An Implicit Enumeration Method for LTL Network Design

BRUCE W. LAMAR AND YOSEF SHEFFI

The optimal design of less-than-truckload (LTL) motor carrier networks is cast as a fixed-charge network design problem. A recently developed lower bound is applied iteratively with a link inclusion heuristic in an implicit enumeration framework. Initial computational results for solving this class of problems are reported.

Less-than-truckload (LTL) motor carriers haul freight from many origins to many destinations. In this paper the emphasis is on the line-haul (i.e., intercity) operations of an LTL carrier rather than on the local or pickup-and-delivery operations. [Line-haul operations typically account for 40 percent of the operating expenses for an LTL motor carrier *(1)*.] The line-haul network design for LTL operations is frequently referred to as a load plan. In such a plan, shipments are considered to originate and terminate at the carrier's end-of-line (EOL) (i.e., city) terminals, and an origin-destination (OD) pair of EOL terminals is referred to as a market. Associated with each market is a shipment demand specifying the shipments that the carrier must haul from the origin to the destination. [Shipment demands are expressed in flow units such as hundredweight (CWT) per day or cubic feet per week.] Because the shipment demand for an individual market is typically much smaller than the capacity of a trailer (40,000 lb or 6,250 ft^3), it is economical to combine the demand for several markets at the origin EOL terminal and send this consolidated demand to an intermediate terminal, referred to as a breakbulk (BB) terminal. At the BB terminal, consolidated shipments on incoming trucks are unloaded, sorted, and reloaded onto outgoing trucks bound either for the final destination or another BB terminal.

In designing a line-haul network, a motor carrier must make the following two interrelated sets of decisions: (a) which links should be used and (b) how the shipment demand should be routed over the selected links. The motor carrier's criterion for making these decisions is to minimize its total transportation and handling costs while providing a high level of service. The handling costs, which occur at the BB and EOL terminals, are assumed to be proportional to the volume of freight sent through these terminals. Thus, each terminal in the carrier's line-haul network can be represented by a pair of nodes connected by a single arc, and the costs of these special arcs are simply the handling costs of the respective BB or EOL terminals.

B. W. Lamar, Graduate School of Management, University of California, Irvine, Calif. 92717. Y. Sheffi, Department of Civil Engineering, Massachusetts Institute of Technology, Cambridge, Mass. 02139.

The representation of the transportation costs for an LTL motor carrier is somewhat more involved. The link cost function for a typical LTL line-haul link is shown in Figure 1. The daily link cost depends on the average daily volume of freight (i.e., flow) carried on that link. If the flow is zero, the link is not used and so naturally the cost is zero. If the flow is small, the carrier will send a prespecified number of trucks per day over that link even though these trucks will be only partly full. This is done in order to maintain a satisfactory level of service *(2)*. In Figure 1, this minimum frequency is set at two truckloads per day. If the link flow is below this minimum frequency, the carrier's transportation cost (fuel, maintenance, etc.) for this link is essentially set at a fixed value. On the other hand, if the link flow is above the minimum frequency, the carrier will implement a "go when full" truck-dispatching policy and the link cost will be proportional to the number of truckloads of freight carried over that link.

The highly nonlinear nature of the transportation costs experienced by an LTL motor carrier makes network design problems difficult to evaluate analytically. For this reason, the transportation cost function shown in Figure 1 is approximated by the fixed-charge cost function shown in Figure 2. This approximation captures both the cost for the minimum service requirement inherent in low flow levels and the cost proportional to freight volume inherent in high flow levels. As with any approximation, however, this simplified cost structure strikes a balance between a realistic representation of the problem and a mathematical form that is analytically tractable.

The mathematical model described in this paper provides two tools to aid an LTL motor carrier in designing and operating a line-haul network. First, the model generates a series of heuristic solutions that the carrier can implement to route and consolidate shipment demand. Second, the model provides a lower bound to the minimum total transportation and handling costs for the carrier's network. This lower bound is used to determine whether a heuristic solution is "close enough" to optimal. (It can also be used to evaluate the near-optimality of heuristics generated by other procedures.) The network design model can be solved whenever the carrier is updating the line-haul network. Depending on the volatility of the shipment demand in the OD markets, this updating could vary from a weekly to a quarterly basis. Moreover, the model can answer strategic "what if" questions concerning cities that the carrier is not currently serving and locations of new BB terminals. For each terminal configuration the model can suggest efficient strategies for including these markets in the carrier's network.

The paper is organized as follows: (a) the model formulation and its relaxation, (b) an implicit enumeration solution procedure, and (c) some computational results.

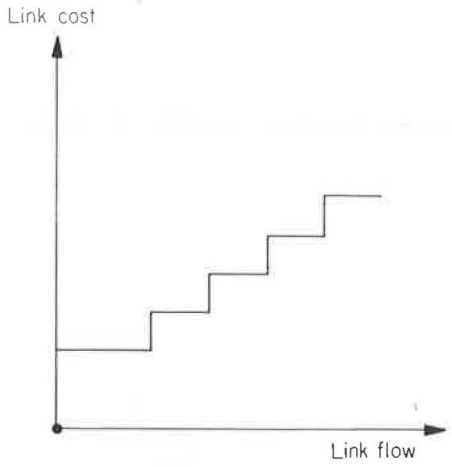

FIGURE 1 Typical link cost function.

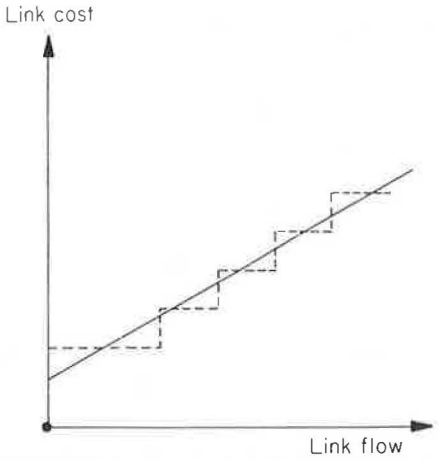

FIGURE 2 Approximation of LTL link cost function.

PROBLEM FORMULATION

This section is in two parts. First the fixed-charge network design problem addressed here as an integer program (IP) is formulated, and second, an LP relaxation of this formulation is discussed.

IP Formulation

The following notation is used to define the problem:

N = set of nodes with generic element n,
A = set of directed arcs with generic element a, and
M = set of OD pairs ("markets") with generic element m.

The shipment demand is characterized by a market flow vector, $\mathbf{q} = (. . ., q_m, . . .)$. The units of this demand are, for example, hundredweight per week or cubic feet per day.

All arcs in this problem are design elements. Each arc (a) is associated with a fixed charge (f_a) and a variable cost (c_a). In the application discussed here, f_a stems from the minimum service requirement, whereas c_a represents the (average) marginal cost of hauling freight over arc a (see Figure 3).

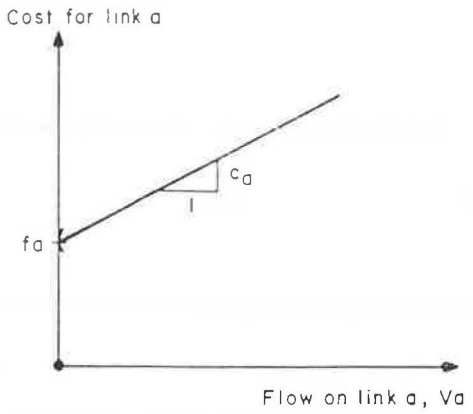

FIGURE 3 Fixed-charge link cost function.

To characterize the decision variables here, let $y_a = 1$ if link a is in the network, and $y_a = 0$ otherwise. Also let $x_{a,m} = 1$ if link a carries flow q_m (i.e., if it is used to carry flow between the origin and the destination associated with the mth OD pair), and $x_{a,m} = 0$ otherwise. Thus $\mathbf{y} = (. . ., y_a, . . .)$ is the network design vector, and $\mathbf{x} = (. . ., x_{a,m}, . . .)$ is the flow-routing vector.

The vector of link flows $\mathbf{v} = (. . ., v_a, . . .)$ can be expressed as

$$v_a = \sum_m q_m x_{a,m} \qquad \forall\, a \tag{1}$$

(When no other indication is given, the notations Σ and $\forall$ refer to all members of the relevant set. Thus in Equation 1, Σ_m is equivalent to $\Sigma_{m \in M}$, and $\forall\ a$ is equivalent to $\forall\ a \in A$.) A parameter used later in the program formulation is the capacity of each arc, denoted by the vector $\mathbf{u} = (. . ., u_a, . . .)$. Because the problem discussed here is not naturally capacitated, u_a can be defined as the largest possible value that v_a can attain. Depending on the network structure, this maximum may be equal to or less than the combined shipping demand for the entire network (i.e., $\Sigma_m q_m$).

To formulate the flow conservation constraints, let $I(a)$ and $J(a)$ denote the "tail" node and "head" node, respectively, of arc a. Similarly, let $O(m)$ and $D(m)$ define the origin and destination nodes, respectively, for market m. In addition, let A_n be the set of arcs whose tail node is n and let B_n be the set of arcs whose head node is n.

The multicommodity network flow balance constraints can now be written as

$$\sum_{a \in A_n} x_{a,m} - \sum_{a \in B_n} x_{a,m} = \left\{ \begin{array}{ll} 1 & \text{if } n = O(m) \\ -1 & \text{if } n = D(m) \\ 0 & \text{for all other } n \in N \end{array} \right\} \quad \forall\ n,m \tag{2a}$$

$$= x_{a,m} \geq 0 \qquad \forall\ a,m \tag{2b}$$

To avoid repeating this set of constraints throughout the paper, let X define the set of routing vectors $\mathbf{x} = (. . ., x_{a,m}, . . .)$ that comply with Equations 2.

The fixed-charge network design problem (denoted IP) can now be stated as the following integer program:

Program IP:

$$\min_{x \in X} \sum_m \sum_a c_a q_m x_{a,m} + \sum_a f_a y_a \tag{3a}$$

subject to

$$\sum_m q_m x_{a,m} \leq u_a y_a \qquad \forall\, a \tag{3b}$$

$$y_a \in \{0, 1\} \qquad \forall\, a \tag{3c}$$

Let (**x***, **y***) denote the optimal solution vectors, let **v*** denote the optimal link flow vector, and let z^*[IP] denote the optimal objective function value for program IP. The objective function 3a in this program minimizes the total system costs, including both variable and fixed charges, over all the links in the network. Carrying the minimization over X guarantees that the feasible solutions are restricted to directed paths over the network. Constraints 3b guarantee that a link carries flow only if it is in the network, and constraints 3c ensure the integrality of the decision variables. (Even without a specific constraint, it will always be the case that $x_{a,m} \in \{0, 1\}\ \forall\ a,m$ because the problem is uncapacitated.) Note that alternatively constraints 3b can be written in a disaggregate form:

$$x_{a,m} \leq y_a \qquad \forall\ a,m \tag{4}$$

Although this form generates a tighter LP relaxation than the LP relaxation of IP, the advantage of using constraints 3b is that the LP relaxation is then particularly easy to solve. This ease is exploited in the procedure that calculates the lower bound for z^*[IP], as shown by Lamar et al. *(3)*. For additional discussion of aggregate and disaggregate formulations, see work by Wong *(4)*, Rardin and Choe *(5)*, and Balakrishnan *(6)*. A general discussion of fixed-charge network design problems has been given by Wong *(4)* and Magnanti and Wong *(7)*.

LP Relaxation

The linear programming relaxation of IP is formed by replacing constraints 3c with the nonnegativity constraints

$$y_a \geq 0 \qquad \forall\, a \tag{5}$$

Let LP denote this relaxation of IP. Let **x***, **v***, and **z***[LP] denote the optimal values of the routing decision variables, the arc flows, and the objective function, respectively, in program LP. Note that constraints 5 can be omitted from LP because the nonnegativity of $\{y_a\}$ is ensured by constraints 3b and the nonnegativity of **x** (which is required in the set X).

The LP relaxation is then given by

Program LP:

$$\min_{x \in X} \sum_m \sum_a q_m (c_a + f_a/u_a) x_{a,m} \tag{6}$$

The equivalence between program LP as formulated in Equation 6 and its original formulation (Equations 2, 3a, 3b, and 5) can be shown on the basis of Balinski's observation *(8)* that constraints 3b will always be satisfied with equality in the optimal solution of LP. Because $u_a > 0\ \forall\ a$, constraints 3b can be solved explicitly for y_a and substituted in the objective function 3a to obtain the formulation of LP given in Equation 6. As mentioned in connection with Equation 4, program LP is easy to solve. The reason for this is that it decomposes by origin (and by OD pair) into a set of independent shortest-path problems (with flow assignment). Thus this program can be solved by a many-to-many algorithm [see, e.g., the paper by Floyd *(9)*] or by repeated application of a one-to-many algorithm [see, e.g., the discussion by Moore *(10)*]. These algorithms, coupled with recent list-processing techniques [see, e.g., the paper by Glover et al. *(11)*], are very efficient. Consequently, LP can be solved repeatedly in the course of finding both a lower bound and an heuristic solution of the integer program, IP.

The solution procedure for IP is described next.

SOLUTION PROCEDURE

The implicit enumeration (IE) procedure presented in this section determines an ε-optimal solution to the integer program, IP. That is, the algorithm determines a heuristic (i.e., feasible) solution to IP whose objective function value is always within $100 \cdot \varepsilon$ percent of z^*[IP] (or determines that IP is infeasible). The value of ε is assumed to be specified a priori.

A distinctive difference between the standard implicit enumeration method [see, e.g., work by Geoffrion and Marsten *(12)*] and the one presented in this section is that here the lower bound for each subproblem is not obtained directly from the LP relaxation of the integer problem. Instead, the lower bound is based on a "capacity improvement" procedure (described in the following discussion). This results in tighter lower bounds and a reduction in the computational effort required for the IE algorithm.

In the following procedure the basic "divide and conquer" strategy is used, which can be represented as a binary enumeration tree. Each point in the tree represents a subproblem of IP in which the binary decision variables, $\{y_a\}$, are partitioned into "out" and "in" subsets. The IE procedure determines a lower bound (to the optimal objective function value) and a heuristic solution for each feasible subproblem that it evaluates. The best (i.e., lowest-cost) heuristic solution (from among the subproblems that have been evaluated) is retained as the incumbent solution to IP. The process continues until the lower bound of all subproblems is within $100 \cdot \varepsilon$ percent of the objective function value of the incumbent.

The point in the binary enumeration tree being evaluated at any given time is referred to as the "current" point. This current point can be specified by a constraint set Y whose elements are of the form $\{y_a = 0\}$ or $\{y_a = 1\}$. The current integer subproblem, denoted IP_Y, is formed by adding the constraints in Y to IP; and the current linear subproblem, denoted LP_Y, is formed by adding the constraints in Y to problem LP. (Observe that LP_Y can be expressed as a shortest-path problem simply by modifying the objective function coefficients in LP rather than by adding constraints to this problem.) In addition, the IE procedure described here evaluates a

family of linear subproblems formed by parametrically altering an arc capacity vector, $\mathbf{u}_Y$. Let $LP_Y(\mathbf{u}_Y)$ denote this family of current linear subproblems.

Figure 4 is a flowchart of the basic steps in the IE algorithm. Step 0 initializes the algorithm by placing program IP in the candidate list. This list identifies the subproblems that are to be evaluated in the IE procedure. Along with the specification of IP, the capacity parameter vector $\mathbf{u}$ is also stored in the candidate list. [This vector is used later in the IE algorithm as the initial vector in the CI procedure (see discussion of Step 8).]

If a feasible solution to IP is known a priori, then in Step 0 this solution is used as the current incumbent solution to IP. Let h denote the objective function value of the current incumbent and let t denote a target value used in the CI algorithm. This target value is set at

$$t = \frac{h}{1 + \varepsilon} \tag{7}$$

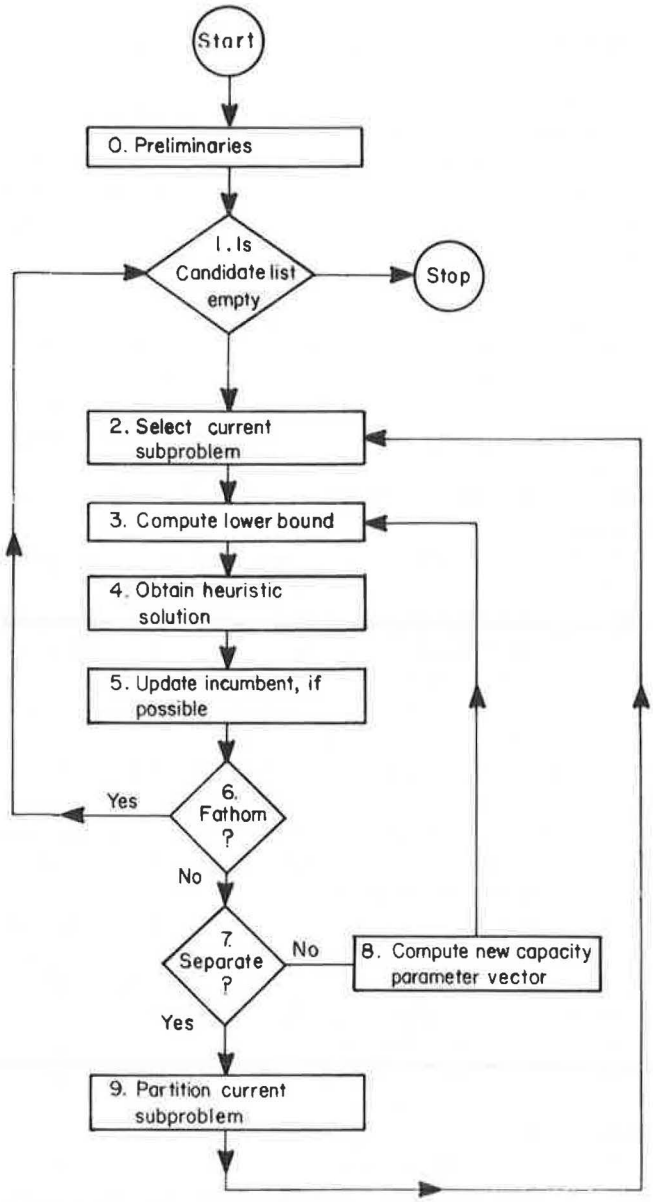

FIGURE 4 Flowchart of implicit enumeration algorithm.

where ε is the prespecified optimality measure. (If no incumbent feasible solution to IP is available, then h and t are set to ∞.)

In Steps 1 and 2, the subproblems in the candidate list are reviewed. If this list is empty, the IE algorithm terminates and the current incumbent is an ε-optimal solution to IP. (If the list is empty and there is no incumbent, then IP is infeasible.) If the candidate list is nonempty, a subproblem is selected on a last-in–first-out (LIFO) basis. Note that a LIFO selection rule generates a depth-first search in the IE enumeration tree, which in turn minimizes the number of subproblems contained in the candidate list at any one time. This is important because the candidate list stores an arc-length capacity vector along with each subproblem specification. Thus, the LIFO subproblem selection rule is used because it minimizes computer memory requirements associated with the candidate list.

In Step 3 a lower bound to $z^*[IP_Y]$ is computed. Here, the current linear subproblem is solved and l_Y, the lower bound to $z^*[IP_Y]$, is computed as follows:

$$l_Y = \min \{t, z^*[LP_Y(\mathbf{u}_Y)]\} \tag{8}$$

This lower bound is guaranteed to be at least as tight as $z^*[LP_Y]$ and computational experience has shown that l_Y can be considerably tighter than $z^*[LP_Y]$ [see work by Lamar et al. *(3)*]. Moreover, the lower bound given in Equation 8 is easy to compute. It involves only the solution of $LP_Y(\mathbf{u}_Y)$ (i.e., a set of shortest-path problems with flow assignment) and a comparison of $z^*[LP_Y(\mathbf{u}_Y)]$ with the target value t.

In Step 4 a new incumbent solution to IP is sought by obtaining a heuristic solution to the current integer subproblem, IP_Y. The rationale of this heuristic procedure is that because $LP_Y(\mathbf{u}_Y)$ is easy to solve, this current linear subproblem can be used repeatedly in determining a feasible solution to the current integer subproblem, IP_Y.

At each iteration of the heuristic procedure, $LP_Y(\mathbf{u}_Y)$ is solved and the links with large flow are selected to be part of the candidate network design of IP_Y. For the next iteration, the fixed charge on these links is assumed to be a sunk investment, and only the variable cost is used in solving $LP_Y(\mathbf{u}_Y)$ again. This iterative process of solving $LP_Y(\mathbf{u}_Y)$ and selecting additional arcs for inclusion in the heuristic network design is repeated until a feasible solution to IP_Y is reached. At this point, the arcs included in the heuristic network design form a path for each OD pair in the market set M. All arcs with zero flow can now be discarded and the procedure terminates. Note that if $LP_Y(\mathbf{u}_Y)$ is feasible, a feasible solution to IP_Y can always be obtained by this process. [For additional discussion of heuristic procedures for LTL network design problems, see work by Powell and Sheffi *(13)* and Sheffi and Powell *(14)*.]

In Step 5, h_Y, the cost of the current heuristic solution for IP_Y, is compared with h, the cost of the current incumbent solution for IP. If $h_Y < h$, the current heuristic solution is retained as the new incumbent solution and the value of h is updated. The target value (t) in Equation 7 is also updated.

In Steps 6 and 7 it is determined whether additional effort should be expended on the current subproblem. In Step 6 it is determined whether $l_Y = t$. If so, the objective function value of all feasible solutions to IP_Y is greater than or equal to $(1 - \varepsilon)h$. Thus, if $l_Y = t$, the current subproblem can be fathomed and the

IE algorithm transfers to Step 1 to review the candidate list. On the other hand, if $l_Y < t$, the algorithm goes to Step 7.

In Step 7 the improvement in l_Y, the lower bound to the current subproblem, is tested by using a relative improvement criterion. If there is sufficient improvement in l_Y, the IE algorithm goes to Step 8 to determine a new capacity parameter vector in an effort to fathom the current subproblem. On the other hand, if the improvement in l_Y is slight, it is more efficient to partition the current subproblem in Step 9 and evaluate each of these new subproblems separately.

In Step 8 the new capacity parameter vector for the current subproblem is determined. As shown by Lamar et al. *(3)*, this step is equivalent to solving a set of linearized knapsack problems, one for each arc in the network. This means that a new capacity parameter vector, $\mathbf{u}_Y$, can be computed very efficiently. After this step has been completed, the algorithm goes to Step 3 to compute a new lower bound for the current subproblem.

In Step 9 a "partitioning" arc (denoted as arc d) is selected and the current subproblem is separated into two new subproblems—one with $Y_d = 0$ and the other with $Y_d = 1$. Remember that the lower bound in this IE procedure is obtained from the CI procedure rather than just the LP relaxation of the subproblem. Thus, partitioning methods based on the LP solution of the new subproblems—such as "up and down penalty" methods [see, e.g., work by Driebeek *(15)* or Tomlin *(16)*]—do not necessarily identify the best partition. Therefore, the partition criterion used here is to select a link whose flow in the current linear subproblem is significantly greater than zero but less than the value of the current improved capacity parameter for that link. This choice of a partitioning arc has performed well in computational experiments.

Once the partitioning link (d) has been determined, two new subproblems—$Y + \{y_d = 0\}$ and $Y + \{y_d = 1\}$—are added to the candidate list in Step 9. The capacity parameter vector for the current subproblem is also placed in the candidate list. (It is used as the initial capacity parameter vector in the CI procedure when each of the new subproblems is evaluated.) The IE algorithm then goes to Step 2 to select the next problem from the candidate list.

The IE algorithm outlined earlier produces an ε-optimal solution to IP or determines that IP is infeasible. (Note that IP is feasible if and only if LP is feasible. Thus, the feasibility of IP is determined in the first pass of the algorithm.) As mentioned earlier, the optimality criterion (ε) must be specified a priori. In cases where there is no natural tolerance (due, for example, to data accuracy), the IE algorithm can be solved several times by starting with a large value of ε. This value is then decreased until the computational effort to obtain an incumbent solution outweighs the incremental value of such a solution.

This IE algorithm has performed well in computational experiments with small LTL networks as shown in the next section.

COMPUTATIONAL EXPERIENCE

In this section the computational performance of the implicit enumeration (IE) procedure is described. It is presented in three parts: the network generation procedure used in the computational tests, the procedure for a small prototype LTL line-haul network, and some computational experience for a series of LTL networks of various sizes and with several different levels of OD shipment demand.

Network Generation Procedure

The network and OD shipment demand generation procedure used in this section is outlined as follows.

For each network generated, the node set (N) is taken as a subset of a set of 50 metropolitan areas distributed throughout the continental United States. Set N is partitioned into two sets: N_E and N_B. Here, N_E contains the set of EOL terminals where shipments originate or terminate, or both, and N_B contains the set of BB terminals where transshipment occurs. The nodes in N_E are randomly selected from among the 50 cities. The nodes in N_B are selected using a p-median heuristic *(17)*, where the $|N_B|$ BB terminals are taken as the p-medians. The medians are selected from the set of 50 cities that are not contained in N_E. (Note that in this paper, the location of all terminals is considered fixed; it is the selection of network links and the flow on these links that is designed.)

The networks generated are fully connected; that is,

$$A = \{a: I(a) \in N, J(a) \in N, I(a) \neq J(a)\}$$

Each arc $a \in A$ is a design link with associated fixed charge f_a and variable cost c_a. In addition, the BB terminals are represented as arcs in order to represent handling costs at these terminals. There are no fixed charges associated with these links.

The set of OD markets (M) consists of all EOL city pairs; that is, $|M| = |N_E|^2 - |N_E|$. For each $m \in M$, the shipment demand (q_m) is generated using a gravity-type model *(18)*. Specifically,

$$q_m = \alpha_0 \frac{(r_m)^{\alpha_1} (s_m)^{\alpha_2}}{(d_m)^{\alpha_3}} \tag{9}$$

where

r_m = daily retail sales at city $O(m)$,
s_m = daily retail sales at city $D(m)$,
d_m = highway mileage from $O(m)$ to $D(m)$, and
$\alpha_0, \alpha_1, \alpha_2, \alpha_3$ = fixed parameters.

The numerical quantities used in the computational experiments are set to reflect a realistic LTL shipping environment. For each link $a \in A$, the fixed charge f_a (in dollars) is taken as the cost of running a single truck per day over that link at \$1.10/mi. The variable cost c_a (in dollars per pound) is taken as the cost of running a single truck over link a assuming an effective payload capacity of 300 CWT. The handling cost at each BB terminal is \$1/CWT. For each $m \in M$, the values for r_m and s_m are taken as the gross retail sales for the associated Basic Trading Area (BTA) adjusted to current dollars. The distances (d_m) are in statute miles. For all networks generated, the value of the parameters in Equation 9 is fixed at $\alpha_1 = 1.0$, $\alpha_2 = 0.5$, and $\alpha_3 = 0.5$. Note that α_1 does not equal α_2, so that nonsym-

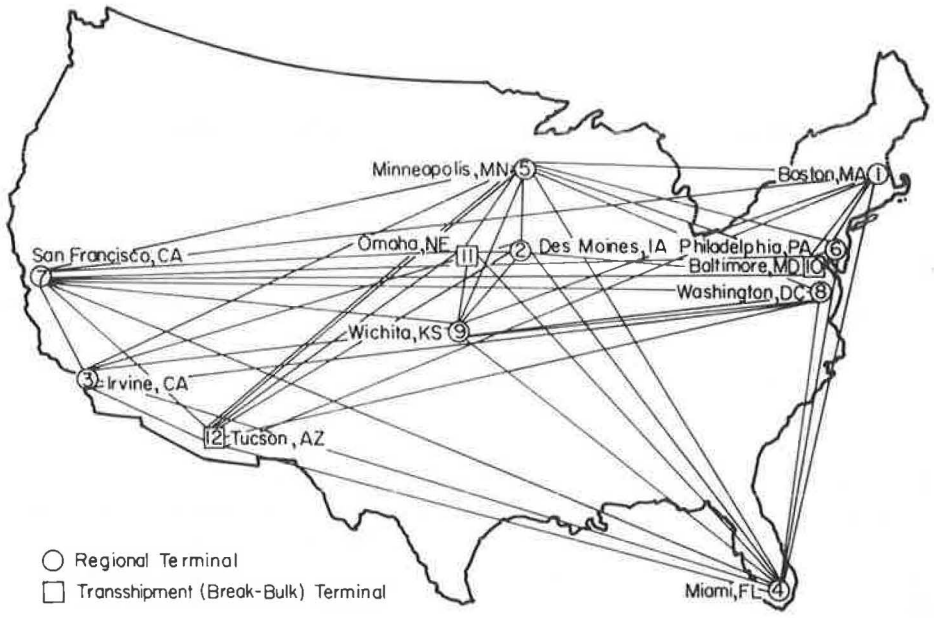

FIGURE 5 Terminals and links for prototype network.

metric shipping demands are modeled. The size of the networks is adjusted by varying $|N_E|$ and $|N_B|$ and the volume of shipment demand is adjusted by varying α_0.

Prototype LTL Load Plan

With the generation procedure just described, a small network was created to illustrate how the procedure described in this paper can aid an LTL motor carrier in designing and evaluating a line-haul network. The hypothetical network is shown in Figure 5. It consists of nine EOL terminals and three BB terminals. The network is completely connected (i.e., there is a link between every pair of terminals). Daily shipment demand between each pair of EOL terminals is given in Table 1.

As mentioned in the previous section, the IE procedure generates a heuristic network design whose objective function value is within $100 \cdot \varepsilon$ percent of the value of the optimal network design. For this example, ε was set at 0.01 (i.e., the heuristic solution is to be within 1 percent of optimal). This solution, shown in Figure 6, can be used by the LTL motor carrier in strategic studies that involve a load-plan design. Note that most of the freight in this example is routed through the BB terminals, although some shipment demand, such as that between San Francisco and Irvine, is sent directly. If the shipment demand between the EOL terminals or the number or location of the EOL or BB terminals is altered, the program can be rerun to suggest modifications in the load plan. It should be noted that such a procedure cannot be used for actual load planning because it ignores many important issues in LTL network design. These include actual trailer routing, equipment balancing, weekly demand cycles, terminal capacity, and many others. It may be applicable, however, to obtaining a general grasp of the costs of strategic alternatives.

The computational performance of the IE procedure is discussed next.

Performance Curves

In the following paragraphs an assessment is made of the tradeoffs between accuracy and computational effort that are associated with the IE procedure for several trial networks generated by the procedure described at the beginning of this

TABLE 1 SHIPMENT DEMAND FOR PROTOTYPE NETWORK

Origin Terminal[a]	Shipment Demand (CWT/day) by Destination Terminal[a]								
	1	2	3	4	5	6	7	8	9
1	–	55	12	101	101	273	91	195	40
2	25	–	3	20	48	29	24	25	16
3	2	1	–	2	2	2	6	2	1
4	89	39	10	–	69	107	70	96	32
5	84	89	11	65	–	99	80	86	44
6	294	69	15	131	127	–	109	417	51
7	105	62	44	92	110	117	–	100	51
8	178	51	11	100	93	354	80	–	38
9	15	13	2	13	19	17	16	15	–

[a]Terminal numbers refer to numbers on Figure 5.

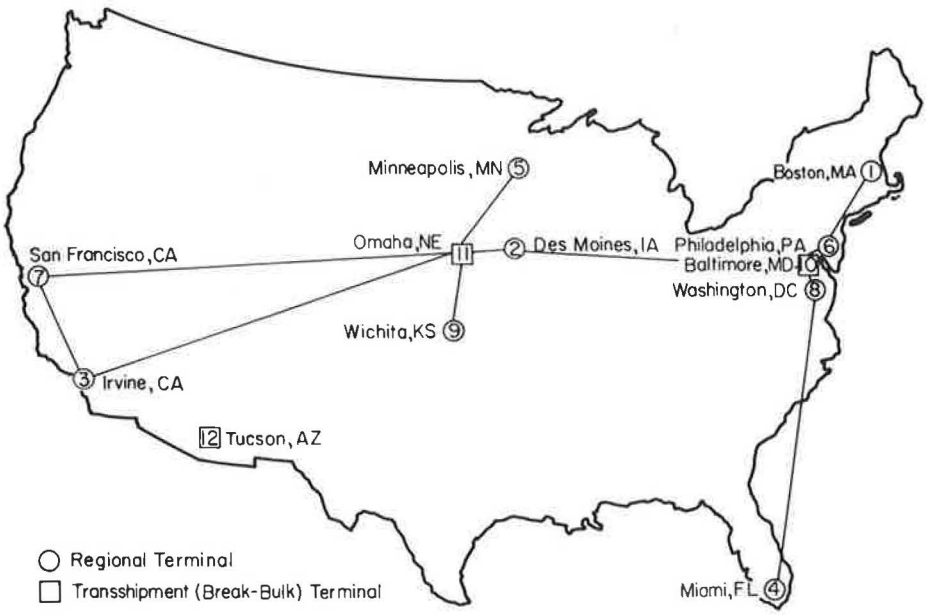

FIGURE 6 Heuristic design for prototype network.

section. For each of the networks tested, the computational time for the IE procedure was measured for various levels of the optimality criterion ε. The resulting performance curves are shown in Figures 7–13, in which ε is plotted versus the CPU seconds on a Digital Equipment Corporation VAX 780 minicomputer.

Figures 7 through 10 show the performance curves of the IE procedure for four networks ranging in size from $|N_E| = 10$, $|N_B| = 2$ [involving 90 OD pairs (markets) and 132 fixed-charge links] to $|N_E| = 40$, $|N_B| = 6$ (involving 1,560 OD pairs and 2,070 fixed-charge links). The dashed line in each figure shows the performance curve for a standard implicit enumeration (SIE) procedure in which the lower bound at each point in the enumeration tree is simply the LP relaxation of the corresponding subproblem. These curves show that the SIE produces relatively little improvement beyond the original LP relaxation (shown as the square symbol in the figures), even with significant computational effort.

The SIE procedure can be compared with the IE procedure presented in this paper. This latter procedure is shown as a solid

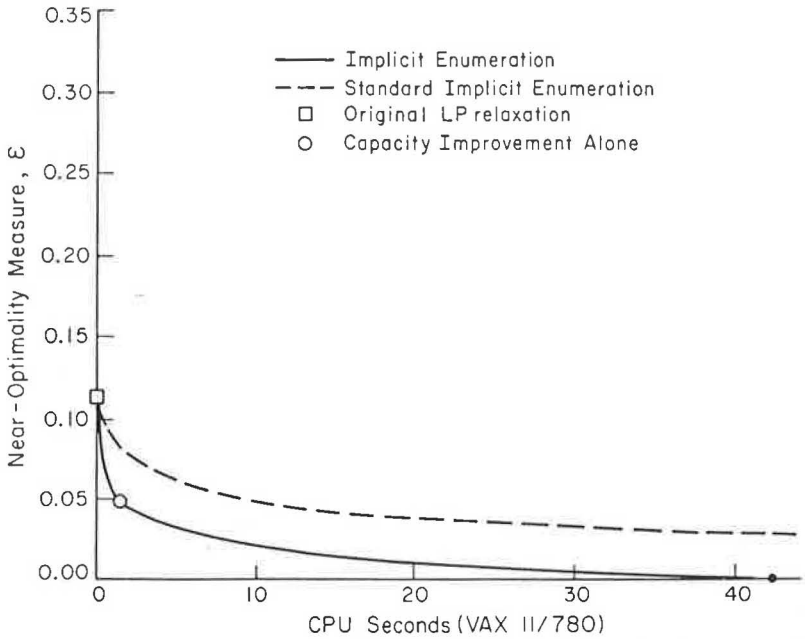

FIGURE 7 Near-optimality measure ε versus CPU time: $|N_E| = 10$, $|N_B| = 2$.

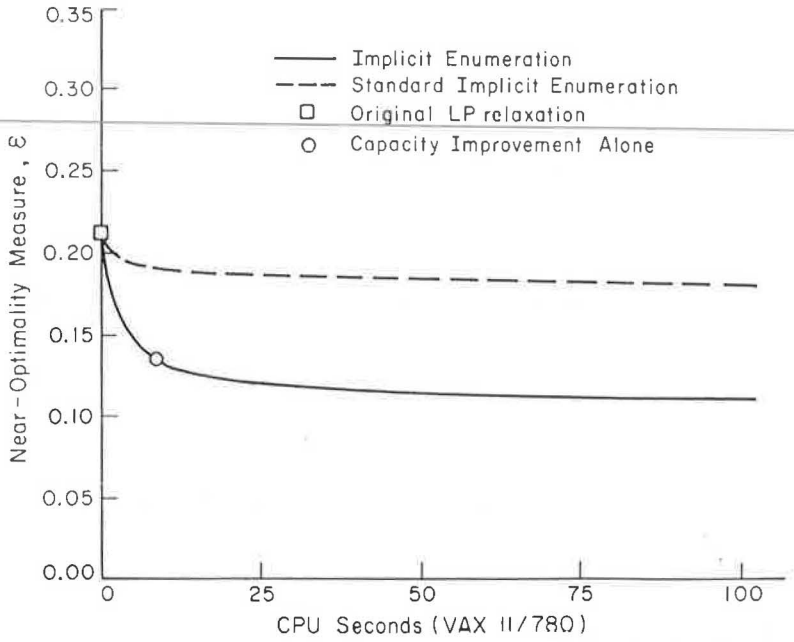

FIGURE 8 Near-optimality measure ε versus CPU time: $|N_E| = 10$, $|N_B| = 6$.

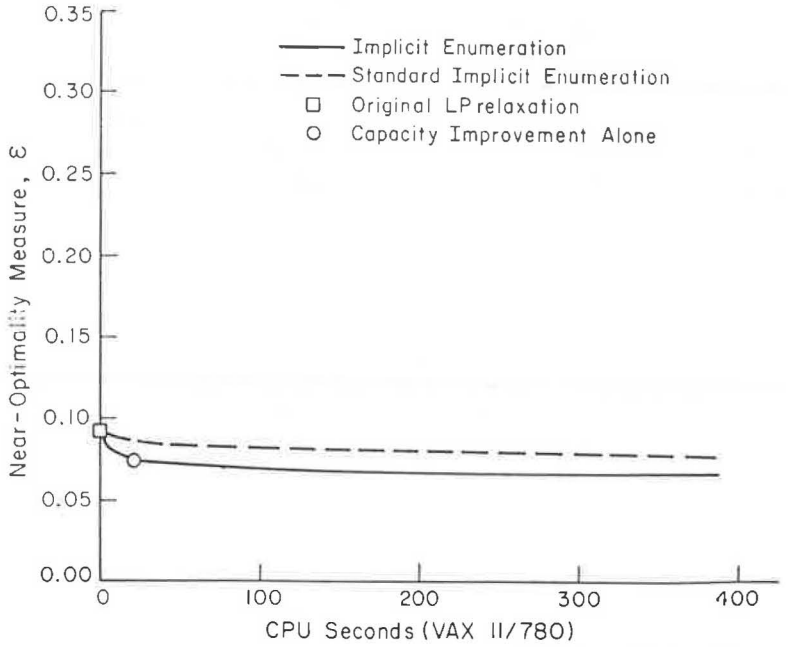

FIGURE 9 Near-optimality measure ε versus CPU time: $|N_E| = 40$, $|N_B| = 2$.

line in the figures. These curves point out the benefit obtained by incorporating the capacity improvement (CI) procedure into the IE algorithm. In fact, the majority of improvement of the IE over the SIE is due to application of the CI procedure to the root-node subproblem of the enumeration tree. This improvement is shown by the circle symbol on the curves. Beyond this point, although additional improvements in ε were obtained by the IE procedure, these improvements were usually achieved at substantial computational cost. Also, for the larger networks, although only small gains were achieved by the IE, this corresponds to essentially no improvement over the original LP relaxation for the SIE.

Note that increasing the number of OD markets increases the overall flow in the network. For the cases in which the flow

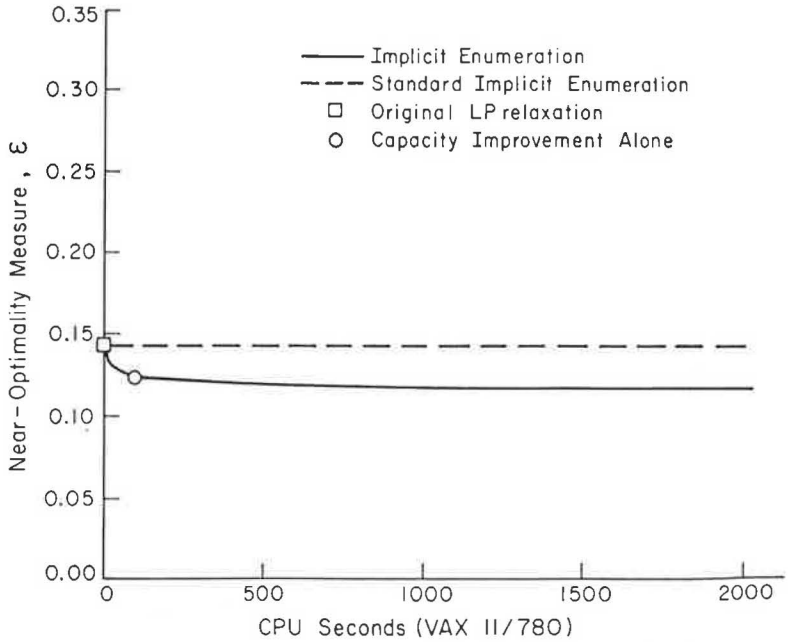

FIGURE 10 Near-optimality measure ε versus CPU time: $|N_E| = 40$, $|N_B| = 6$.

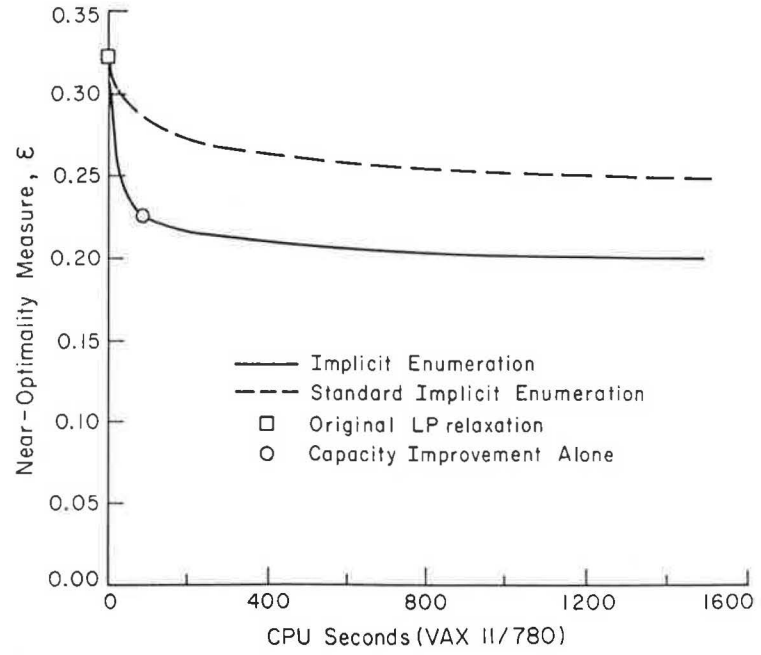

FIGURE 11 Near-optimality measure ε versus CPU time for largest trial network: low flow level.

level is relatively high, flow in an LP subproblem is more likely to be carried on direct links from the origin to destination rather than on links going to transshipment points. Thus, in this case, the CI procedure in the IE has a limited ability to tighten the lower bound to a subproblem. This inability, combined with the closeness of the original LP lower bound to z^*[IP], explains why the effect of the CI procedure in the IE diminishes as the size of the network increases in Figures 7–10.

To test the effects of flow level directly, the largest network ($|N_E| = 40$, $|N_B| = 6$) was optimized with three different flow levels: low, medium, and high. (The four networks exhibited in Figures 7–10 fall into the medium-flow category.) The volume

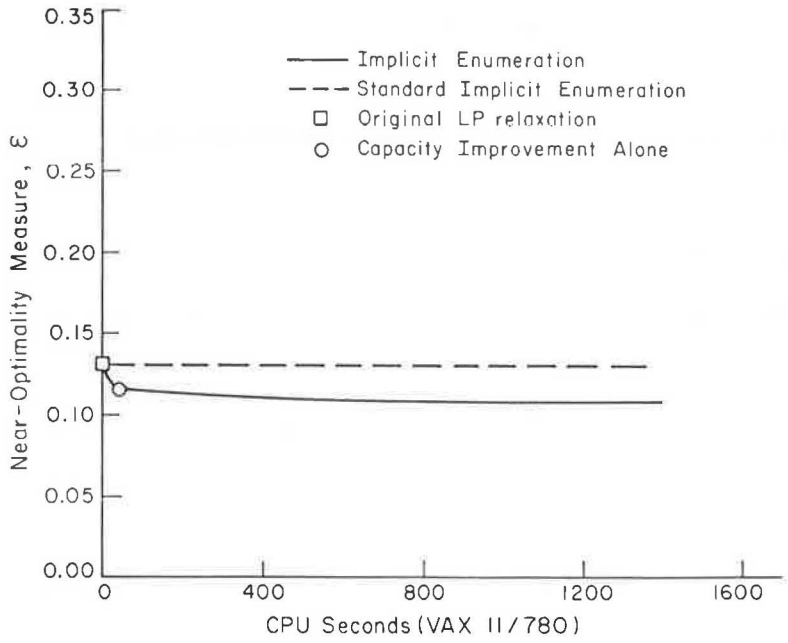

FIGURE 12 Near-optimality measure ε versus CPU time for largest trial network: medium flow level.

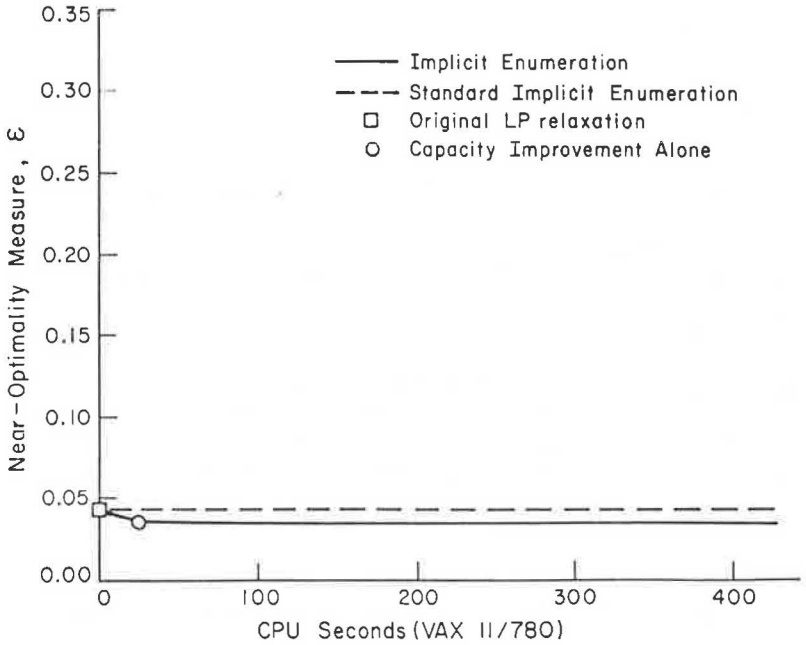

FIGURE 13 Near-optimality measure ε versus CPU time for largest trial network: high flow level.

of flow in the network can be characterized by the average link flow in the heuristic solution. For low, medium, and high flow, the average nonzero link flow in the heuristic solution was one-third, five, and eight truckloads, respectively. The fixed charge in all cases was equivalent to the cost of carrying one truckload. The results are shown in Figures 11–13. [Note that curves for the medium case (Figure 12) are not identical to those for the largest network (Figure 10) because the networks in each case were generated separately.] The designation of the SIE and IE curves is the same as that used in Figures 7–10. As expected, the IE always outperforms the SIE, but this improvement diminishes as the flow level increases.

In reviewing Figures 7–13, it seems that where optimality is not attained, the IE (and SIE) procedure cannot overcome

certain values of ε. That is, for small tolerances, the exponential growth of the enumeration tree predominates and the IE curves track those of the SIE. This overall pattern reflects the difficult nature of the problem.

The IE procedure presented in this paper appears to have strong merits, particularly for situations with relatively low levels of flow, as in LTL freight networks. Markets in LTL line-haul networks are characterized by flows that are at the low range of these computational experiments. The reason for this is that high link flow levels would typically indicate an opportunity for bypassing a transshipment point or opening a new terminal and consequently LTL carriers operate at the low flow range on most of their links (excluding links between break-bulks, which many operate with very high flows). Thus, the design problem has to do with the portions of the LTL motor carrier's network with low flow. In these cases, the fixed charges correspond to a minimum frequency of, say, one truck-load per day and arc flows are in the range of one-third to two truckloads per day. It is in this interesting range that the IE procedure yields the most improvement.

ACKNOWLEDGMENTS

This research was supported by I. U. International, Inc., through a research contract with Massachusetts Institute of Technology and Princeton University. The authors would like to thank John Terry and Al Pinkerton of I. U. International, Inc., and the personnel in the Industrial Engineering Department of Ryder/PIE truck lines for their assistance and support throughout this project.

REFERENCES

1. A. W. LaMond. *Competition in the General-Freight Motor-Carrier Industry.* Lexington Books, Lexington, Mass., 1980.
2. W. B. Powell and Y. Sheffi. The Load Planning Problem of Motor Carriers: Problem Description and a Proposed Solution Approach. *Transportation Research*, Vol. 17A, No. 6 (Nov.), 1983, pp. 471–480.
3. B. W. Lamar, Y. Sheffi, and W. B. Powell. *A Lower Bound Procedure for Uncapacitated, Multicommodity Fixed Charge Network Design Problems.* Report UCI-ITS-WP-86-4. Institute of Transportation Studies, University of California, Irvine, 1986.
4. R. T. Wong. *Accelerating Bender's Decomposition for Network Design.* Ph.D. dissertation. Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, 1978.
5. R. L. Rardin and U. Choe. *Tighter Relaxations of Fixed Charge Network Flow Problems.* Report J-79-18. School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, 1979.
6. A. Balakrishnan. *Valid Inequalities and Algorithms for the Network Design Problem with an Application to LTL Consolidation.* Ph.D. dissertation. Sloan School of Management, Massachusetts Institute of Technology, Cambridge, 1984.
7. T. L. Magnanti and R. T. Wong. Network Design and Transportation Planning: Models and Algorithms. *Transportation Science,* Vol. 18, No. 1 (Feb.), 1984, pp. 1–55.
8. M. L. Balinski. Fixed-Cost Transportation Problems. *Naval Research Logistics Quarterly*, Vol. 8, No. 1 (March), 1961, pp. 41–54.
9. R. W. Floyd. Algorithm 97—Shortest Path. *ACM: Communications of the ACM,* Vol. 5, No. 6 (May), 1962, p. 345.
10 E. Moore. The Shortest Path Through a Maze. In *Proceedings of the International Symposium on the Theory of Switching,* Harvard University Press, Cambridge, Mass., 1957, pp. 285–292.
11. F. Glover, D. Karney, and D. Klingman. Implementation and Computational Comparisons of Primal, Dual, and Primal-Dual Computer Codes for Minimum Cost Network Flow Problems. *Networks*, Vol. 4, No. 3 (Fall), 1974, pp. 191–212.
12. A. M. Geoffrion and R. E. Marsten. Integer Programming Algorithms: A Framework and State-of-the-Art Survey. In *Perspectives on Optimization: A Collection of Expository Articles* (A. M. Geoffrion, ed.), Addison-Wesley, Reading, Mass., 1972.
13. W. B. Powell and Y. Sheffi. Design and Implementation of an Interactive Optimization System for Network Design in the Motor Carrier Industry. *Operations Research,* in press.
14. Y. Sheffi and W. B. Powell. Interactive Optimization for Motor Carrier Load Planning. *Journal of Business Logistics*, Vol. 7, No. 2, 1986.
15. N. J. Driebeek. An Algorithm for the Solution of Mixed Integer Programming Problems. *Management Science,* Vol. 12, No. 7 (March), 1966, pp. 576–587.
16. J. A. Tomlin. An Improved Branch-and-Bound Method for Integer Programming. *Operations Research,* Vol. 19, No. 4, (July–Aug.), 1971, pp. 1070–1075.
17. M. B. Teitz and P. Bart. Heuristic Methods for Estimating the Generalized Vertex Median of a Weighted Graph. *Operations Research,* Vol. 16, No. 5 (Sept.-Oct.), 1968, pp. 955–961.
18. P. R. Stopher and A. H. Meyburg. *Urban Transportation Modelling and Planning.* D.C. Heath and Co., Lexington, Mass., 1975.