{\beta} \leftarrow F^s \ldots F^1\widehat{Q}(k)$.
2. If $k \neq m$, set $\ell \leftarrow k$, $R^\ell \leftarrow I$, go to 4.
3. Set $\tilde{L}^m \leftarrow I$, where I is m by m.
   Set $\tilde{L}^m_{mm} \leftarrow 1/\beta_m$.
   Set $\tilde{U}^m \leftarrow I$, where I is m by m.
   Set $\tilde{U}^m_{jm} \leftarrow -\beta_j$, for $1 \leq j < m$.
   Stop with $\widehat{Q}^{-1} = U^1 \ldots U^{m-1}\tilde{U}^m\tilde{L}^m F^s \ldots F^1$.
4. Set $\alpha \leftarrow R^\ell[k]U^{\ell+1}(\ell+1)$.
   Set $R^{\ell+1} \leftarrow R^\ell$.
   Set $R^{\ell+1}_{k,\ell+1} \leftarrow \alpha$.
   Set $\tilde{U}^\ell \leftarrow U^{\ell+1}$.
   Set $\tilde{U}^\ell_{k,\ell+1} \leftarrow 0$.
   $(R^\ell U^{\ell+1} = \tilde{U}^\ell R^{\ell+1})$
   Set $\ell \leftarrow \ell+1$.
5. If $\ell < m$, go to 4.
   $(U^{k+1} \ldots U^m = \tilde{U}^k \ldots \tilde{U}^{m-1}R^m.)$
   Set $\boldsymbol{\beta} \leftarrow R^m\boldsymbol{\beta}$.
6. Set $\tilde{L}^m \leftarrow I$, where I is m by m.
   Set $\tilde{L}^m_{kk} \leftarrow 1/\beta_k$.
   Set $\tilde{L}^m_{jk} \leftarrow -\beta_j/\beta_k$, for $k < j \leq m$.
   Set $\tilde{U}^m \leftarrow I$, where I is m by m.
   Set $\tilde{U}^m_{jk} \leftarrow -\beta_j$, for $1 \leq j \leq k$.
   Set $\tilde{U}^m_{kk} \leftarrow 1$.
   Stop with $\widehat{Q}^{-1} = U^1 \ldots U^{k-1}\tilde{U}^k\tilde{U}^{k+1} \ldots \tilde{U}^m\tilde{L}^m R^m F^s \ldots F^1$.

We now present the justification for step 3 of ALG 1. For $k = m$, we claim that $\widehat{Q}^{-1} = U^1 \ldots U^{m-1}\tilde{U}^m\tilde{L}^m F^s \ldots F^1$. Note that $\widehat{Q}^{-1}\widehat{Q}(m) = U^1 \ldots U^{m-1}\tilde{U}^m\tilde{L}^m\boldsymbol{\beta}$. But by construction $\tilde{U}^m\tilde{L}^m\boldsymbol{\beta} = \mathbf{e}^m$. Consider

Proposition 1.

Let $\boldsymbol{\beta}$ be any m-vector and E^i be any column eta. If $\beta_i = 0$, then $E^i\boldsymbol{\beta} = \boldsymbol{\beta}$.

By Proposition 1, $U^1 \ldots U^{m-1}\mathbf{e}^m = \mathbf{e}^m$. Therefore, $\widehat{Q}^{-1}\widehat{Q}(m) = \mathbf{e}^m$. For $1 \leq \ell < m$, let $\boldsymbol{\gamma} = F^s \ldots F^1Q(\ell)$. By construction $\gamma_j = 0$ for $\ell < j \leq m$ and $\gamma_\ell = 1$. By Proposition 1, $U^{\ell+1} \ldots U^{m-1}\tilde{U}^m\tilde{L}^m\boldsymbol{\gamma} = \boldsymbol{\gamma}$. By the construction of $U^1 \ldots U^\ell$, we have $U^1 \ldots U^\ell\boldsymbol{\gamma} = \mathbf{e}^\ell$. Therefore, if the leaving column is Q(m), then step 3 of ALG 1 produces $\widehat{Q}^{-1}$.

We now present a theoretical justification for step 4 of ALG 1.

Proposition 2.

Let

$$U^{p+1} = \left[\begin{array}{c:c:c} I & & \\ & \underline{\boldsymbol{\eta}} & \\ & & I \end{array}\right] \quad \text{and } R^{p} = \left[\begin{array}{ccc} I & & \\ \hdashline & \underline{\boldsymbol{\gamma}} & \\ \hdashline & & I \end{array}\right] \leftarrow \text{row } \ell^*,$$

$$\begin{array}{c} \uparrow \\ \text{column } \ell \end{array}$$

where $\ell \neq \ell^*$.
If

$$\widetilde{U}^{p} = \left[\begin{array}{c:c:c} I & & \\ & \underline{\boldsymbol{\alpha}} & \\ & & I \end{array}\right] \quad \text{and } R^{p+1} = \left[\begin{array}{ccc} I & & \\ \hdashline & \underline{\boldsymbol{\beta}} & \\ \hdashline & & I \end{array}\right] \leftarrow \text{row } \ell^*$$

$$\begin{array}{c} \uparrow \\ \text{column } \ell \end{array}$$

where

$$\alpha_i = \begin{cases} 0, \text{ if } i = \ell^* \\ n_i, \text{ otherwise, and} \end{cases}$$

$$\beta_i = \begin{cases} \mathbf{n}\boldsymbol{\gamma}, \text{ if } i = \ell \\ \gamma_i, \text{ otherwise,} \end{cases}$$

then $R^pU^{p+1} = \widetilde{U}^pR^{p+1}$.

Proposition 2 is a theoretical justification for step 4 of ALG 1. The proposition to follow shows the precise structure of $R^mF^s \ldots F^1Q$. Consider

Proposition 3.

Let $U^* = F^s \ldots F^1Q$. If $\widetilde{U}^* = R^mU^*$, then

$$\widetilde{U}^*[i] = \begin{cases} U^*[i], \; i \neq k \\ \mathbf{e}^k, \text{ otherwise.} \end{cases}$$

We now present the results to prove that $\widehat{Q}^{-1} = U^1 \ldots U^{k-1}\widetilde{U}^k \ldots \widetilde{U}^m\widetilde{L}^mR^mF^s \ldots F^1$.

Proposition 4.

$$U^1 \ldots U^{k-1}\widetilde{U}^k \ldots \widetilde{U}^m\widetilde{L}^mR^mF^s \ldots F^1\widehat{Q}(k) = \mathbf{e}^k.$$

Proposition 5.

$$U^1 \ldots U^{k-1}\widetilde{U}^k \ldots \widetilde{U}^m\widetilde{L}^mR^mF^s \ldots F^1\widehat{Q}(i) = \mathbf{e}^i \text{ for } i \neq k.$$

By Propositions 4 and 5, we have

Corollary 6.

$$\hat{Q}^{-1} = U^1 \ldots U^{k-1}\tilde{U}^k \ldots \tilde{U}^m\tilde{L}^m R^m F^s \ldots F^1.$$

Hence, ALG 1 produces the updated working basis inverse.

5.2 Key Column Leaves The Basis

In this section, we present an algorithm for updating the working basis inverse to accomplish a switch between a key column and a nonkey column. That is, $\hat{Q} = RQ^{-1}$where R is given by (13) and

$$Q^{-1} = U^1 \ldots U^m F^s \ldots F^1. \tag{20}$$

We wish to obtain $\hat{Q}^{-1}$ in the same form as (20).

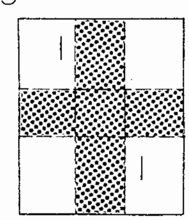
To accomplish this update, we begin with $\hat{Q}^{-1} = RU^1 \ldots U^m F^s \ldots F^1$. We apply Proposition 2 to RU^1 creating the factorization $\hat{Q}^{-1} = \tilde{U}^1 R^2 U^2 \ldots U^m F^s \ldots F^1$. We continue with the application of Proposition 2 until we obtain $\hat{Q}^{-1} = \tilde{U}^1 \ldots \tilde{U}^{k-1} R^k U^k \ldots U^m F^s \ldots F^1$. Proposition 2 does not apply to $R^k U^k$. However, a simple update would be to let $\tilde{U}^k = \ldots = \tilde{U}^m = I$ and use the below factorization:

$$\hat{Q}^{-1} = \underbrace{\tilde{U}^1 \ldots \tilde{U}^m}_{\text{LEFT FILE}} R^k \underbrace{U^k \ldots U^m F^s \ldots F^1}_{\text{RIGHT FILE}}.$$

This update simply involves application of Proposition 2 until it does not apply ($\ell = \ell^*$) and then shifting the remainder of the left file to the right file. We call this update the *TYPE 1 UPDATE*.

We will now give an update in which $R^k U^k \ldots U^m$ is modified as opposed to moving them to the right file. Let

$$E^k = R^k U^k =$$



← row k

Then we define matrices $\tilde{U}^{k+1}$ and E^{k+1}such that $E^k U^{k+1} = \tilde{U}^{k+1} E^{k+1}$. Following this procedure, $R^k U^k \ldots U^m$ can be replaced by $\tilde{U}^{k+1} \ldots \tilde{U}^m E^{m+1}$ so that

$$\hat{Q}^{-1} = \tilde{U}^1 \ldots \tilde{U}^{k-1}\tilde{U}^{k+1} \ldots \tilde{U}^m E^{m+1} F^s \ldots F^1.$$

Further, we define a row eta $\tilde{R}$ and a column eta $\tilde{F}$ such that $E^{m+1} = \tilde{R}\tilde{F}$. Therefore,

$$\hat{Q}^{-1} = \underbrace{\tilde{U}^1 \ldots \tilde{U}^{k-1}\tilde{U}^{k+1} \ldots \tilde{U}^m}_{\text{LEFT FILE}} \underbrace{\tilde{R}\tilde{F}F^s \ldots F^1}_{\text{RIGHT FILE}}.$$

We call this update the *TYPE 2 UPDATE*.

We now present a set of propositions which justify the *TYPE 2 UPDATE*.

Proposition 7.

Let

$$U^{p+1} = \left[\begin{array}{c|c|c} I & \begin{matrix}\eta_1\\ \vdots \\ \eta_{\ell-1}\end{matrix} & 0 \\ \hline & \eta_\ell & \\ \hline 0 & \begin{matrix}\eta_{\ell+1}\\ \vdots \\ \eta_n\end{matrix} & I \end{array}\right] \quad \text{and } E^p = \left[\begin{array}{c|c|c} I & \begin{matrix}\mu_1\\ \vdots \\ \mu_{\ell^*-1}\end{matrix} & 0 \\ \hline \gamma_1 \;\ldots\; \gamma_{\ell^*-1} & \gamma_{\ell^*} & \gamma_{\ell^*+1} \;\ldots\; \gamma_n \\ \hline 0 & \begin{matrix}\mu_{\ell^*+1}\\ \vdots \\ \mu_n\end{matrix} & I \end{array}\right]$$

where $\ell \neq \ell^*$ and $\mu_\ell = 0$.
If

$$\tilde{U}^{p+1} = \left[\begin{array}{c|c|c} I & \begin{matrix}\alpha_1\\ \vdots \\ \alpha_{\ell-1}\end{matrix} & 0 \\ \hline & \alpha_\ell & \\ \hline 0 & \begin{matrix}\alpha_{\ell+1}\\ \vdots \\ \alpha_n\end{matrix} & I \end{array}\right] \quad \text{and } E^{p+1} = \left[\begin{array}{c|c|c} I & \begin{matrix}\mu_1\\ \vdots \\ \mu_{\ell^*-1}\end{matrix} & \\ \hline \lambda_1 \;\ldots\; \lambda_{\ell^*-1} & \lambda'_{\ell^*} & \lambda_{\ell^*+1} \;\ldots\; \lambda_n \\ \hline 0 & \begin{matrix}\mu_{\ell^*+1}\\ \vdots \\ \mu_n\end{matrix} & I \end{array}\right]$$

where

$$\lambda_i = \begin{cases} \boldsymbol{\gamma\eta}, & \text{if } i = \ell, \\ \gamma_i, & \text{otherwise,} \end{cases}$$

$$\alpha_i = \begin{cases} 0, & \text{if } i = \ell^*, \\ \eta_i + \mu_i \eta_{\ell^*}, & \text{otherwise,} \end{cases}$$

then $E^p U^{p+1} = \tilde{U}^{p+1} E^{p+1}$.

The following proposition is used to replace the cross matrix E^{m+1} with a row eta $\tilde{R}$ and a column eta $\tilde{F}$.

Proposition 8.
Let

$$E = \left[\begin{array}{c:c:c} I & \begin{matrix}\mu_1\\ \vdots \\ \mu_{\ell-1}\end{matrix} & 0 \\ \hdashline \gamma_1 \;\cdots\; \gamma_{\ell-1} & \gamma_\ell & \gamma_{\ell+1} \;\cdots\; \gamma_n \\ \hdashline 0 & \begin{matrix}\mu_{\ell+1}\\ \vdots \\ \mu_n\end{matrix} & I \end{array}\right]$$

If

$$\tilde{R} = \left[\begin{array}{c:c:c} I & \multicolumn{2}{c}{0} \\ \hdashline \gamma_1 \;\cdots\; \gamma_{\ell-1} & X & \gamma_{\ell+1} \;\cdots\; \gamma_n \\ \hdashline \multicolumn{2}{c:}{0} & I \end{array}\right] \quad \text{and } \tilde{F} = \left[\begin{array}{c:c:c} I & \begin{matrix}\mu_1\\ \vdots \\ \mu_{\ell-1}\end{matrix} & 0 \\ \hdashline & Y & \\ \hdashline 0 & \begin{matrix}\mu_{\ell+1}\\ \vdots \\ \mu_n\end{matrix} & I \end{array}\right]$$

where X and Y are such that

$$XY = \gamma_\ell - \sum_{\substack{i=1\\ i\neq \ell}}^{n} \gamma_i \mu_i,$$

then $E = \tilde{R}\tilde{F}$.

We now present the update algorithm for the case in which the ℓ^{th} column of T is being switched with the k^{th} column of C. Let $\boldsymbol{\gamma} = \mathbf{e}^{\ell} T^{-1} C$.

ALG 2: LU UPDATE FOR A KEY LEAVING COLUMN

1. Set $R^1 \leftarrow I$.
   Set $R^1[k] \leftarrow \boldsymbol{\gamma}$.
   Set $i \leftarrow 1$.
2. If $i = k$, go to 4.
   Set $\alpha \leftarrow R^i[k]U^i(i)$.
   Set $R^{i+1} \leftarrow R^i$.
   Set $R^{i+1}_{ki} \leftarrow \alpha$.
   Set $\tilde{U}^i \leftarrow U^i$.
   Set $\tilde{U}^i_{ki} \leftarrow 0$.
3. Set $i \leftarrow i + 1$ and go to 2.
4. Set $\tilde{U}^k \leftarrow I$.

5. Apply Proposition 7 to E^iU^{i+1} to form $\tilde{U}^{i+1}E^{i+1}$.
   Set $i \leftarrow i + 1$.
6. If $i < m$, go to 5.
7. Apply Proposition 8 to E^m to obtain $\tilde{R}\tilde{F}$ where $X = 1$.
   At the completion of step 7 we have $\hat{Q}^{-1} = \tilde{U}^1 \ldots \tilde{U}^m\tilde{R}\tilde{F}F^s \ldots F^1$.

VI. COMPUTATIONAL EXPERIMENTATION

Three test problems were selected for the experiment. Sc205 is a staircase linear program which was generated by Ho and Loute [12] and transformed into a network with side constraints. Gifford-Pinchot is a model of the Gifford-Pinchot National Forest [10] which has also been transformed into a network with side constraints. RAN is a randomly generated problem.

These problems were first solved and the pivot agenda was saved. That is, entering and leaving columns for each pivot were saved on a file. This file was then used by each code so that all three basis updates follow the same path to the optimum. The number of nonzeroes required to represent Q^{-1} at various points in the solution process is illustrated in Figures 1 and 2. For both problems, the LU Type 2 update dominated both the LU Type 1 update and the product-form code in terms of nonzeroes in the inverse. The average core storage required for Q^{-1} using the product-form update is approximately double that required for the best LU update.

Given the above results, we developed three specialized network with side constraints codes and computationally compared them with three general in-core LP systems and a special system for multicommodity network flow problems. All codes are written in FORTRAN and have not been tailored to either our equipment or our FORTRAN compiler. None of the codes were tuned for our problem set. A brief description of each code follows.

NETSIDE1, NETSIDE2 AND NETSIDE3 are our specialized network with side constraints systems. The first maintains Q^{-1} in product form, while the second and third maintain Q^{-1} in LU form using a Type 1 and Type 2 update, respectively. All use the Hellerman and Rarick [11] reinversion routine. The working basis is reinverted every 60 iterations. The pricing routine uses a candidate list of size 6 with block size of 200.

MINOS [15] stands for "a Modular In-Core Nonlinear Optimization System" and is designed to solve problems of the following form:

$$\begin{aligned} \text{minimize} \quad & f(\mathbf{x}) + \mathbf{cx} \\ \text{subject to:} \quad & A\mathbf{x} = \mathbf{b} \\ & \ell \leq \mathbf{x} \leq \mathbf{u} \end{aligned}$$

where $f(\mathbf{x})$ is continuously differentiable in the feasible region. For this

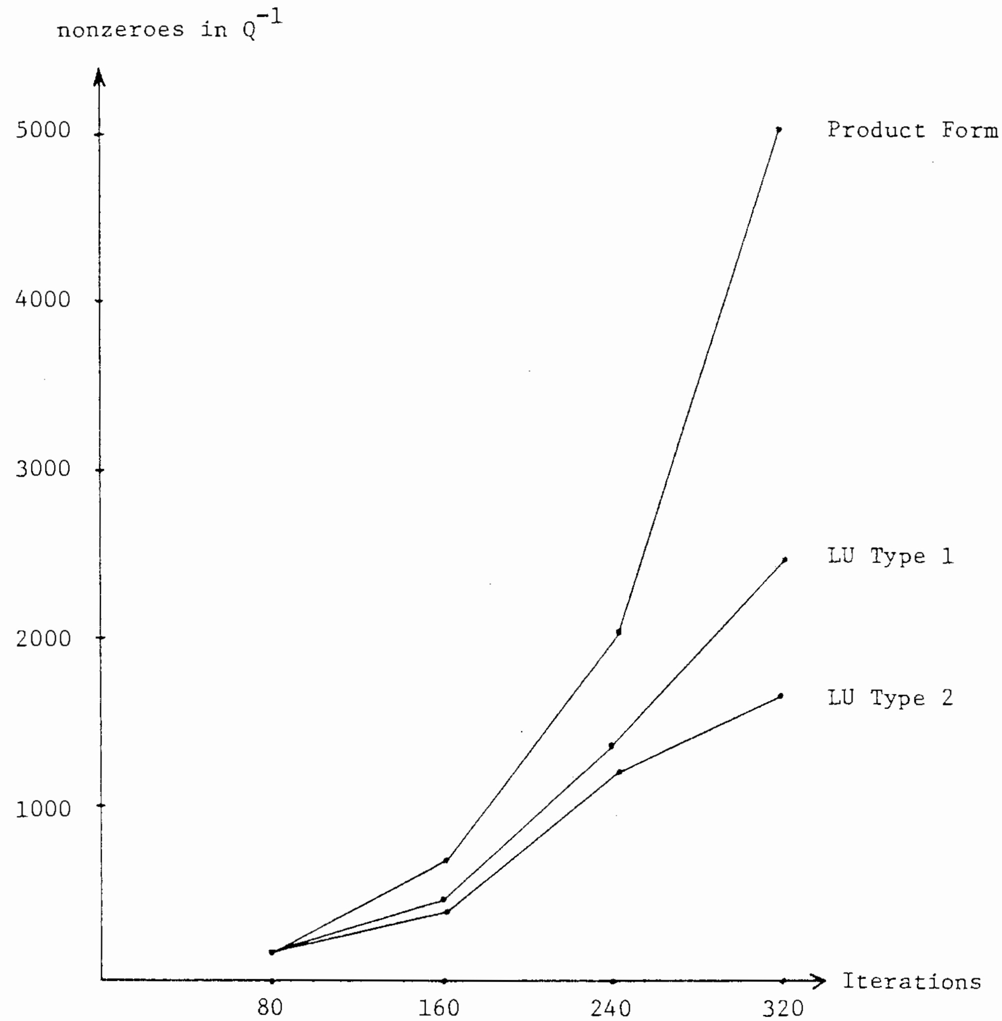


Figure 1. Nonzero Buildup In The Working Basis Inverse On SC205 [22].
(317 columns, 119 nodes, 87 side constraints)

study f(x) = 0 at all **x** and therefore none of the nonlinear subroutines were used for problem solution.

For linear programs, MINOS uses the revised simplex algorithm with all data and instructions residing in core storage. The basis inverse is maintained as an LU factorization using a Bartels-Golub update. The reinversion routine uses the Hellerman-Rarick [11] pivot agenda algorithm.

XMP is a library of FORTRAN subroutines which can be used to solve linear programs. The basis inverse is maintained in LU factored form. The pricing routine uses a candidate list of size 6 with two hundred columns being scanned each time the list is refreshed. The basis is reinverted every 50 iterations.

LISS stands for "Linear In-Core Simplex System" and is an in-core LP solver with the basis inverse maintained in product form. The reinversion routine is a modification of the work of Hellerman and Rarick [11]. The basis inverse is refactored every 50 iterations. A partial pricing scheme is used with 20 blocks.

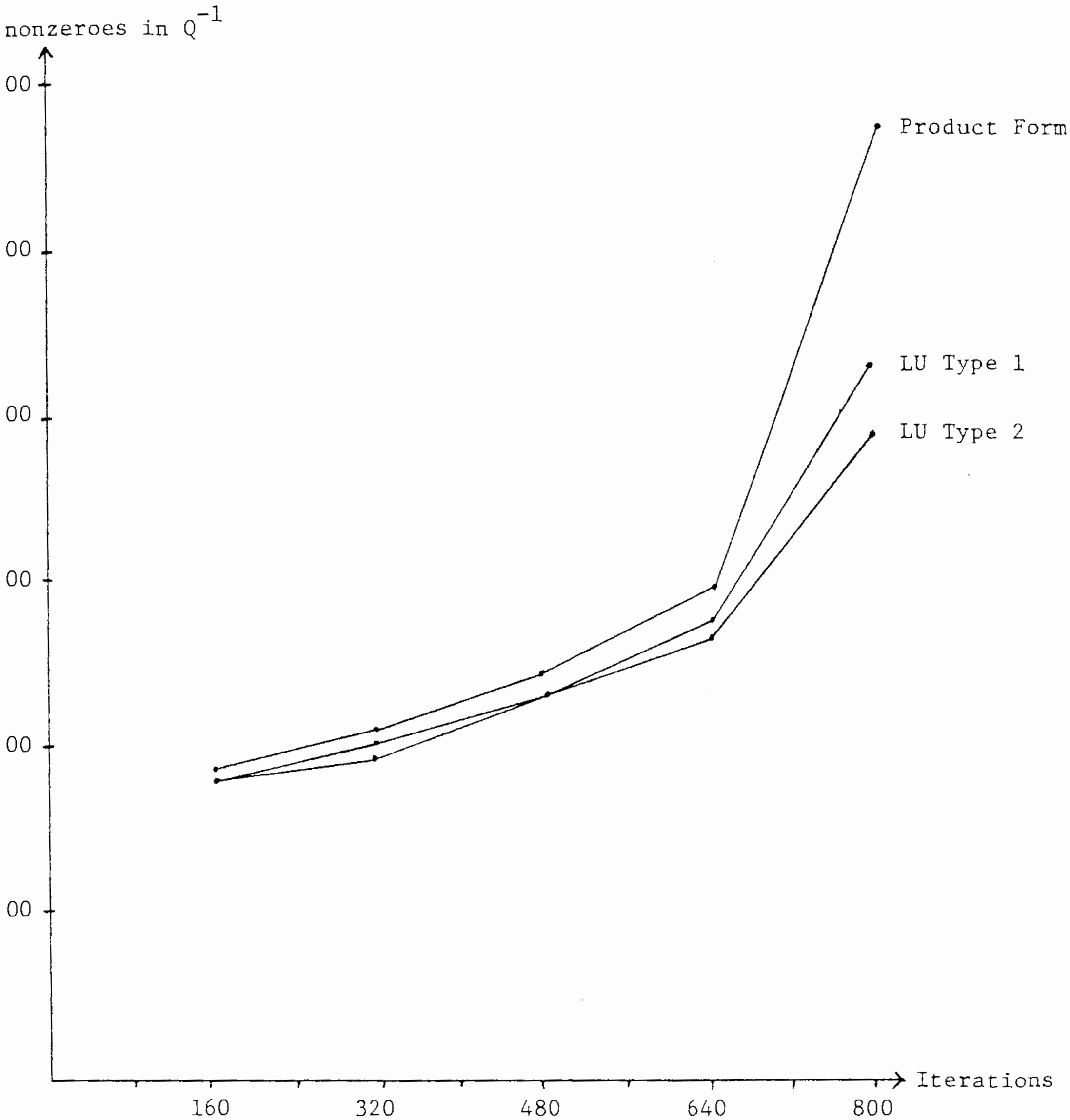


Figure 2. Nonzero Buildup In The Working Basis Inverse On Gifford Pinchot [20]. (1160 columns, 533 nodes, 84 side constraints)

MCNF stands for "Multicommodity Network Flow". MCNF uses the primal partitioning algorithm also. The basis inverse is maintained as a set of rooted spanning trees (one for each commodity) and a working basis inverse in product form. This working basis inverse has dimension equal to the number of binding GUB constraints. A partial pricing scheme is used. Our computational experience is given in Table 1.

The row entitled GUB Constraints, gives the number of LP rows which correspond to "GUB Constraints". The row, entitled "Binding GUB Constraints", gives the number of GUB constraints met as equalities at optimality using MCNF. All runs were made on the CDC 6600 at Southern Methodist University using the FTN compiler with the optimization feature enabled.

Table 1 **Comparison of Codes for Solving Multicommodity Network Flow Problems**

(All Times Exclude Input and Output)

PROB DESC.																					
Number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	Totals
LP Rows	500	400	401	501	499	415	576	561	547	496	559	477	513	490	532	552	544	571	544	577	
LP Cols	995	720	841	896	851	732	850	972	901	900	934	842	843	840	840	848	900	841	872	841	
% Network Rows	100	100	99.4	99.4	80	96	56	86	25	32	89	63	47	56	45	62	63	66	54	73	
GUB Const.	0	0	1	1	99	15	256	81	412	336	59	177	273	215	292	212	204	196	254	157	
Binding GUB Const.	0	0	0	0	2	3	5	3	6	6	7	9	8	11	13	17	21	23	26	31	
Number Nonzeroes	1910	1440	1701	1794	2553	1746	2550	2916	2703	2700	2104	2526	2529	2520	2394	2414	2700	2523	2616	2523	
Number Commodities	5	20	20	5	10	20	4	12	3	4	5	6	4	5	3	4	4	5	3	6	
MINOS [31]																					
Scaled Time	984	810	839	862	563	688	386	633	484	517	451	361	329	231	356	281	198	130	172	154	
Time (sec.)	99.40	43.07	44.29	82.61	56.70	41.86	61.76	83.66	29.09	31.70	79.56	44.63	42.55	44.19	45.28	55.86	62.92	63.35	50.72	67.39	1131
Pivots	1207	692	684	1068	739	655	750	978	376	427	986	619	564	592	594	712	781	793	692	831	
XMP [28]																					
Scaled Time	632	582	559	620	426	528	356	498	452	507	435	376	269	272	381	329	207	189	222	275	
Time (sec.)	63.81	30.98	29.49	59.43	42.94	32.12	57.06	65.78	27.17	31.10	76.70	46.48	34.90	52.11	48.43	67.53	65.85	92.11	65.73	120.38	1110
Pivots	1198	751	720	1109	806	737	945	1135	499	619	1243	906	661	945	877	1099	1117	1375	1085	1687	
LISS [2]																					
Scaled Time	398	326	337	364	433	454	565	622	1125	1503	400	1223	675	627	709	588	360	380	334	493	
Time (sec.)	40.20	17.34	17.79	34.88	43.63	27.63	90.55	82.24	67.62	92.13	70.53	151.15	87.36	120.04	90.19	116.87	114.78	185.46	98.85	215.86	1765
Pivots	1267	766	769	1133	1319	1084	2087	2091	1799	2220	1767	2416	2048	2309	2137	2233	2400	2660	2120	2762	
MCNF [26]																					
Scaled Time	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	
Time (sec.)	10.10	5.32	5.28	9.58	10.08	6.08	16.02	13.22	6.01	6.13	17.63	12.36	12.95	19.15	12.72	19.89	31.84	48.84	29.54	43.81	337
Pivots	722	493	451	717	466	443	595	610	212	246	745	530	515	677	461	662	989	1272	885	1191	
NETSIDE1																					
Scaled Time	230	240	235	245	177	209	185	201	285	243	159	167	146	145	203	143	108	74	126	111	
Time (sec.)	23.18	12.79	12.41	23.43	17.87	12.68	29.64	26.51	17.11	14.87	28.06	20.63	18.96	27.85	25.87	28.50	34.25	36.31	37.33	48.65	720
Pivots	885	602	570	887	628	577	662	852	298	309	917	613	465	716	547	703	853	885	816	1180	
NETSIDE2																					
Scaled Time	229	239	230	242	182	209	191	200	296	253	161	167	152	148	209	148	111	76	134	110	
Time (sec.)	23.17	12.72	12.16	23.23	18.34	12.72	30.61	26.42	17.77	15.48	28.44	20.61	19.71	28.40	26.59	29.46	35.49	37.23	39.49	48.40	731
Pivots	885	602	570	887	628	577	662	852	298	309	917	613	465	716	547	703	853	885	823	1177	
NETSIDE3																					
Scaled Time	232	247	234	247	179	212	195	202	292	255	163	171	155	158	212	150	115	79	136	117	
Time (sec.)	23.41	13.16	12.38	23.69	18.00	12.86	31.19	26.73	17.56	15.65	28.78	21.17	20.03	30.25	17.01	29.74	36.74	38.37	40.22	51.38	816
Pivots	885	602	570	887	628	577	682	852	298	309	917	613	465	716	547	703	853	885	823	1180	

Based on these results, we conclude that for lightly constrained multicommodity network flow problems

(i) XMP and MINOS run at approximately the same speed,

(ii) NETSIDE1, NETSIDE2 and NETSIDE3 run at approximately the same speed, and

(iii) the three NETSIDE codes are approximately twice as fast as XMP and MINOS.

References

1. Ali, A., R. Helgason, and J. Kennington, "An Air Force Logistics Decision Support System Using Multicommodity Network Models", Technical Report 82-OR-1, Department of Operations Research, Southern Methodist University, Dallas, Texas 75275, (1982).
2. Barnett, D., J. Binkley, and B. McCarl, "The Effects of U. S. Port Capacity Constraints on National and World Grain Shipments", Technical Paper, Purdue Agricultural Experiment Station, Purdue University, West Lafayette, Indiana, (1982).
3. Bartels, R., and G. Golub, "The Simplex Method of Linear Programming Using LU Decomposition", **Communications of ACM,** 12, 266–268, (1969).
4. Bixby, R. E., "Recent Algorithms for Two Versions of Graph Realization and Remarks on Applications to Linear Programming", Technical Report.
5. Forrest, J. J. H., and J. A. Tomlin, "Updated Triangular Factors of the Basis to Maintain Sparcity in the Product Form Simplex Method", **Mathematical Programming,** 2, 3, 263–278, (1972).
6. Glover, F., and D. Klingman, "The Simplex Son Algorithm for LP/Embedded Network Problems", Technical Report CCS 317, Center for Cybernetic Studies, The University of Texas, Austin, Texas, (1977).
7. Glover, F., R. Glover, J. Lorenzo, and C. McMillan, "The Passenger-Mix Problem in the Scheduled Airlines", **Interfaces,** 12, 3, 73–80, (1982).
8. Glover, F., R. Glover, and F. Martinson, "The U. S. Bureau of Land Management's New Netform Vegetation Allocation System", Technical Report, Division of Information Science Research, University of Colorado, Boulder, Colorado, (1982).
9. Graves, G. W., and R. D. McBride, "The Factorization Approach to Large-Scale Linear Programming", **Mathematical Programming,** 10, 1, 91–110, (1976).

10. Helgason, R., J. Kennington, and P. Wong, "An Application of Network Programming for National Forest Planning", Technical Report OR 81006, Department of Operations Research, Southern Methodist University, Dallas, Texas, (1981).
11. Hellerman, E., and D. Rarick, "Reinversion With the Preassigned Pivot Procedure", **Mathematical Programming,** 1, 195–216, (1971).
12. Ho, J. K., and E. Loute, "A Set of Staircase Linear Programming Test Problems", **Mathematical Programming,** 20, 2, 245–250, (1981).
13. Kennington, J. L., and R. V. Helgason, **Algorithms for Network Programming,** John Wiley and Sons, New York, New York, (1980).
14. McLain, D. R., "A Multicommodity Approach to a Very Large Aero-medical Transportation Problem", (working paper) Operations Research Division, Military Airlift Command, Scott Air Force Base, Illinois, (1983).
15. Murtagh, B., and M. Saunders, "MINOS User's Guide", Technical Report 77-9, Systems Optimization Laboratory, Department of Operations Research, Stanford University, Stanford, California, (1977).
16. Wagner, D. K., "An Almost Linear-Time Graph Realization Algorithm", unpublished dissertation, Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, Illinois, (1983).

JEFFERY L. KENNINGTON

Jeffery L. Kennington is a professor and chairman of the Operations Research Department of Southern Methodist University. He received his Ph.D. from the Department of Industrial and Systems Engineering at Georgia Institute of Technology. He is a co-author of the John Wiley book entitled *Algorithms for Network Programming.* His other publications have appeared in *Mathematical Programming, Operations Research, Management Science, Naval Research Logistics Quarterly, Networks,* and *Institute of Industrial Engineering Transactions.*

RICHARD S. BARR

Richard S. Barr is Associate Professor of Operations Research at the School of Engineering and Applied Science, Southern Methodist University, Dallas, TX. He received his B.S. in Electrical Engineering, M.B.A. and Ph.D. in Operations Research, all from the University of Texas in Austin. His current research interest include ultra-large scale mathematical programming, with an emphasis on network optimization; microcomputer applications of operations research; microeconomic simulation models; and algorithms for new computer architectures. He is a contributing editor for *Interfaces,* and has published in *Operations Research, Mathematical Programming,* and *European Journal of Operational Research.*

KEYVAN FARHANGIAN

Keyvan Farhangian is a systems designer with Consilium Associates, Inc. of Palo Alto, CA. He received his B.A. in Business Administration from the University of Tehran, Iran, and his M.S. and Ph.D. in Operations Research from Southern Methodist University.