Bounding Procedures for Fixed Charge, Multicommodity Network Design Problems

by

Bruce W. Lamar*
Yosef Sheffi*
Warren B. Powell**

December, 1984

*Department of Civil Engineering, Massachusetts Institute of Technology, Cambridge, MA
**Department of Civil Engineering, Princeton University, Princeton, NJ

ABSTRACT

A new lower bound for fixed charge, uncapacitated, multicommodity network design problems is presented. This bound is significantly tighter than the simple LP relaxation of the problem. An upper bound (and a candidate network design) is produced using a link inclusion heuristic. The lower and upper bounding procedures are incorporated into an implicit enumeration algorithm. This algorithm is applied to the load planning problem of less-than-truckload (LTL) motor carriers. Computational results show that the algorithm compares favorably with other methods reported in the literature.

INTRODUCTION

Fixed charge network design problems concern flows over links which cannot be used before a quantum level of cost is incurred. This quantum may represent, for example, capital investment of a new facility or the minimum level of resources necessary to operate a given service.

The work reported here is motivated by the load planning problems of less-than-truckload (LTL) motor carriers. These carriers haul freight from many origings to many destinations. The freight is collected in city terminals and carried to one or more breakbulk (transshipment) terminals where it is sorted and reconsolidated. It is then moved to destination city terminals where it is distributed to the individual customers. The load planning problem concerns the structure of the network used to haul the freight between these terminals. Thus the relevant network includes the terminals as nodes and the direct services between them as links. As argued by Powell and Sheffi (1983), a direct service between any two terminals is always operated with certain minimum frequency. Consequently, the carrier's problem of deciding which links to include in the network can be viewed as a fixed charge network design problem. The problem is uncapacitated since beyond the minimum frequency, each link can carry practically any flow level. It has multi-commodity aspects, since shipments between every origin-destination pair represent a distinct flow.

As shown by Johnson et al (1978), network design problems are NP-hard, meaning that (probably) no efficient solution procedure exists for large scale networks. Some of the largest networks for which optimal solutions have been obtained are reported by Magnanti, Mireault, and Wong (1984) and by Barr, Glover, and Klingman (1981). Magnanti et al applied Bender's decomposition to obtain optimal network design solutions for uncapacitated general networks containing 30 nodes and 90 fixed-charge links. Barr et al tailored the branch-and-bound penalty procedure proposed by Driebeek (1966) to obtain optimal solutions for uncapaciated bipartite (i.e. transportation-type) networks containing 50 source nodes, 150 sink nodes, and 600 fixed-charge links.

For large-scale network design problems (involving, say, over 50 nodes and 1000 fixed-charge links), analysts have typically relied on heuristic procedures to provide near-optimal solutions. For example, Powell and Sheffi (1983) used a local improvement heuristic for uncapaciated network design problems with fixed-plus-piecewise linear link cost functions to optimize networks containing more than 300 nodes and 18,000 such links. [An interactive optimization model for this same problem, which is being used in practice by a major carrier, is described by Sheffi and Powell (1985).] Wong (1980) used a link inclusion heuristic procedure for uncapaciated problems with a budget constraint for networks containing 100 nodes and nearly 5000 fixed-charge links. Steenbrink (1974) applied an incremental traffic assignment heuristic for network design problems with link capacities and nonlinear link cost functions to obtain a feasible design for a network containing approximately 2000 nodes and 6000 links. A comprehensive review of other exact and heuristic procedures for network design problems has been compiled by Magnanti and Wong (1984a).

This paper presents an implicit enumeration (branch and bound) method for obtaining near-optimal network designs for uncapacitated, multicommodity problems. The focus of the paper is on a procedure for calculating lower bounds. This procedure generates bounds which are significantly tighter than the simple linear programming (LP) relaxation. Coupled with a network design heuristic (which generates a cost upper bound), the implicit enumeration provides solutions which compare favorably with other methods reported in the literature.

This paper is organized as follows. Section 1 discusses the problem formulation and outlines a solution procedure. Section 2 focuses on the development of a lower bound. Section 3 presents the lower bound algorithm and a heuristic algorithm for obtaining an upper bound. Section 4 describes the implicit enumeration procedure, and Section 5 reports some comparative numerical results.

1. PROBLEM FORMULATION AND SOLUTION OUTLINE

This section is comprised of three parts. The first formulates the fixed charge network design problem addressed here as an integer program (IP), the second discusses an LP relaxation of thisformulation, and the third outlines the solution methodology for the IP (which uses the LP relaxation).

- IP Formulation

The following notation is used to define the problem. Let

$\underset{\sim}{N}$ = set of nodes with generic element n

$\underset{\sim}{A}$ = set of directed arcs with generic element a

$\underset{\sim}{M}$ = set of origin-destination (OD) pairs ("markets") with generic element m

The freight movement demand is characterized by a market flow vector, $\underline{q} = (\ldots, q_m, \ldots)$. The units of this demand are, for example, CWT/week or cubic feet/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, while c_a represents the (average) marginal cost of hauling freight over arc a. Figure 1 shows the arc cost as a function of flow for a typical arc.

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 m-th OD pair), and $x_{a,m} = 0$ otherwise. Thus $\underline{y} = (\ldots, y_a, \ldots)$ is the network design vector, and $\underline{x} = (\ldots, x_{a,m}, \ldots)$ is the flow routing vector.

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

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

[When no other indication is given, the notation "Σ" and "$\forall$" refer to all members of the relevant set. Thus in eq. (1), "Σ_m" is equivalent to "$\Sigma_{m\varepsilon\underset{\sim}{M}}$", and "$\forall a$" is equivalent to "$\forall\ a\varepsilon\underset{\sim}{A}$".] A parameter used later in the program formulation is the capacity of each arc, denoted by the vector $\underline{u} = (\ldots,\ u_a, \ldots)$. Since 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 "from" node and "to" node, respectively, of arc a (i.e. if $I(a) = i$ and $J(a) = j$, then arc a goes from node i to node j where $a\varepsilon\underset{\sim}{A}$ and $i,j\varepsilon\underset{\sim}{N}$). Similarly, let $O(m)$ and $D(m)$ define the origin and destination nodes, respectively, for market m (i.e. if $O(m) = r$ and $D(m) = s$, then market m includes flow from node r to node s, where $m\varepsilon\underset{\sim}{M}$ and $r,s\varepsilon\underset{\sim}{N}$). In addition, let $\underset{\sim}{A}_n$ be the set of arcs whose "from" node is n (i.e., $a\varepsilon\underset{\sim}{A}_n$ if $I(a) = n$), and let $\underset{\sim}{B}_n$ be the set of arcs whose "to" node is n (i.e. $a\varepsilon\underset{\sim}{B}_n$ if $J(a) = n$).

The multicommodity network flow conservation constraints can now be written as

$$\sum_{a\varepsilon\underset{\sim}{A}_n} x_{a,m} - \sum_{a\varepsilon\underset{\sim}{B}_n} x_{a,m} = \left\{\begin{array}{lll} 1 & \text{if} & n = O(m) \\ -1 & \text{if} & n = D(m) \\ 0 & \text{for all other } n\varepsilon\underset{\sim}{N} & \end{array}\right\} \quad \forall\ n,m \qquad (2a)$$

$$x_{a,m} \geq 0 \qquad \forall\ a,m \qquad (2b)$$

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

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

Program P:

$$\min_{\underline{x} \in \underset{\sim}{X}} z = \sum_m \sum_a c_a q_m x_{a,m} + \sum_a f_a y_a \qquad (3a)$$

subject to

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

$$y_a \varepsilon \{0,1\} \qquad \forall a \qquad (3c)$$

Let $(\underline{x}^*, \underline{y}^*)$ denote the optimal solution vector, let z^* denote the optimal objective function value, and let $\underline{v}^*$ denote the optimal link flow vector for program P. 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 $\underset{\sim}{X}$ guarantees that the feasible solutions are restricted to directed paths over the network. Constraints (3b) guarantees 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 $x_{a,m} \varepsilon \{0,1\} \forall a,m$ due to the network structure of the problem]

Note that alternatively constraints (3b) can be written in a disaggregate form:

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

While this form generates a tighter LP relaxation than the LP relaxation of P, the advantage of using constraint (3b) is that the LP relaxation is then particularly easy to solve. This ease is exploited in the procedure which calculates the lower bound for P, as shown in Section 2.

- LP Relaxation

The linear programming relaxation is formed by replacing constraints (3c) with the nonnegativity constraints

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

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

To see how program $\overline{P}$ can be solved, note that it can be expressed equivalently as

$$\min_{\underline{x}\epsilon\underset{\sim}{X}} z = \sum_m \sum_a q_m(c_a + \frac{f_a}{u_a})x_{a,m} \tag{6}$$

The equivalence between program $\overline{P}$ as formulated in (6) and its original formulation [(3a), (3b)] can be shown on the basis of Balinski's (1961) observation that constraints (3b) will always be satisfied with equality in the optimal solution of $\overline{P}$. Since $u_a > 0 \;\forall\, a$, constraint (3b) can be solved explicitly for y_a as

$$y_a = \frac{1}{u_a} \sum_m q_m x_{a,m} \tag{7}$$

This expression can be substituted for y_a in the objective function (3a) to obtain the formulation of $\overline{P}$ given in (6).

[The equivalency between the original formulation of $\overline{P}$ and that given in (6) can also be shown by forming a Lagrangian relaxation of the original formulation with respect to constraint (3b). Since the variables $\{y_a\}$ are unconstrained in the original formulation, the objective function of the Lagrangian will remain finite only if the value of the dual variable associated with each constraint in (3b) is equal to f_a/u_a. The formulation in (6) follows immediately.]

As mentioned in connection with eq. (4), program $\overline{P}$ is easy to solve. The reason 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 (e.g. Floyd (1962)) or by repeated application of a one-to-many algorithm (e.g. Moore (1957)). These algorithms, coupled with recent list processing techniques (see, for example, Glover et al (1974)) are very efficient. Consequently, $\overline{P}$ can be solved repeatedly in the course of finding both a lower bound and an upper bound to the optimal solution of the integer program, P.

- Preview of Solution Procedure

The implicit enumeration used in this paper is based on a branch and bound (B & B) procedure in which a lower bound is generated at each node of the B & B tree. This lower bound can be improved iteratively, up to a point at which a decision to branch is reached (if no fathoming occurs). The details of the procedure are given in Section 4. The following paragraphs outline the lower bound procedure so that the reader can put the material in Sections 2 and 3 in perspective.

The lower bound procedure is based on an iterative solution of program $\overline{P}$ in which the capacity parameters $\{u_a\}$ are systematically reduced. To describe the process, let $\overline{P}(\underline{u})$ denote program $\overline{P}$ with parameter vector $\underline{u} = (\ldots, u_a, \ldots)$. The optimal value of the objective function of this program is $\overline{z}^*(\underline{u})$.

For a given set of parameters $\underline{u}^\dagger = (\ldots, u_a^\dagger, \ldots)$, program $\overline{P}(\underline{u}^\dagger)$ can be solved to obtain the optimal value $\overline{z}^*(\underline{u}^\dagger)$. If the capacity parameters are set so that they do not restrict the (unknown) optimal flow vector of the integer program P (denoted $\underline{v}^* = (\ldots, v_a^*, \ldots)$), then $\overline{z}^*(\underline{u}^\dagger)$ is a valid lower bound to z^*. Moreover, if $\underline{u}^{\dagger} \leq \underline{u}$, then the feasible region of $\overline{P}(\underline{u}^{\dagger})$ is contained within that of $\overline{P}(\underline{u})$, meaning that $\overline{z}^*(\underline{u}^{\dagger}) \geq \overline{z}^*(\underline{u})$. In other words, the lower bound generated by solving $\overline{P}(\underline{u}^{\dagger})$ is better than that generated by solving $\overline{P}(\underline{u})$ (which is the original LP relaxation).

The lower bound $\overline{z}^*(\underline{u}^{\dagger})$ can be tightened by iteratively reducing the value of $\underline{u}^{\dagger}$ and solving $\overline{P}(\underline{u}^{\dagger})$. The difficulty here is that for $\overline{z}^*(\underline{u}^{\dagger})$ to be a valid lower bound, $\underline{u}^{\dagger}$ must be greater than or equal to the unknown optimal solution vector, $\underline{v}^*$. The lower bound procedure described in Sections 2 and 3 provides a mechanism for determining those values of $\underline{u}^{\dagger}$ that produce high values of $z^*(\underline{u}^{\dagger})$ while ensuring that $z^*(\underline{u}^{\dagger})$ is a valid lower bound to z^*.

2. DEVELOPMENT OF A LOWER BOUND

This section develops a procedure for obtaining a lower bound to tne optimal value of (the integer program) P by changing the link capacity parameter vector $\underline{u}$. The presentation is divided into three separate parts: The first presents a capacity improvement program for a single link and discusses discusses its use, the second outlines and explains an algorithm for solving this program, and the third demonstrates how the results of this algorithm can be used to define a lower bound to the optimal objective function value of the original fixed charge program, P.

- Single Link Capacity Improvement Program

Consider a particular link $b \in \underset{\sim}{A}$. The following paragraphs describe a subprogram and a program that are used to obtain a better (smaller) value of u_b (denoted u_b^+). The subprogram, referred to as $\overline{P}_b(w_b)$, is as follows:

Program $\overline{P}_b(w_b)$:

$$\min_{\underline{x} \in \underset{\sim}{X}} z_b(w_b) = \sum_m \sum_a q_m(c_a + \frac{f_a}{u_a})x_{a,m} \tag{8a}$$

subject to

$$\sum_m q_m x_{b,m} \geq w_b \tag{8b}$$

Note that program $\overline{P}_b(w_b)$ is focused on the single network arc b. This program is simply program $\overline{P}(\underline{u})$ augmented with constraint (8b) which forces the flow on link b to be equal to or greater than the parameter w_b. [Note that program $\overline{P}_b(w_b)$ is also a function of parameter vector $\underline{u}$. The evaluation of $\overline{P}_b(w_b)$, however, will not involve changes in $\underline{u}$. Therefore, $\underline{u}$ is not explicitly denoted as a parameter in this program.] Program $\overline{P}_b(w_b)$ can be evaluated parametrically as a function of w_b, starting with $w_b = 0$. As w_b increases past $\overline{v}_b^*$ [which is the flow on link b at the optimal solution to $\overline{P}(\underline{u})$], program $\overline{P}_b(w_b)$ becomes more constrained and its optimal objective function value, $\overline{z}_b^*(w_b)$, increases. In the domain

$$\overline{v}_b^* < w_b < u_b \tag{9}$$

$\overline{z}_b^*(w_b)$ is an increasing function of w_b, as shown in Figure 2. At $w_b = \overline{v}_b^*$, the value of the objective function, $\overline{z}_b^*(w_b)$, equals $\overline{z}^*$ [which is the optimal objective function of $\overline{P}(\underline{u})$] since, at this point, constraint (8b) is non-binding. Now let $\overline{z}_b^{max}$ denote the highest finite value of the objective function, $\overline{z}_b^*(w_b)$, as w_b approaches u_b from below, as shown in Figure 2. As w_b changes between $\overline{v}_b^*$ and u_b, the objective function value, $\overline{z}_b^*(w_b)$ changes between $\overline{z}^*$ and $\overline{z}_b^{max}$.

Program $\overline{P}_b(w_b)$ is used as the subprogram to the following program which is also focused on link $b\varepsilon\underset{\sim}{A}$:

Program $\overline{Q}_b(t)$:

$$\min \quad w_b \tag{10a}$$

subject to

$$\overline{z}_b^*(w_b) \geq t \tag{10b}$$

Program $\overline{Q}_b(t)$ is referred to as the single link capacity improvement program, and the parameter, t, is referred to as the target value. Let $\overline{w}_b^*(t)$ denote the optimal objective function of program $\overline{Q}_b(t)$. Observe that for t in the domain

$$\overline{z}^* < t < \overline{z}_b^{max} \tag{11}$$

the objective function value, $\overline{w}_b^*(t)$, remains finite. It is the minimum amount of flow that must be carried on link b in order to cause $\overline{z}_b^*(w_b)$ to be at least as large as t. This objective function value is used to define, $u_b^{\dagger}(t)$, the new capacity parameter for link b given a target value t. Specifically,

$$u_b^{\dagger}(t) = \begin{cases} 0 & \text{for} \quad t \leq \bar{z}^* \\ \bar{w}_b^*(t) & \text{for} \quad \bar{z}^* < t < \bar{z}_b^{max} \\ u_b & \text{for} \quad t \geq \bar{z}_b^{max} \end{cases} \tag{12}$$

The function $u_b^{\dagger}(t)$ is shown in Figure 3. This function can be viewed as the inverse of the function in Figure 2. That is, one enters with a target value t to obtain a new capacity parameter $u_b^{\dagger}$.

- Single Link Capacity Improvement Algorithm

Once again, consider a particular link $b \in \underline{A}$. For a target value, t, in domain (11), the calculation of $u_b^{\dagger}(t)$, the revised capacity parameter for link b, is based on solving program $\overline{Q}_b(t)$. Note that $\bar{z}_b^*(w_b)$ in constraint (10b) of program $\overline{Q}_b(t)$ is nonlinear but convex in w_b. One method of determining $\bar{w}_b^*(t)$ is to selectively increase w_b and solve the linear program $\overline{P}_b(w_b)$ until the minimum value of w_b such that $\bar{z}_b^*(w_b) \geqq t$ is obtained. The following paragraphs, however, describe an efficient method for obtaining $\bar{w}_b^*(t)$ directly for any value of t in domain (11).

The procedure is based on a given solution of $\overline{P}(\underline{u})$. The first step in the procedure is to determine the cost difference between (i) the optimal routing of 1 unit of flow in market m (i.e., going from O(m) to D(m)) using link b and, (ii) the current routing of the flow in market m (which is optimal in $\overline{P}(\underline{u})$ without the constraint that link b has to be used). Denote this marginal difference by $\Delta_{b,m}$.

The calculation of $\Delta_{b,m}$ is accomplished by decomposing the cost of using link b for market m into three components:

(i) The unit cost from O(m) to I(b)

(ii) The unit cost along link b

(iii) The unit cost from J(b) to D(m)

The marginal difference $\Delta_{b,m}$ is the sum of these three cost components minus the unit cost on the shortest path between O(m) and D(m) in the current solution. All three cost components above (as well as the cost of the current minimum path) are directly available in the current solution of $\overline{P}(\underline{u})$. These various cost components are shown in Fig. 4.

After the marginal differences have been calculated, all markets are sorted in ascending order of $\{\Delta_{b,m}\}$, and the market index, m, is relabeled accordingly. (Thus $\Delta_{b,1}$ is the lowest marginal difference; $\Delta_{b,2}$ is the second lowest, and so on.)

The procedure for calculating $u_b^+(t)$ can now be summarized as follows:

```
Algorithm $G_1$:
    Inputs:  A solution to $\overline{P}(\underline{u})$
             A target value, t
             A designated link, b

    Output:  $u_b^+(t)$

    Step 0:  Preliminaries
        (a)  Set $\tilde{z}_b \leftarrow \overline{z}^*(\underline{u})$, set $w_b \leftarrow 0$, and set $m \leftarrow 1$.
        (b)  If $t \leq \overline{z}^*(u)$
             then go to Step 3.
        (c)  Calculate $\Delta_{b,m}\ \forall\ m \in \underset{\sim}{M}$ and rank all OD pairs accordingly.

        (d)  Set $m_b^{max} \leftarrow \max\{m:\ \Delta_{b,m} < \infty\}$
```

```
Step 1: Stopping Criterion
    (a) If $\tilde{z}_b + \Delta_{b,m} \cdot q_m \geq t$
        then set $w_b \leftarrow w_b + \frac{t - \tilde{z}_b}{\Delta_{b,m}}$, and go to Step 3.
    (b) If $m = m_b^{max}$
        then set $w_b \leftarrow u_b$, and go to Step 3.
Step 2: Update
    (a) Set $w_b \leftarrow w_b + q_m$, set $\tilde{z}_b \leftarrow \tilde{z}_b + \Delta_{b,m} \cdot q_m$.
    (b) Set $m \leftarrow m + 1$, and go to Step 1.
Step 3: Termination
    Set $u_b^{\dagger}(t) \leftarrow w_b$
```

Algorithm G_1 uses the intermediate variables w_b and $\tilde{z}_b$, respectively, to carry the intermediate values of $u_b^{\dagger}$ and $\bar{z}_b^*(u_b^{\dagger})$ throughout. The algorithm works by increasing the flow on link b iteratively, while keeping the cost penalty for doing so at a minimum. It is essentially a "greedy" procedure assigning OD flow to paths that use arc b by the least cost ranking implied by $\{\Delta_{b,m}\}$.

Algorithm G_1 can be followed by studying Figure 5. In Step 0, if the target value is less than the optimal objective function value of Program $\overline{P}(\underline{u})$, then the algorithm is superfluous and stops immediately; otherwise the search for $u_b^{\dagger}$ starts from zero. In Step 2, the value of $u_b^{\dagger}$ is incremented to the next "corner point" in Figure 5, and the next OD flow is considered. This process is continued until one of the stopping criteria in Step 1 is met.

The criterion in Step 1a comes into play when t is in domain (11), whereas the criterion for Step 1b is invoked when $t \geq \bar{z}_b^{max}$. For Step 1a, the most recently assigned OD flow is split between its new path (using link b) and its old path (in the optimal solution of $\overline{P}(\underline{u})$). This split is based on a simple linear interpolation of the cost function which is valid since $\bar{z}_b^*(w_b)$ is evaluated only between adjacent "corner points" and this function is linear in this range. Again, Figure 5 helps illustrate this point. At the end, algorithm G_1 terminates with the value of $u_b^+(t)$ given in eq. (12) for any value of t. This result is stated more formally in Appendix A which interprets the steps of algorithm G_1 in the framework of a Lagrangian relaxation procedure.

Note that the ranking of $\Delta_{b,m}$, the marginal differences, in Step 0c can be carried out using any complete sorting method. Significant computational savings can be attained, however, with a modified bin sort (see, for example, Aho et al (1983)). The idea here is to sort only groups of OD pairs with no in-group sorting. Steps 1 and 2 are then modified for groups of OD pairs. The only group that has to be sorted internally is the one containing the OD market which brings the objective function value equal to the target value.

The most important feature of this procedure is that it relies exclusively on information available directly from the solution of $\overline{P}(u)$. This is another way of saying that once program $\overline{P}(u)$ has been solved, the solution to the capacity improvement program $\overline{Q}_b(t)$, can be easily determined for any link $b \varepsilon \underset{\sim}{A}$ and any target value t. This fact is used to develop the lower bound procedure presented in Section 3. First, however, it must be shown that the improved capacity parameter which results from algorithm G_1 is, in fact, a valid lower bound to z*. This is done next.

- Lower Bound Using Improved Capacity Parameters

The importance of the procedure outlined in Algorithm G_1 is that the result can be used to obtain a lower bound to the optimal objective function of integer program P. This is shown in the following two results:

Lemma 1

If $t > z^*$, then $u_b^\dagger(t) > v_b^*$

This lemma states that if t, a target value, is strictly greater than z*, the (unknown) optimal objective function value in the integer program, then $u_b^\dagger(t)$, the new capacity parameter resulting from the application algorithm G_1, is strictly greater than v_b^*, the flow on link b in the optimal solution of P.

Proof

First, note that for $t \leq \overline{z}^*(\underline{u})$, the lemma is vacuously true and that for $t \geq \overline{z}_b^{max}$, the lemma is obviously true since then $u_b^\dagger = u_b$. Thus attention can be restricted to t in domain (11), in which case $u_b^\dagger(t) = \overline{w}_b^*(t)$ as defined in eq. (12). Next observe that program $\overline{P}_b(w_b)$ is formed by adding the single constraint $v_b \geq w_b$ to program $\overline{P}(\underline{u})$ and that the feasible region of program $\overline{Q}_b(t)$ requires $\overline{z}_b^*(\overline{w}_b^*(t)) \geq t$.

By hypothesis $\overline{z}_b^*(\overline{w}_b^*(t)) \geq t > z^*$. Consequently, the optimal solution of P is not contained in the feasible region of $\overline{P}_b(\overline{w}_b^*(t))$. But program $\overline{P}(\underline{u})$ must contain the optimal solution to P (since it is its LP relaxation). Thus it must be that the optimal solution to P is on the "other side" of constraint $v_b \geq \overline{w}_b^*(t)$ (which was added to $\overline{P}(\underline{u})$) meaning that $u_b^\dagger(t) = \overline{w}_b^*(t) > v_b^*$. This completes the proof.

Now let $\underline{u}^\dagger(t) = (\ldots, u_b^\dagger(t), \ldots)$. That is, for a given t, algorithm G_1 is used separately for each $b \varepsilon \underset{\sim}{A}$ to determine the element $u_b^\dagger(t)$ in the link vector $\underline{u}_b^\dagger(t)$. The result of Lemma 1 leads to the following crucial observation:

Lemma 2

$$\text{minimum } \{t, \bar{z}^*(\underline{u}^{\dagger}(t))\} \leq z^* \quad \text{for any real } t$$

This lemma states that the above minimum is a lower bound to the original IP.

Proof

If $t > z^*$, then by Lemma 1, $u_b^{\dagger}(t) > v_b^*$ $\forall$ $b \varepsilon \underset{\sim}{A}$. This implies that the optimal solution of P is contained in $\bar{P}(\underline{u}^{\dagger}(t))$. Consequently, the optimal objective function value of $\bar{P}(\underline{u}^{\dagger}(t))$ is a lower bound to P. In other words, $\bar{z}^*(\underline{u}^{\dagger}(t)) \leq z^*$.

If, on the other hand, $t \leq z^*$, then t is obviously a lower bound to z^*. This completes the proof.

Let

$$L(t) = \text{minimum } \{t, \bar{z}^*(\underline{u}^{\dagger}(t))\} \tag{13}$$

A typical shape of L(t) is a function of t depicted in Figure 6. The function $\bar{z}^*(\underline{u}^{\dagger}(t))$ is nonincreasing in t. To see this note that as t increases, $u_b^{\dagger}(t)$ also increases for each $b \varepsilon \underset{\sim}{A}$ as shown in Figure 3. Larger values of $u_b^{\dagger}(t)$ result in a larger feasible region for $\bar{P}(\underline{u}^{\dagger}(t))$ and a (possibly) lower optimal value for the objective function.

Observe that the parameters $\{u_b(t)\}$ obtained from algorithm G_1 are not intended to capacitate the problem. Indeed, if $t < z^*$, than the value of $u_b(t)$ is not even guaranteed to be greater than or equal to v_b^*. The purpose of these "improved capacity parameters" is to obtain a tighter lower bound to z^*. This procedure is described in the next section.

3. LOWER AND UPPER BOUND PROCEDURES

This section presents two procedures. First, it shows how the concepts developed in Section 2 can be applied to obtain a lower bound to z^*, the optimal objective function value of P. Second, this section describes a heuristic procedure for obtaining an upper bound to z^* based on repetitive solutions of the relaxed problem, $\overline{P}$.

- Lower Bound

Let L(t) denote a lower bound to z^* given a target value t, as defined in eq. (13). The procedure for determining L(t) is based on the capacity improvement procedure developed in Section 2. That procedure, in turn, is based on the solution to $\overline{P}(\underline{u})$ and thus the first step is to obtain this solution. Algorithm G_1 is then applied separately for each link b in the network (i.e. $\forall\ b \varepsilon \underset{\sim}{A}$) to obtain a set of improved capacity parameters. The vector of improved capacity parameters $\underline{u}^{\dagger} = (\ldots, u_b^{\dagger}, \ldots)$ is then used to solve the LP relaxation over again (as the new program $\overline{P}(\underline{u}^{\dagger})$). This increases $\overline{z}^*$ which is a lower bound to z^*. In addition, the increased value of $\overline{z}^*$ means that a new set of improved link capacity parameters can be obtained by again using algorithm G_1 for each link $b \varepsilon \underset{\sim}{A}$. At the end of each iteration, $u_b\ \forall\ b \varepsilon \underset{\sim}{A}$ is set equal to the minimum of $u_b^{\dagger}$ and the value of u_b in the preceeding iteration. This guarantees that the capacity parameters "move-in" with each iteration. The iterative process of solving $\overline{P}$ and calling algorithm G_1 for each $b \varepsilon \underset{\sim}{A}$ is continued until either (i) $\overline{z}^*$ equals t, or (ii) the value of $\overline{z}^*$ levels off so that further iterations are unproductive.

The procedure for calculating L(t), referred to as the capacity improvement algorithm, can be summarized as follows:

```
Algorithm $G_2$:
    Inputs:   Target value $t$
              Improvement criterion $\psi$
    Outputs:  Lower bound $L(t)$
    Step 0:   Preliminaries
        Set $L^o(t) \leftarrow 0$, and set $k \leftarrow 1$
    Step 1:   Lower Bound
        (a)  Use shortest path algorithm to solve $\overline{P}(\underline{u})$ and obtain $\overline{z}^*(\underline{u})$
        (b)  Set $L^k(t) \leftarrow \min \{t, \overline{z}^*(\underline{u})\}$
    Step 2:   Stopping Rule
        (a)  If $L^k(t) \geqq t$
             then go to step 4
        (b)  If $\frac{L^k(t) - L^{k-1}(t)}{L^{k-1}(t)} < \psi$
             then go to step 4
    Step 3:   Update
        (a)  Set $k \leftarrow k+1$
        (b)  Use capacity parameter algorithm $G_1$ to determine $u_b^\dagger$ for all $b \varepsilon \underset{\sim}{A}$
        (c)  Set $u_b \leftarrow \min \{u_b, u_b^\dagger\}$ for all $b \varepsilon \underset{\sim}{A}$
        (d)  Go to step 1
    Step 4:   Termination
             Set $L(t) \leftarrow L^k(t)$
```

The determination of the input parameters is relevant with respect to the implicit enumeration procedure and so discussion of these values is deferred to Section 4. Each iteration of the procedure includes a solution of $\overline{P}$ (in step 1) and an application of algorithm G_1 for each link in the network (in step 3). The superscript k in the description of algorithm G_2 is the iteration counter, and $L^k(t)$ is the lower bound at the k-th iteration. The determination of $L^k(t)$ in step 1b follows directly from eq. (13) and Lemma 2 guarantees that this is a valid lower bound; i.e., $L^k(t) \leq z^*$ for each k. The procedure must eventually satisfy one of the two stopping criteria given in Step 2 since for a fixed target value t,

$$L^k(t) \geq L^{k-1}(t) \qquad \text{for each } k \tag{14}$$

In other words, $L^k(t)$ is nondecreasing in k. [To show that inequality (14) is true, observe that the minimization in step 3c causes each element in $\underline{u}$ to be nonincreasing in k. This implies that $\overline{z}^*(\underline{u})$ is nondecreasing in k and (14) follows from that.] Upon termination of the algorithm, L(t) is set equal to the current value of $L^k(t)$.

The performance of this capacity improvement procedure is strongly influenced by choice of the target value t. Figures 7a and 7b illustrate the typical behavior of algorithm G_2 for a small and a large value of t, respectively. The figures depict $L^k(t)$ versus the iteration counter k. In both cases the initial value, $L^1(t)$, equals the optimal objective function value of the original LP relaxation, $\overline{P}$. Figure 7a illustrates the consequences of using a small value of t--$L^k(t)$ reaches the target value, t, within a few interations. In this case, stopping rule (a) in step 2 is met, and the lower bound is L(t) = t. Figure 7b illustrates the consequences of using a high value of t--$L^k(t)$ tends toward a limit which is strictly lower than the target value. In this case, stopping rule (b) is met, and the lower bound is L(t) < t.

The two cases described above indicate that there is a critical target value, $\hat{t}$, above which algorithm G_2 will generate a lower bound which is lower (looser) than t. i.e.,

$$\hat{t} = \text{maximum } \{t : L(t) = t\} \tag{15}$$

(For the purposes of definition (15), it is assumed that the improvement criterion is $\psi = 0$.) The determination of the critical value $\hat{t}$ is explored further in Section 5, which discusses some computational results.

If a target value which is greater than $\hat{t}$ is desired, the network must be partitioned and the lower bound procedure applied to each partition. This is the idea behind the implicit enumeration procedure presented in Section 4. The implicit enumeration procedure also requires the determination of an incumbent solution for P, the objective function value of which is an upper bound to z*. This is described next.

- Upper Bound

The rationale for the heuristic procedure presented here is that since program $\overline{P}$ is easy to solve, it can be used repeatedly in determining a feasible solution to P; the objective function value of this heuristic will naturally be an upper bound to z*. Let H denote this upper bound.

At each iteration of the heuristic procedure, program $\overline{P}$ is solved and the links with large flows are selected to be a part of the network design. For the next iteration, the fixed cost on these links is assumed to be a sunk investment, and only the variable cost is used in solving $\overline{P}$ again. This iterative process of solving $\overline{P}$ and selecting additional links for inclusion in the network design is repeated until a feasible solution is reached. At this point, the links included in the network form a path between each OD pair $m \in \underset{\sim}{M}$. All links with zero flow can now be discarded, and the procedure terminates.

The performance of this heuristic procedure can be enhanced by systematically altering the value of the link capacity parameters $\{u_a\}$. Prior to solving $\overline{P}$, the current value of u_a (for each link not yet selected) is set to a convex combination of its previous value and $\overline{v}_a^*$, the flow on link a in the previous solution to $\overline{P}$. This will tend to accelerate the algorithm driving links with low flow in one iteration to even lower flow in the next one (and eventually to zero flow).

To describe the algorithm formally, let $\underset{\sim}{\hat{Y}}$ denote a (temporary) constraint set with elements of the form $\{y_a = 1\}$, and let $A_{\underset{\sim}{\hat{Y}}}$ denote the set of links in

$\underset{\sim}{A}$ for which $y_a = 1$. At each iteration of the heuristic procedure $A_{\underset{\sim}{Y}}$ denotes the set of links currently included in the network design. Let $\hat{\underline{u}} = (\ldots, \hat{u}_a, \ldots)$ denote a (temporary) link capacity vector, and let $\underline{r} = (\ldots, r_a, \ldots)$ be a link threshold vector. The algorithm chooses to include link a if its flow is greater than r_a at a given iteration.

Let $\overline{P}_{\hat{\underset{\sim}{Y}}}(\hat{\underline{u}})$ denote the linear program $\overline{P}$ augmented with constraint set $\hat{\underset{\sim}{Y}}$ and using capacity parameter vector $\hat{\underline{u}}$. Including $\hat{\underset{\sim}{Y}}$ in the problem does not actually enlarge the number of constraints in $\overline{P}$. Rather, it alters the form of the objective function. Let $F_{\hat{\underset{\sim}{Y}}}$ denote the sunk investment cost associated with $\hat{\underset{\sim}{Y}}$, i.e.

$$F_Y = \sum_{a \in A_{\hat{\underset{\sim}{Y}}}} f_a y_a$$

and define $\hat{f}_a$ ∀ a as

$$\hat{f}_a = \begin{cases} 0 & \forall \quad a \in A_{\hat{\underset{\sim}{Y}}} \\ f_a & \forall \quad a \notin A_{\hat{\underset{\sim}{Y}}} \end{cases}$$

The LP can now be written as follows:

Program $\overline{P}_Y(\hat{\underline{u}})$:

$$\min_{\underline{x} \in \underset{\sim}{X}} \; z_{\hat{\underset{\sim}{Y}}}(\hat{\underline{u}}) = F_{\hat{\underset{\sim}{Y}}} + \sum_m \sum_a q_m \left(c_a + \frac{\hat{f}_a}{\hat{u}_a}\right) x_{a,m} \qquad (16)$$

Note that for any $\hat{\underset{\sim}{Y}}$ and $\hat{\underline{u}}$ this is still a shortest path problem (with flow assignment).

The heuristic upper bound procedure, referred to as algorithm G_3, can now be summarized as follows:

Algorithm G_3

Input: Reduction rule for vector $\underline{r}$

Output: Upper bound H

Heuristic network design

```
Steo 0: Prelinimaries
        Set $\underset{\sim}{\hat{Y}} \leftarrow \emptyset$, set $A_{\underset{\sim}{\hat{Y}}} \leftarrow \emptyset$, and set $\hat{\underline{u}} \leftarrow \underline{u}$
Step 1: Solve LP
        (a) Compute $F_{\underset{\sim}{\hat{Y}}}$and $\hat{f}_a$ for all a
        (b) Use shortest path algorithm to solve $\bar{P}_{\underset{\sim}{\hat{Y}}}(\hat{\underline{u}})$ to obtain link
            flow vector $\hat{\underline{v}} = (\ldots, \hat{v}_a, \ldots)$ and objective function value $\hat{z}$.
Step 2: Stopping Rule
        If $\hat{v}_a = 0 \;\; \forall\, a \notin A_{\underset{\sim}{\hat{Y}}}$
        then go to step 4
Step 3: Update
        (a) For each $a \notin A_{\underset{\sim}{\hat{Y}}}$  if  $\hat{v}_a > r_a$
            then add $\{y_a = 1\}$ to $\underset{\sim}{\hat{Y}}$ and add link a to $A_{\underset{\sim}{\hat{Y}}}$
        (b) For each $a \notin A_{\underset{\sim}{\hat{Y}}}$
            reduce $r_a$ and reset $\hat{u}_a$ to a convex combination of $\hat{u}_a$ and $\hat{v}_a$
        (c) Go to Step 1
Step 4: Termination
        (a) If $\hat{v}_a = 0$ for some $a \in A_{\underset{\sim}{\hat{Y}}}$, then set $\hat{z} \leftarrow \hat{z} - f_a$
        (b) Set $H \leftarrow \hat{z}$, and STOP.  The current set of links in $A_{\underset{\sim}{\hat{Y}}}$ is the
            heuristic network design.
```

The next section describes the implicit enumeration procedure which incorporates the aforementioned upper and lower bound algorithms.

4. IMPLICIT ENUMERATION PROCEDURE

This section looks at determining an ε-optimal solution to the integer program P. It outlines a procedure for obtaining a feasible solution to P whose objective function value is always within $100 \cdot \varepsilon$ percent of z^*. The value of ε is assumed to be specified a priori.

The implicit enumeration (IE) procedure presented in this section is based on the usual branch and bound (B&B) framework (see, for example, Geoffrion and Marsten (1972)). This framework can be represented by a binary B&B tree structure. The root node of the tree represents the original LP relaxation, $\overline{P}$, and the other B&B points represent a partition of $\overline{P}$ in which a subset of the $\{y_a\}$ decision variables are set to either 0 or 1. At each point in the tree a candidate integer solution and a lower bound are obtained.

The principle distinction between the standard IE procedure and the one in this paper is that here the lower bound at each point in the B&B tree is not obtained directly from the LP relaxation of the problem. Instead, the lower bound is based on the capacity improvement procedure described in Section 2. This results in tighter lower bounds and a reduction in the computation effort required for the IE procedure.

The notation used to describe the IE is an extension of that used in the upper bound procedure presented in Section 3. Let $\underset{\sim}{Y}$ denote a constraint set with elements of the form $\{y_a = 0\}$ or $\{y_a = 1\}$ and let $A_{\underset{\sim}{Y}}$ denote the set of links in $\underset{\sim}{A}$ whose decision variable is fixed in set $\underset{\sim}{Y}$ (i.e., if $a \varepsilon A_{\underset{\sim}{Y}}$ then either $y_a = 0$ or $y_a = 1$). Let $F_{\underset{\sim}{Y}}$ denote the sunk investment cost associated with the links whose decision variable is set to 1 in $\underset{\sim}{Y}$. That is,

$$F_{\underset{\sim}{Y}} = \sum_{a \varepsilon A_{\underset{\sim}{Y}}} f_a y_a \qquad (17)$$

Also, the modified fixed charges are defined as

$$\tilde{f}_a = \begin{cases} K(1-y_a) & \forall \; a \varepsilon \underset{\sim}{A}_{\underset{\sim}{Y}} \\ f_a & \forall \; a \not\varepsilon \underset{\sim}{A}_{\underset{\sim}{Y}} \end{cases} \tag{18}$$

where K is a large positive constant. The effect of eq.(18) is that links whose decision variable have been set to 1 in $\underset{\sim}{Y}$ incur zero fixed charge while links whose decision variable have been set to 0 incur a large (effectively infinite) fixed charge. The fixed charge remains unchanged for links whose decision variable is not set in $\underset{\sim}{Y}$.

Using this notation, any point in the B&B tree is characterized by a specific set $\underset{\sim}{Y}$. Let $P_{\underset{\sim}{Y}}(\underline{u})$ denote the LP relaxation evaluated at $\underset{\sim}{Y}$. This program can be summarized as follows:

Program $\overline{P}_{\underset{\sim}{Y}}(\underline{u})$:

$$\min_{\underset{\sim}{\underline{x}\varepsilon X}} \; z_{\underset{\sim}{Y}}(\underline{u}) = F_{\underset{\sim}{Y}} + \sum_m \sum_a q_m \left(c_a + \frac{\tilde{f}_a}{u_a}\right) x_{a,m} \tag{17}$$

Observe that for any $\underset{\sim}{Y}$, $\overline{P}_{\underset{\sim}{Y}}(\underline{u})$ remains a shortest path problem with flow assignment.

The basic step in the IE procedure consists of determines a lower bound $L_{\underset{\sim}{Y}}(t)$ and an upper bound $H_{\underset{\sim}{Y}}$ for any given $\underset{\sim}{Y}$ in the B&B tree. $L_{\underset{\sim}{Y}}(t)$ is obtained by applying the capacity improvement procedure (algorithm G_2) to program $\overline{P}_{\underset{\sim}{Y}}(\underline{u})$ (rather than to program $\overline{P}(\underline{u})$). Similarly, $H_{\underset{\sim}{Y}}$ is obtained by using $\overline{P}_{\underset{\sim}{Y}}(\underline{u})$ in the heuristic procedure (algorithm G_3) which can be solved for the given $\underset{\sim}{Y}$ in the B&B tree. Let H denote the best (minimum) value of $H_{\underset{\sim}{Y}}$ for all B&B points $\underset{\sim}{Y}$ that have been evaluated up to and including the current $\underset{\sim}{Y}$.

[The heuristic network design associated with this upper bound is retained as the incumbent solution to P.] To ensure that the final upper bound value is within $100\cdot\varepsilon$ of z^*, the target value, t, is updated at every point of the B&B tree to be

$$t = \frac{H}{1+\varepsilon} \tag{19}$$

This target value is used to obtain the abovementioned lower bound, $L_{\underset{\sim}{Y}}(t)$. If $L_{\underset{\sim}{Y}}(t)<t$, the current solution cannot be ascertained to be $100\cdot\varepsilon$ percent of z^*. The current problem must then be partitioned by adding another constraint to $\underset{\sim}{Y}$. This generates a new point in the B&B tree which is then evaluated by the procedure described above. On the other hand, if $L_{\underset{\sim}{Y}}(t)\geq t$, then the current problem is fathomed. That is, any further partitions at this point in the tree will result in lower bounds at least as high as t. Therefore, the IE procedure backtracks over the B&B tree to determine the next point $\underset{\sim}{Y}$ to be evaluated. The IE procedure continues until all B&B points have been fathomed.

The following paragraphs describe the partitioning and backtracking procedures in more detail.

- Partitioning

The purpose of the partitioning procedure is to select a link d ($d\varepsilon\underset{\sim}{A}-\underset{\sim}{A}_{\underset{\sim}{Y}}$) which is used to partition the current problem into two parts -- one with $y_d = 0$ and the other with $y_d = 1$. The criterion for choosing the partitioning link d is that the lower bound obtained for $\underset{\sim}{Y}+\{y_d=0\}$ and $Y+\{y_d=1\}$ should be as large as possible. [Remember that lower bounds are obtained from the capacity improvement procedure rather than the direct application of an LP. Thus, methods for selecting a partitioning variable based on the LP solution -- such as "up and down penalty" methods (see for example, Driebeek (1966), Tomlin (1971))-- do not necessarily identify the best partition.]

The partitioning procedure used here is based on the link flows $\{\bar{v}_a{}^*\}$ in the current optimal solution to $\bar{P}_{\underset{\sim}{Y}}(\underline{u})$. If $\bar{v}_a{}^*$ is much larger than zero, then partitioning on link a will cause a large increase in the lower bound for $\underset{\sim}{Y} + \{y_a=0\}$. Similarly, if $\bar{v}_b{}^*$ is much smaller than u_b, then partitioning on link b will cause a large increase in the lower bound for $Y + \{y_b=1\}$. Therefore, in order to increase the lower bound for both partitions, the partitioning link, d, should have a flow $\bar{v}_d{}^*$ that is, in some sense, "in the middle" between 0 and u_d.

To state the partition selection procedure, define a link partitioning measure, $E_a(\theta)$, as follows:

$$E_a(\theta) = \theta\bar{v}_a{}^* + (1-\theta)(u_a-\bar{v}_a{}^*) \qquad \forall a\epsilon\underset{\sim}{A}-\underset{\sim}{A}_{\underset{\sim}{Y}}$$

The parameter θ is a predetermined weighting constant in the range $0\leq\theta\leq1$. The set $\{E_a(\theta)\}$ is defined over the currently active links (i.e., those not set in $\underset{\sim}{Y}$). For $\theta = 1$, $\{E_a(\theta)\}$ represents the set of link flows whereas, for $\theta=0$, $\{E_a(\theta)\}$ represents the set of "unused link capacities." The partitioning rule is to select a link, d, such that

$$E_d(\theta) = \max_{a\epsilon\underset{\sim}{A}-\underset{\sim}{A}_{\underset{\sim}{Y}}} \{E_a(\theta)\}$$

Ties are broken arbitrarily. The choice of θ was determined empirically. A value of $\theta = 0.90$ seems to perform well.

- <u>Backtracking</u>

The purpose of the backtracking procedure is to determine the next point in the B&B tree to evaluate once the current point has been fathomed. The method used here is to update the constraint set $\underset{\sim}{Y}$ using a "depth first" search.

In the computer, the set $\underset{\sim}{Y}$ is represented as a stack (i.e., a LIFO data structure). Constraints are added to $\underset{\sim}{Y}$ when partitioning occurs. By convention, when the current problem is partitioned, the constraint $\{y_a=0\}$ is added to stack $\underset{\sim}{Y}$ and the constraint $\{y_a=1\}$ is added to a temporary stack $\underset{\sim}{T}$. When the current problem is fathomed, the B&B tree is backtracked by deleting constraints from $\underset{\sim}{Y}$ (in LIFO order) until either (i) a constraint of the form $\{y_a=0\}$ is encountered and deleted or (ii) $\underset{\sim}{Y}=\emptyset$. As long as $\underset{\sim}{T}$ is non-empty, there are points in the B&B tree yet to be evaluated. In this case, a single constraint is transferred from $\underset{\sim}{T}$ to $\underset{\sim}{Y}$. This updated set, $\underset{\sim}{Y}$, represents the next point in the B&B tree to be evaluated. On the other hand, if $\underset{\sim}{T}$ is empty, then all B&B points have been fathomed and the IE procedure terminates.

In addition to updating $\underset{\sim}{Y}$, the appropriate link capacity vector $\underline{u}$ must be determined. In general, if a set of link capacity parameters $\{u_a\}$ is determined at a point in the B&B tree, these parameters are also valid at any decendent point in the tree. Thus, no special handling of the vector $\underline{u}$ is required in the partitioning process. During backtracking, however, the appropriate link capacity vector will not be available unless it has been previously stored. Accordingly, along with the temporary constraint stack $\underset{\sim}{T}$, a temporary vector stack $\underset{\sim}{U}$ is used. When a constraint is added to $\underset{\sim}{T}$, the current link capacity vector, $\underline{u}$, is added to $\underset{\sim}{U}$; when a constrained is deleted from $\underset{\sim}{T}$, a vector is deleted from $\underset{\sim}{U}$. This deleted vector becomes the current value for $\underline{u}$. [If computer storage is limited, a subset of $\underset{\sim}{U}$ can be stored and the appropriate link capacity vector can be determined by a look-up table. This reduces storage requirements at the expense of computational effort.]

- Algorithm

The implicit enumeration procedure, referred to as algorithm G_4, can be summarized as follows.

```
Algorithm $G_4$

    Inputs:   Optimality criterion, $\varepsilon$
              Lower bound improvement criterion, $\psi$
              Partition weighting constant, $\theta$

    Outputs: (i) Heuristic network design whose objective function value, H,
                 is within $100 \cdot \varepsilon$ percent of optimality, or
            (ii) Determination that there is no feasible network design.

    Step 0:   Preliminaries
              (a) Set $\underline{u}$ to vector of maximum possible link flows
              (b) Set $\underset{\sim}{Y} \leftarrow \emptyset$, set $\underset{\sim}{T} \leftarrow \emptyset$, and set $\underset{\sim}{U} \leftarrow \emptyset$
              (c) Set $H \leftarrow \infty$

    Step 1:   Evaluate Bounds
              (a) Use heuristic algorithm $G_3$ to determine $H_{\underset{\sim}{Y}}$
              (b) If $H_{\underset{\sim}{Y}} < H$
                  then set $H \leftarrow H_{\underset{\sim}{Y}}$; save heuristic solution as incumbent network
                  design; set $t \leftarrow H/(1+\varepsilon)$
              (c) Use capacity improvement algorithm $G_2$ to determine $L_{\underset{\sim}{Y}}(t)$

    Step 2:   Stopping Rule
              (a) If $H = \infty$
                  then STOP.  Program P is infessible.
              (b) If $L_{\underset{\sim}{Y}}(t) \geq t$ and $T = \emptyset$
                  then STOP.  H is within $\varepsilon$ of optimality.
```

```
        (c) If $L_{\underset{\sim}{Y}}(t)<t$
            then go to Step 3.
        (d) Else go to Step 4.
Step 3: Partition
        (a) Set $\hat{\underset{\sim}{A}}_{\underset{\sim}{Y}} \leftarrow \{a:\ a\varepsilon\underset{\sim}{A}-\underset{\sim}{A}_{\underset{\sim}{Y}},\quad 0<\bar{v}^{*}_{a}<u_{a}\}$
        (b) Compute $S_{a}\leftarrow\Theta\bar{v}_{a}{}^{*}+(1-\Theta)\ (u_{a}-\bar{v}^{*}_{a})\quad\forall a\varepsilon\hat{\underset{\sim}{A}}_{\underset{\sim}{Y}}$
        (c) Set $S^{max}\leftarrow\max_{a\varepsilon\hat{\underset{\sim}{A}}_{\underset{\sim}{Y}}}\{S_{a}\}$
        (d) Set $d\leftarrow\{a:S_{a}=S^{max}\}$    Break ties arbritrarily.
        (e) Add $\{y_{d}=0\}$ to $\underset{\sim}{Y}$
            Add $\{y_{d}=1\}$ to $\underset{\sim}{T}$
            Add $\underline{u}$ to $\underset{\sim}{U}$
        (f) Go to Step 1.
Step 4: Backtrack
        (a) Delete constraints from $\underset{\sim}{Y}$ (in LIFO order) until either a
            constraint of the form $y_{a}=1$ is encountered or $\underset{\sim}{Y}=\emptyset$.
        (b) Delete a single constraint from $\underset{\sim}{T}$ (in LIFO order) and add
            it to $\underset{\sim}{Y}$.
        (c) Delete a single vector $\hat{\underline{u}}$ from $\underset{\sim}{U}$ (in LIFO order) and set $u\leftarrow\hat{\underline{u}}$.
        (d) Go to Step 1.
```

There are three input parameters to the algorithm. First, the optimality criterion, ε, must be determined a priori. Consequently, in cases where there is no natural tolerance (due, for example, to data accuracy), the IE procedure may have to be solved several times starting with a large value of ε. This value is then decreased until the computational effort needed to obtain a solution outweighs the incremental value of such a solution. Second, the improvement criterion, ψ, provides trade-off between the work done at a single point in the B&B tree and the number of points that must be evaluated. The smaller the value of ψ, the greater the number of iterations of the capacity improvement algorithm and the higher the lower bound for each subproblem. As mentioned in Section 3, however, the computational effort here exhibits decreasing marginal returns and branching is eventually appropriate. Empirical results indicate that ψ should be in the range of 0.003 to 0.001. The third input is the weighting constant, θ, described earlier in this section. As mentioned there, a value of $\theta = 0.90$ seems to perform well.

Step 1 of the algorithm determines lower and upper bounds for the current subproblem to P. The first subproblem, at the route node of the B&B tree, is the LP relexation of P. Here, the threshold vector, $\underline{r}$, in algorithm G_3 is reduced gradually in order to obtain a good initial heuristic solution. Once the B&B root node has been evaluated, however, numerical experimentation showed that the best choice of the threshold vector for all subsequent subproblems is $\underline{r}=\underline{0}$. This amounts to a simple "rounding-up" of all fractional y_a decision variables in program $\overline{P}_{\underset{\sim}{Y}}(\underline{u})$. In fact, since $\overline{P}_{\underset{\sim}{Y}}(\underline{u})$ must be solved repeatedly in the link capacity improvement procedure, this "rounding-up" step can be performed within algorithm G_2 with no additional computational effort.

The IE procedure continues until one of the two stopping rules in step 2 is met. Step 2a terminates the problem if there is no feasible solution and step 2b terminates the procedure at the ε-optimal solution. Step 2c is used to invoke partitioning in cases where no fathoming occured at the current B&B point and step 2d initiates backtracking in case the current B&B point is fathomed. Partitioning and backtracking are performed in steps 3 and 4, respectively.

The next section evaluates the computational performance of this implicit enumeration framework.

5. COMPUTATIONAL EXPERIENCE

This section describes the computational performance of the capacity improvement (CI) and implicit enumeration (IE) procedures presented in this paper. For each procedure, optimization tests were carried out on networks of various sizes and with several different levels of OD shipment flows.

The network and OD flow generation procedures are summarized in Appendix B. Each network included a set of city (end-of-line) terminals, $\underset{\sim}{N}_E$, and a set of transshipment (breakbulk) terminals, $\underset{\sim}{N}_B$. Terminals in $\underset{\sim}{N}_E$ were used purely as origin/destination points; terminals in $\underset{\sim}{N}_B$ were used purely at transshipment points. All networks were complete with a fixed charge and a variable cost component on each (directed) arc. The fixed charge for a link was set equal to the cost of carrying one truckload of freight over that link. In addition, special arcs with only variable costs were used to represent handling costs at the transshipment terminals. The size of the networks ranged from $|\underset{\sim}{N}_E|=10$, $|\underset{\sim}{N}_B|=2$ (involving 90 OD pairs(markets)and 132 fixed charge links) to $|\underset{\sim}{N}_E|=40$, $|\underset{\sim}{N}_B|=6$ (involving 1560 OD pairs and 2070 fixed charge links). To capture the effect of flow level, in addition to network size, the largest of the networks was tested using several OD flow levels. These runs resulted in an average non-zero link flow in the heuristic network design ranging from 1/3 of a truckload to 8 truckloads (compared to a fixed charge equivalent of 1 truckload).

Two topics are dealt with separately in this section. First, the lower bounds produced by the CI procedure are compared to these generated by the LP relaxation of program P. Second, the computational effort versus accuracy of the IE procedure is evaluated. As a means of comparison, the computational performance of a standard implicit enumeration (i.e., without incorporating the CI procedure) and a dual ascent method are also explored.

- Capacity Improvement Procedure

The following paragraphs describe the ability of the capacity improvement procedure to improve the lower bound to z^*, the optimal objective function value to program P. For each network under consideration, an upper bound H (using heuristic algorithm G_3) and a lower bound L(t) (using CI algorithm G_2) were determined. The near-optimality of these bounds is measured by

$$\varepsilon(t) = \frac{H-L(t)}{L(t)} \tag{20}$$

To measure the effect of the CI procedure, two different target values, t, were used. In the first case, t was set to t = $\bar{z}^*$ where $\bar{z}^*$ is the optimal objective function value of program $\overline{P}$. This causes L($\bar{z}^*$) = $\bar{z}^*$ and so ε($\bar{z}^*$) is simply the near-optimality measure of the original LP relaxation. In the second case, t was set to $\hat{t}$, the critical target value defined in eq. (15). Thus, in this case, L($\hat{t}$) is the highest lower bound and ε($\hat{t}$) is the tightest near-optimality measure attainable with the CI procedure.

Table 1 displays the near-optimality measure, ε(t), of the LP relaxation (i.e. for t= $\bar{z}^*$ in eq.(20)) and the CI procedure (i.e. for t=$\hat{t}$ in eq. (20)) for four trial networks of various sizes. As shown in the table, in all cases, the CI procedure improved the lower bounds. The optimality measure is reduced by anywhere from 60% for the smallest network to 15% for the largest one.

Note that increasing the number of OD markets increases the overall flow in the network. In each of the four networks reported in Table 1, the size of the link fixed charge was equivalent to the cost of carrying one truckload.

The average non-zero link flow in the heuristic solution, however, varied from 2 truckloads in the smallest networks to over 4 truckloads in the largest networks. For the cases in which the flow level is relatively high, the LP link costs closely approximate the fixed-plus-linear link costs of the IP and thus the simple LP low bound is not far from z*. Moreover, the improved capacity parameters u_b+ must always be greater than or equal to the link flows in the LP relaxation (see algorithm G_1). Thus, in the high flow level case, the CI procedure has a limited ability to reduce the link capacity parameters. This inability combined with the closeness of the LP lower bound to z* means that the CI procedure will not improve the lower bound substantially. On the other hand, for the cases where the volume of flow is relatively low, the lower bounds generated by the CI procedure are significantly better than those obtained from the LP relaxation. This explains why the effect of the CI procedure diminishes as the size of the network increases in Table 1.

To test the effects of flow level directly, the largest network ($|N_E|=40$, $|N_B|=6$) was optimized with three different levels of flow: low, medium and high. [The four networks exhibited in Table 1 fall into the medium flow category.] The volume of flow in the network can be characterized by the link flow in the solution of the LP relaxation. For the low, medium and high flow cases, the average non-zero link flow in the heuristic solution was 1/3, 5, and 8 truckloads, respectively. The fixed charge in all cases was equivalent to the cost of carrying 1 truckload. The results of this test are shown in Table 2. (Note that the figures for the medium case in Table 2 are not identical to the largest network in Table 1 since the networks in each case were generated separately.) As this table shows, the CI procedure yields substantial improvements in the low volume but only marginal benefits in the medium and high volume cases.

The conclusion, then,from Tables 1 and 2 is that the relative flow level is more important than the network size in estimating the ability of the CI procedure to improve upon the lower bound generated by the LP relaxation.

Any evaluation of the tightness of bounds must be compared with the computational cost required to obtain these bounds. One measure of the computational effort of the CI procedure is the number of iterations of algorithm G_2 required to obtain a given bound. This relationship is brought out in Figure 8, which describes the low volume network shown in Table 2. For an optimality measure, ε, greater than or equal to $\varepsilon(\bar{z}^*)$ (i.e., for $t \leq \bar{z}^*$), the initial LP relaxation provides a sufficiently tight lower bound and so the algorithm stops in the first iteration. For all values of ε between $\varepsilon(\hat{t})$ and $\varepsilon(\bar{z}^*)$ (i.e. $\bar{z}^*<t<\hat{t}$), the lower bound $L(t)$ reaches its target value, t. But, as shown in the figure, as ε approaches $\varepsilon(\hat{t})$ the convergence of the algorithm shows a diminishing rate of return. Thus for values of ε near (or below) $\varepsilon(\hat{t})$, it is more efficient to incorporate the CI procedure within an implicit enumeration framework, as in algorithm G_4.

The computational aspects of the IE procedure are discussed next.

- Comparison of Performance Curves

To assess the accuracy/computational effort tradeoffs associated with the implicit enumeration (IE) algorithm, G_4, this procedure was carried out at various levels of ε for each of the trial networks discussed above. The resulting curves are shown in Figures 9 and 10. These figures display the near-optimality measure $\varepsilon(t)$ versus the computational effort (in CPU seconds on a Digital Equipment Corporation VAX 782 minicomputer).

Figures 9a through 9d correspond to the four trial networks shown in Table 1. The solid line in each figure shows the performance curve for the IE procedure described in this paper. The computational cost associated with the original LP relaxation (at $\varepsilon(\bar{z}^*)$) and the critical target value (at $\varepsilon(t)$) for the CI procedure are shown on this line. As expected from the previous discussion, only small improvements are achievable in the largest networks whereas substantial gains are obtained in the smaller networks. In fact, for the smallest network (Figure 9a),optimality was achieved. The majority of the improvement in the optimality measure was obtained at the root node in the B&B tree (i.e. ε in the range $\varepsilon(\hat{t}) \leq \varepsilon \leq \varepsilon(z^*)$). Thereafter, while additional improvements in ε were obtained, these improvements were usually achieved at a high computational cost.

The IE procedure presented in this paper can be compared to a standard implicit enumeration (SIE) procedure where the lower bound at each point in the B&B tree is simply the LP relaxation of the corresponding subproblem. The performance curves for the SIE are shown as dashed lines in the figures. These curves point out the benefit obtained by incorporating the CI procedure within the IE. In all cases, the IE dominates the SIE. Moreover, for the larger networks, while only small gains were achieved by the IE, this compares to essentially no improvement over the original LP relaxation for the SIE.

The computational quality of the IE procedure may also be guaged by comparing it to the performance of the dual ascent (DA) method proposed by Magnanti and Wong (1984b). The DA procedure (which is summarized in Appendix C) solves the dual to the disaggregate LP relaxation composed of objective function (3a) with constraints (2), (4), and (5). The performance curves of the DA procedure are shown as dotted lines in Figures 9a-9d. For the small networks, the IE procedure alone (at the point $\varepsilon(\hat{t})$) achieves a tighter lower than the DA procedure, and with significantly less computational effort. For the larger networks, the IE procedure was better for larger tolerances whereas the DA procedure was superior for tighter tolerances. This was due, in large part, to the fact that the larger networks had a greater level of flow and so the CI procedure was less able to improve the solution.

To measure the effect of flow level , the performance of the procedures was evaluating using the largest network with three different flow levels (see Table 2). The performance curves are shown in Figures 10a through 10c. [The designation of the IE, SIE, and DA curves is the same as that used in Figures 9.] As expected, the IE always outperforms the SIE, but this improvement diminishes as the flow level increases. The IE procedure is better than the DA for the low flow volume case, and for higher values of ε, in the medium volume case. In the high volume case, $\bar{z}^*$ is already close to z^* (within 4.3%) and so the bounds obtained by the IE and the DA are also close.

In reviewing Figures 9 and 10, it seems that, where optimality is not attained, the IE procedure cannot overcome certain values of ε. That is, for small tolerances, the exponential growth of the B&B tree predominates and so the IE curves track those of the SIE. This overall pattern reflects the difficult nature of the problem. The authors of the DA algorithm have suggested that their procedure could also be imbedded within a branch and bound framework. However, on the basis of the curves in these figures, it does not seem likely that significant improvements can be obtained.

Magnanti and Wong (1984b) have found that their dual ascent procedure compares favorably with other methods for uncapacitated fixed charge network design problems. Thus, it appears that the CI and IE procedures presented in this paper have strong merits, particularly for situations with relatively low levels of flow, as in LTL freight networks. Markets with LTL networks are characterized by flows which are at the low range of our experiments. The reason is that high link flows would typically indicate an opportunity for bypassing a transshipment point and part of that flow would be diverted to another link. Thus the design problem has to do with the portions of the network with low flow. In these cases, the fixed charges correspond to a minimum frequency of, say,one truckload per day and arc flow are in the range of 1/3 to 2 truckloads per day. It is in this interesting range that the CI and IE algorithms perform particularly well.

APPENDIX A

Section 2 describes the single link capacity improvement program, $\overline{Q}_b(t)$, which, together with its subprogram $\overline{P}_b(w_b)$, is used to determine the improved capacity parameter u_b^+. This appendix uses Lagrangian relaxation procedures to characterize the solution to these two programs and to interpret the steps of algorithm G_1 which is used to solve problem $\overline{Q}_b(t)$.

Consider first program $\overline{P}_b(w_b)$ given in eqs. (8). The Lagrangian relaxation of this program (with respect to constraint (8b)) is as follows:

Program $\overline{R}_b(\lambda_b, w_b)$:

$$\bar{z}_b^*(\lambda_b, w_b) = \min_{\underset{\sim}{x} \in \underset{\sim}{X}} \left\{ \sum_m q_m \left[\sum_{a \neq b} (c_a + \frac{f_a}{u_a}) x_{a,m} + (c_b + \frac{f_b}{u_b} - \lambda_b) x_{a,m} \right] + \lambda_b w_b \right\} \qquad \text{(A1)}$$

where λ_b is the multiplier associated with constraint (8b). The Lagrangian program is then

Program $\overline{S}_b(w_b)$:

$$\max_{\lambda_b \geq 0} \left\{ \bar{z}_b^*(\lambda_b, w_b) \right\} \qquad \text{(A2)}$$

Since $\overline{S}_b(w_b)$ is the Lagrangian program of $\overline{P}_b(w_b)$, both programs have tne same optimal objective function value, $\bar{z}_b^*(w_b)$.

The following paragraphs describe a method for solving $\overline{S}_b(w_b)$; i.e. to determine $\bar{\lambda}_b^* = \bar{\lambda}_b^*(w_b)$. To characterize $\bar{\lambda}_b^*$, let $\partial(\lambda_b)$ denote a subgradient (i.e., "slope") of objective function $\bar{z}_b^*(\lambda_b, w_b)$. Since $\bar{z}_b^*(\lambda_b, w_b)$ is a piecewise-linear concave function in λ_b, (see Figure A), the optimal value of λ_b will satisfy the following conditions:

$$\partial(\lambda_b) \geq 0 \quad \forall \quad \lambda_b \leq \bar{\lambda}_b^* \qquad \text{(A3a)}$$

$$\partial(\lambda_b) \leq 0 \quad \forall \quad \lambda_b \geq \lambda_b^* \qquad \text{(A3b)}$$

Also, since $\bar{z}^*_b(\lambda_b,w_b)$ is subdifferentiable (see, for example, Fisher (1981)), $\partial(\lambda_b)$ is given by

$$\partial(\lambda_b) = w_b - \sum_m q_m \bar{x}^*_{a,m} = w_b - \bar{v}^*_b(\lambda_b,w_b) \tag{A4}$$

where $\bar{v}^*_b(\lambda_b,w_b)$ is the flow on link b in the optimal solution to program $\overline{R}_b(\lambda_b,w_b)$.

The optimal flow $\bar{v}^*_b(\lambda_b,w_b)$ in program $\overline{R}_b(\lambda_b,w_b)$ can be described for any value of λ_b. Let $\{\Delta_{b,m}\}$ be the set of $|\underset{\sim}{M}|$ marginal costs defined in Section 2. It is assumed that all markets are sorted in ascending order of $\{\Delta_{b,m}\}$ and that the market index, m, is relabeled accordingly. Let m^{max} denote the largest index m such that $\Delta_{b,m}$ remains finite (i.e., $m^{max} = \max\,\{m:\Delta_{b,m} < \infty\}$). In addition, let $v_{b,m}$ denote the cumulative flow on link b for the m-th OD pair; i.e.,

$$v_{b,m} = \sum_{\ell=1}^{m} q_\ell \tag{A5}$$

Also let $v_b^{max} = v_{b,m}$ for $m = m^{max}$. [In program $\overline{P}$, u_b is initially set so that $u_b \geq v_b^{max}$.] Now, observe that as λ_b increases, the cost of link b decreases (see objective function (A1)) and so more flow will be attracted to this link. Specifically, for each m (m=1,..., m^{max}), if λ_b is in the domain

$$\Delta_{b,m-1} \leq \lambda_b \leq \Delta_{b,m} \tag{A6a}$$

then the optimal flow on link b in program $\overline{R}_b(\lambda_b,w_b)$ is

$$\bar{v}^*_b(\lambda_b,w_b) = v_{b,m-1} \tag{A6b}$$

(For purposes of definition, let $\Delta_{b,0} \equiv 0$ and $v_{b,0} \equiv 0$.)

The determination of $\overline{\lambda}^*_b$ is now straightforward. If $v_b^{max} < w_b$, then program $\overline{P}_b(w_b)$ is infeasible and the Lagrangian program $S_b(w_b)$ is unbounded (i.e., $\overline{\lambda}^*_b = \infty$).

Otherwise, define $\hat{m}$ as follows:

$$\hat{m} = \underset{m=1,\ldots,m^{max}}{\text{minimum}} \left\{ m : v_{b,m} \geq w_b \right\} \tag{A7}$$

Now, consider the subgradient $\partial(\lambda_b)$ for $\lambda_b \geq \Delta_{b,m}$. Using eqs. (A4) and (A6), the definition of $\hat{m}$ implies that $\partial(\lambda_b) \leq 0$ for $\lambda_b \geq \Delta_{b,\hat{m}}$. This definition also implies that $\partial(\lambda_b) \geq 0$ for $\lambda_b \leq \Delta_{b,\hat{m}}$. (If this were not true then $\hat{m}$ would not be the minimum in (A7).) Hence, $\Delta_{b,\hat{m}}$ satisfies conditions (A3) giving the desired result:

$$\overline{\lambda}^*{}_b(w_b) = \Delta_{b,\hat{m}} \tag{A8}$$

The identification of $\hat{m}$ in eq. (A7) can also be used to specify $\bar{z}^*_b(w_b)$, the optimal objective function value of programs $\overline{P}_b(w_b)$ and $\overline{S}_b(w_b)$. Refer again to Figure A1. For $\lambda_b = 0$, subprogram $\overline{R}_b(\lambda_b,w_b)$ is identical to the original LP relaxation $\overline{P}(\underline{u})$. Thus $\bar{z}^*_b(0,w_b) = \bar{z}^*$ as shown in the figure. For λ_b in the domain $0 \leq \lambda_b \leq \Delta_{b,\hat{m}}$, $\bar{z}^*_b(\lambda_b,w_b)$ is a piecewise-linear increasing function in λ_b. Each linear segment in this domain has a width given by (A6a) and a corresponding subgradient (i.e., "slope") given by (A6b). A typical segment is illustrated in Figure A1. Combining this information, the optimal value $\bar{z}^*_b(w_b) = \bar{z}^*_b(\overline{\lambda}^*_b,w_b)$ can be expressed as $\bar{z}^*$ plus the sum of the vertical components of the first $\hat{m}$ linear segments, i.e.,

$$\bar{z}^*_b(w_b) = \bar{z}^* + \sum_{m=1}^{\hat{m}} (w_b - v_{b,m-1})(\Delta_{b,m} - \Delta_{b,m-1}) \tag{A9}$$

where $\hat{m}$ is defined in eq. (A7). Eq. (A9) can be simplified by rearranging terms. This yields

$$\bar{z}^*_b(w_b) = \bar{z}^* + \sum_{m=1}^{\hat{m}-1} \Delta_{b,m}\, q_m + (w_b - v_{b,\hat{m}-1})\Delta_{b,\hat{m}} \tag{A10}$$

Eq. (A10) can be used to determine the optimal solution to program $\overline{Q}_b(t)$. Note that eq. (A10) holds for any w_b in domain (9). In particular, it holds for the minimum value of w_b that satisfies $\bar{z}_b^*(w_b) \geq t$ for any t in domain (11). Since $\bar{z}_b^*(w_b)$ is an increasing function in w_b, the minimum value, $\bar{w}_b^* = \bar{w}_b^*(t)$ will satisfy $\bar{z}_b^*(\bar{w}_b^*) = t$. Thus, the value of $\bar{w}_b^*$ can be expressed explicitly by substituting t for the left-hand-side in eq. (A10) and solving for w_b. This gives

$$\bar{w}_b^*(t) = v_{b,\hat{m}-1} + \frac{t - \left(\bar{z}^* + \sum_{m=1}^{\hat{m}-1} \Delta_{b,m}\cdot q_m\right)}{\Delta_{b,\hat{m}}} \tag{A11}$$

The value of $\bar{w}_b^*$ in eq. (A11) is the same as that determined in algorithm G_1, although it is expressed in a slightly different form. To bring out this equivalence, observe that the final value of w_b in algorithm G_1 is determined in Step 1a. It is

$$w_b \leftarrow w_b + \frac{t - \tilde{z}_b}{\Delta_{b,m}} \tag{A12}$$

Also, prior to executing (A12), the assignments $w_b \leftarrow w_b + q_m$ and $\tilde{z}_b \leftarrow \tilde{z}_b + \Delta_{b,m}\cdot q_m$ are executed $(\hat{m}-1)$ times in Step 2a. Thus, prior to executing (A12), the values assigned to w_b and $\tilde{z}_b$ are

$$w_b \leftarrow v_{b,\hat{m}-1}$$

$$\tilde{z}_b \leftarrow \bar{z}^* + \sum_{m=1}^{\hat{m}-1} \Delta_{b,m}\cdot q_m$$

so that assignment (A12) is equivalent to eq. (A11).

The above discussion did not explicitly specify $\hat{m}$ (since w_b was not known a priori). It is straightforward to show, however, that in executing algorithm G_1, the first m (i.e., the smallest m) that satisfies the stopping condition

$$\tilde{z}_b + \Delta_{b,m}\cdot q_m \geq t$$

in step 1a, also satisfies definition (A7). Thus, algorithm G_1 determines the optimal solution to program $\overline{Q}_b(t)$.

APPENDIX B

This appendix outlines the network and OD flow generation procedure used in the computational experience section.

For each network generated, the node set, $\underset{\sim}{N}$, is taken as a subset of a set of 50 metropolitan areas distributed throughout the continental United States. Set $\underset{\sim}{N}$ is partitioned into two sets: $\underset{\sim}{N}_E$ and $\underset{\sim}{N}_B$. Here, $\underset{\sim}{N}_E$ contains the set of end-of-line terminals where shipments originate and/or terminate and $\underset{\sim}{N}_B$ contains the set of breakbulk terminals where transshipment occurs. The nodes in $\underset{\sim}{N}_E$ are randomly selected from among the 50 cities. The nodes in $\underset{\sim}{N}_B$ are selected using a p-median heuristic (Teitz and Bart (1968)), where the $|\underset{\sim}{N}_B|$ breakbulk terminals are taken as the p medians. The medians are selected from the set of 50 cities which are not contained in $\underset{\sim}{N}_E$. [Note that, in this paper, the location of all terminals is considered fixed; it is the selection of network arcs and the flow on these links that is designed.]

The networks that are generated are fully connected; i.e.,

$$\underset{\sim}{A} = \{a: I(a)\varepsilon\underset{\sim}{N},\ J(a)\varepsilon\underset{\sim}{N},\ I(a) \neq J(a)\}$$

Each arc $a\varepsilon\underset{\sim}{A}$ is a design link with associated fixed charge f_a and variable cost c_a. In addition, the breakbulk 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 end-of-line city pairs (i.e., $|\underset{\sim}{M}| = |\underset{\sim}{N}_E|^2 - |\underset{\sim}{N}_E|$). For each $m\varepsilon\underset{\sim}{M}$, the shipment demand, q_m, is generated using a gravity-type model (Stopher and Meyburg (1971)). Specifically,

$$q_m = \alpha_0 \frac{(r_m)^{\alpha_1} \cdot (s_m)^{\alpha_2}}{(d_m)^{\alpha_3}} \tag{B1}$$

where r_m is the weekly retail sales at city O(m), s_m is the weekly retail sales at city D(m), d_m is the highway milage from O(m) to D(m), and α_0, α_1, α_2, and α_3 are fixed parameters.

The numerical quantities used in the computational experiments are set to reflect a realistic LTL shipping environment. For each link $a \varepsilon \underset{\sim}{A}$, the fixed charge, f_a, (in dollars) is taken as the cost of running a single truck per week over that link at $1.10/mile. The variable cost, c_a, (in dollars per lb.) 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 breakbulk terminal is $1 per CWT. For each $m \varepsilon \underset{\sim}{M}$, the values for r_m and s_m are taken as the gross retail sales for the associated Basic Trading Area (BTA) adjusted to 1984 dollars. The distances (d_m) are in statute miles. For all networks generated, the value of the parameters in eq. (B1) are fixed at α_1=1.0, α_2=0.5, and α_3=0.5. Note that $\alpha_1 \neq \alpha_2$ so that nonsymmetric shipping demands are modelled. Finally, the low, medium, and heavy shipment volumes are generated by setting parameter α_0 at 2.0, 100.0, and 1000.0, respectively.

APPENDIX C

This appendix reviews the dual ascent procedure for uncapacitated fixed charge network design problems proposed by Magnanti and Wong (1984b).

Let P_d denote the disaggregate integer program formed by using objective function (3a) with constraints (2), (3c), and (4). Let $\overline{P}_d$ denote the LP relaxation of P_d in which constraints (3c) are replaced with (5) and let $\overline{D}_d$ denote the dual of $\overline{P}_d$. Let $\{\mu_{n,m}\}$ and $\{\sigma_{a,m}\}$ denote the dual variants associated with constraints (2) and (4), respectively. The dual program is as follows:

Program $\overline{D}_d$:

$$\max \bar{z}_d = \sum_m \mu_{D(m),m} \tag{C1a}$$

subject to

$$\mu_{J(a),m} - \mu_{I(a),m} + \sigma_{a,m} \leq c_a q_m \;\forall\; a,m \tag{C1b}$$

$$\sum_m \sigma_{a,m} \leq f_a \quad \forall\; a \tag{C1c}$$

$$\sigma_{a,m} \geq 0 \quad \forall\; a \tag{C1d}$$

The dual ascent procedure is summarized in the following two algorithms:

Algorithm C:

```
Input:  A node j
        A market m
Output: Updated dual variables, $\mu_{j,m}$ and $\sigma_{a,m}$ $\forall\, a \in \underset{\sim}{B}_j$
Step 0: Preliminaries
        (a)  Select a link $a \in \underset{\sim}{B}_j$
        (b)  Set $i \leftarrow I(a)$, set $j \leftarrow J(a)$, and set $\sigma_{a,m} \leftarrow 0$;
```

```
Step 1: Determine Slack
        (a) If $\mu_{j,m} - \mu_{i,m} + \sigma_{a,m} < c_a q_m$
            then set $\delta_{a,m} \leftarrow c_a q_m - (\mu_{j,m} - \mu_{i,m} + \sigma_{a,m})$
        (b) Else if $\Sigma_m \sigma_{a,m} < f_a$
            then set $\delta_{a,m} \leftarrow f_a - \Sigma_m \sigma_{a,m}$
        (c) Else set $j \leftarrow i$ and use algorithm C recursively.
            Continue recursion until either positive stack is obtained,
            (i.e., $\delta_{a,m} > 0$) or root node $0(m)$ is encountered (i.e., $\delta_{a,m} = 0$).
Step 2: Update Dual Variables
        (a) Repeat steps 0 and 1 for each arc $a \varepsilon \underset{\sim}{B}_j$
        (b) Set minimum slack $\hat{\delta}_{j,m} \leftarrow \min_{a \varepsilon \underset{\sim}{B}_j} \{\delta_{a,m}\}$
        (c) If minimum slack is from step 1a
            then set $\mu_{j,m} \leftarrow \mu_{j,m} + \hat{\delta}_{j,m}$
        (d) Else if minimum slack is from step 1b
            then set $\mu_{j,m} \leftarrow \mu_{j,m} + \hat{\delta}_{j,m}$
            and set $\sigma_{a,m} \leftarrow \sigma_{a,m} + \hat{\delta}_{j,m}$ for all $a \varepsilon \underset{\sim}{B}_j$
```

Dual Ascent Algorithm:

```
Output: Objective function value, $\tilde{Z}_d$
Step 0: Preliminaries
        (a) Set $\mu_{n,m} \leftarrow 0 \;\forall\; n,m$
        (b) Set $\sigma_{a,m} \leftarrow 0 \;\forall\; a,m$
Step 1: Update Dual Variables at Destination
        For each $m \varepsilon \underset{\sim}{M}$ set $j \leftarrow D(m)$
        and use algorithm C to obtain updated $\mu_{D(m),m}$
```

Step 2: Stopping Rule

(a) If at least one $\mu_{D(m),m}$ is increased then go to Step 1

(b) Else set $\tilde{z}_d \leftarrow \sum_m \mu_{D(m),m}$ and STOP.

The dual ascent algorithm seeks to increase the value of dual variables $\{\mu_{D(m),m}\}$ iteratively for each market $m \varepsilon \underset{\sim}{M}$ while maintaining feasibility of program $\overline{D}_d$. Each iteration consists of increasing all $\mu_{D(m),m}$ by $\hat{\delta}_{D(m),m}$. The algorithm continues until $\hat{\delta}_{D(m),m} = 0 \ \forall\, m$. By construction, the dual variable values computed by the algorithm are feasible in $\overline{D}_d$. Thus, the objective function value, $\tilde{z}_d$, using these variables is a valid lower bound to z^*, the optimal objective function value of the integer programs P_d and P.

ACKNOWLEDGEMENT

This research is supported by I. U. International, Inc. through a research contract with Massachusetts Institute of Technology and Princeton University. The authors would like to kindly thank Mr. John Terry and Mr. 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

Aho, A. V., J. E. Hopcroft, and J. D. Ullman (1983). Data Structures and Algorithms, Addison-Wesley, Reading, MA.

Balinski, M. L. (1961). "Fixed-Cost Transportation Problems," Naval Research Logistics Quarterly, Vol. 8, No. 1 (March), pp. 41--54.

Barr, R. S., F. Glover, and D. Klingman (1981). "A New Optimization Method for Large Scale Fixed Charge Transportation Problems," Operations Research, Vol. 29, No. 3 (May--June), pp. 448--463.

Driebeek, N. J. (1966). "An Algorithm for the Solution of Mixed Integer Programming Problems," Management Science, Vol. 12, No. 7 (March), pp. 576--587.

Fisher, M. (1981). "The Lagrangian Relaxation Method for Solving Integer Programming Problems," Management Science, Vol. 27, No. 1 (January), pp. 1--17.

Floyd, R. W. (1962). "Algorithm 97--Shortest Path," ACM: Communications of the ACM, Vol. 5, No. 6 (May), p. 345.

Geoffrion, A. M. and R. E. Marsten (1972). "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, MA.

Glover, F., D. Karney, and D. Klingman (1974). "Implementation and Computational Comparisons of Primal, Dual, and Primal-Dual Computer Codes for Minimum Cost Network Flow Problems," Networks, Vol. 4, No. 3 (Fall), pp. 191--212.

Johnson, D. S., J. K. Lenstra, and A. H. G. Rinnooy Kan (1978). "The Complexity of the Network Design Problem." Networks, Vol. 8, No. 4 (Winter), pp. 279--285.

Magnanti, T. L., P. Mireault, and R. T. Wong (1984). "Tailoring Bender's Decomposition for Uncapacitated Network Design," Working Paper OR 127--84, Operations Research Center, M.I.T., Cambridge, MA (September).

Magnanti, T.L. and R. T. Wong (1984a). "Network Design and Transportation Planning: Models and Algorithms." Transportation Science, Vol. 18, No. 1 (February), pp. 1--55.

Magnanti, T. L. and R. T. Wong (1984b). "A Dual Ascent Approach for Network Design Problems," forthcoming.

Moore, E. (1957). "The Shortest Path Through a Maze," Proceedings of the International Symposium on the Theory of Switching, Harvard University Press, Cambridge, MA, pp. 285--292.

Powell, W. B. and Y. Sheffi (1983). "The Load Planning Problem of Motor Carriers: Problem Description and a Proposed Solution Approach," Transportation Research, Vol. 17A, No. 6 (November), pp. 471--480.

Sheffi, Y. and W. B. Powell (1985). "Interactive Optimization for Motor Carrier Load Planning," Submitted for publication.

Steenbrink, P. A. (1974). "Transport Network Optimization in the Dutch Integral Transportation Study," Transportation Research, Vol. 8, No. 1 (February), pp. 11-27.

Stopher, P. R. and A. H. Meyburg (1975). Urban Transportation Modeling and Planning, Heath and Co., Lexington, MA.

Teitz, M. B. and P. Bart (1968). "Heuristic Methods for Estimating the Generalized Vertex Median of a Weighted Graph," Operations Research, Vol. 16, No. 5 (September - October), pp. 955-961.

Tomlin, J. A. (1971). "An Improved Branch-and-Bound Method for Integer Programming," Operations Research, Vol. 19, No. 4 (July - August), pp. 1070-1075.

Wong, R. T. (1980). "Worst-Case Analysis of Network Design Problem Heuristics," SIAM: Journal of Algebraic and Discrete Methods, Vol. 1, No. 1 (March), pp. 51-63.

LIST OF TABLES

LIST OF FIGURES

Network Size	$\lvert \underset{\sim}{N}_E \rvert = 10$		$\lvert \underset{\sim}{N}_E \rvert = 40$	
Source of Lower Bound	$\lvert \underset{\sim}{N}_B \rvert = 2$	$\lvert \underset{\sim}{N}_B \rvert = 6$	$\lvert \underset{\sim}{N}_B \rvert = 2$	$\lvert \underset{\sim}{N}_B \rvert = 6$
LP relaxation	0.113	0.213	0.092	0.144
Capacity improvement procedure	0.048	0.135	0.074	0.123

Table 1

Measure of Near-Optimality for LP Relaxation and Capacity Improvement Procedure Using Four Test Networks of Different Sizes.

Flow Level* Source of Lower Bound	Low	Medium	High
LP relaxation	0.321	0.131	0.043
Capacity improvement procedure	0.225	0.115	0.035

*The low, medium, and high flow levels are equivalent to an average nonzero link flow of 1/3, 5, and 8 truckloads in the heuristic network design.

Table 2

Measure of Near-Optimality for LP Relaxation and Capacity Improvement Procedure Using Largest Test Network with Three Different OD Flow Levels.

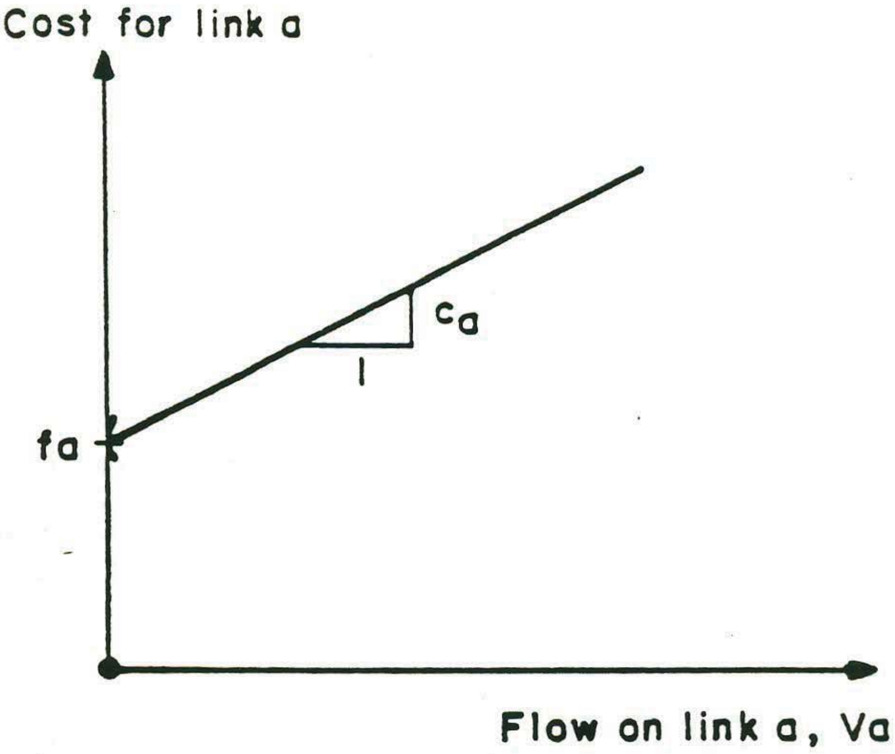


Figure 1

Link Cost Function

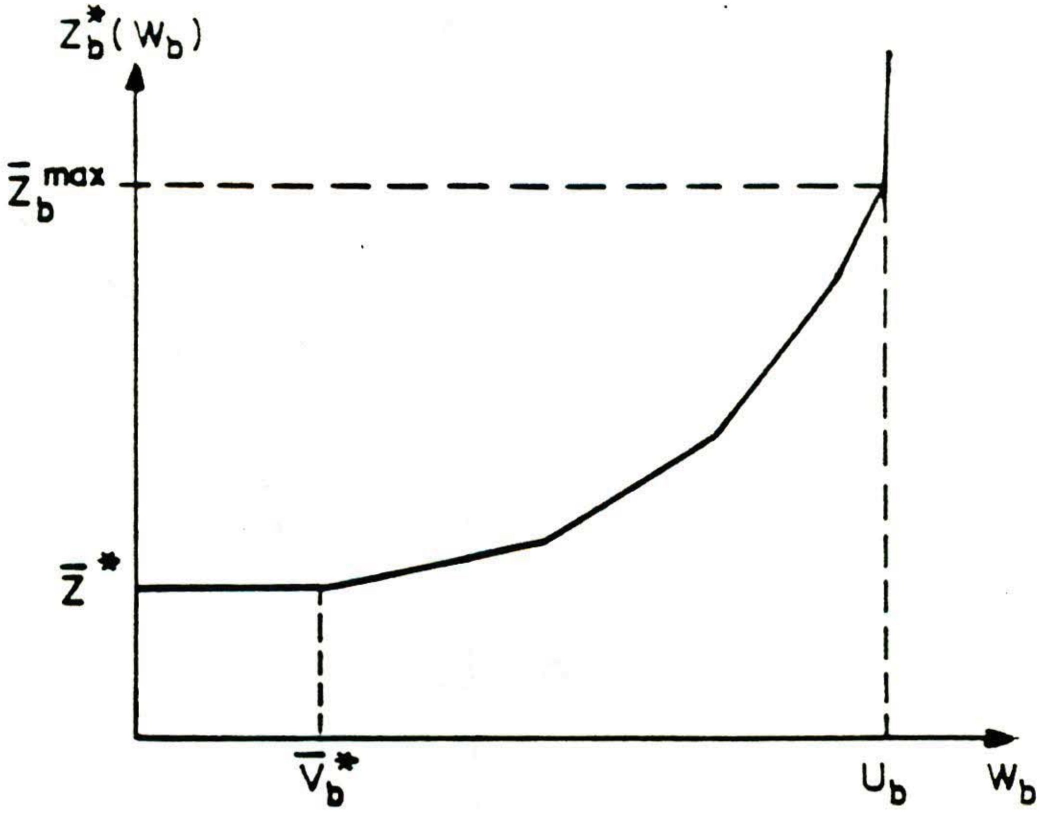


Figure 2

Optimal Objective Function Value of Program $\bar{P}_b(w_b)$ as function of w_b

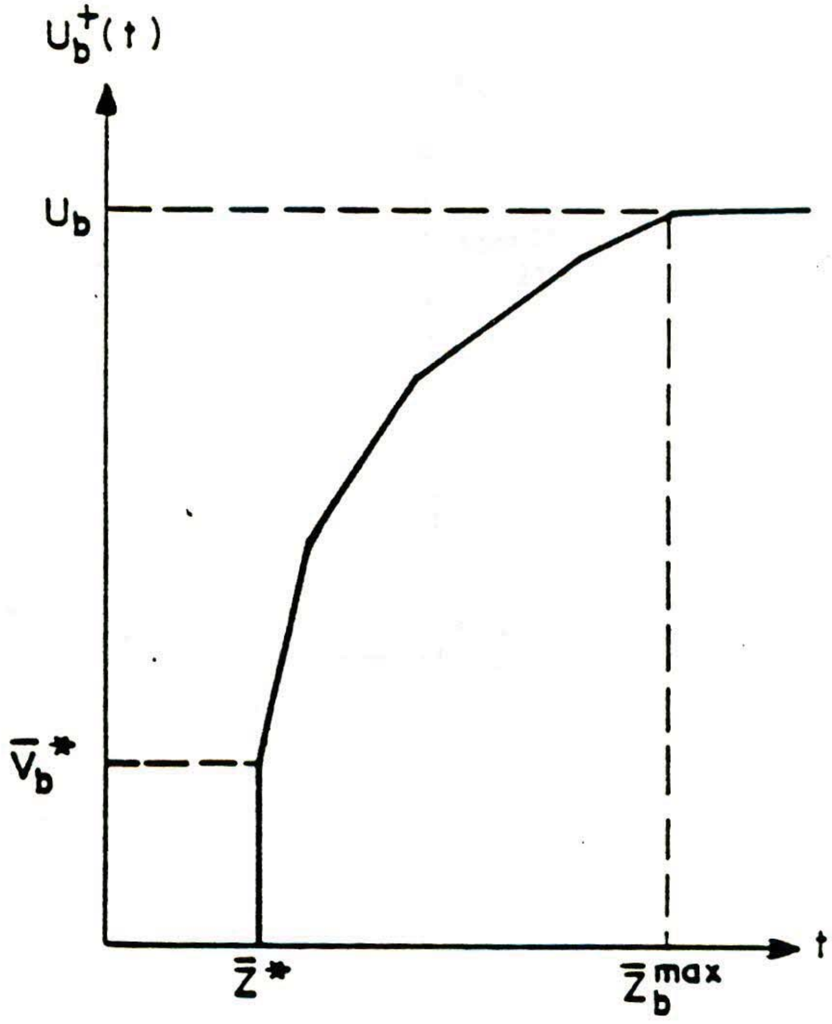


Figure 3

Capacity Improvement Parameter $u_b^+(t)$ for link bεA as function of $\underset{\sim}{t}$

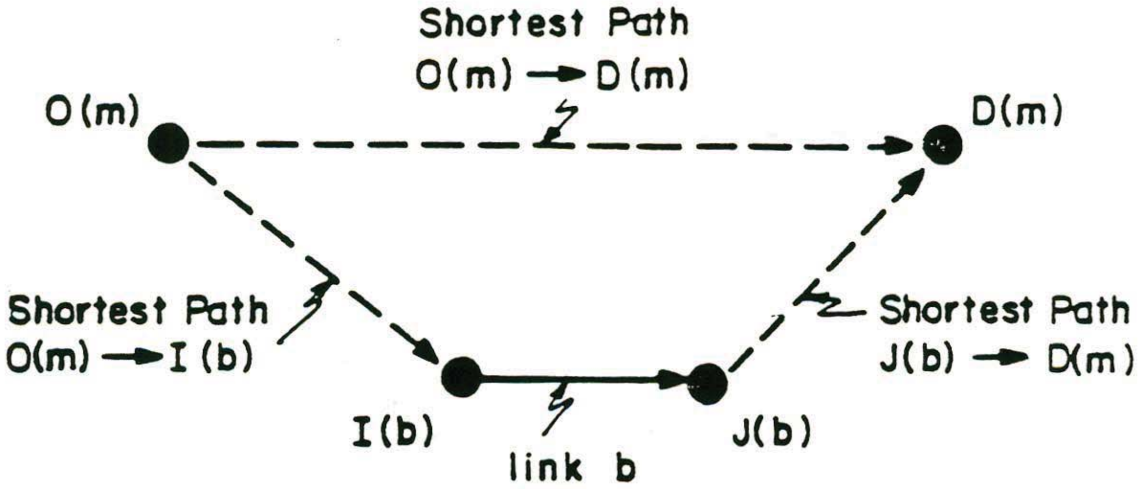


Figure 4

Cost Components of the Marginal Difference $\Delta_{b,m}$ for Link b and Market m

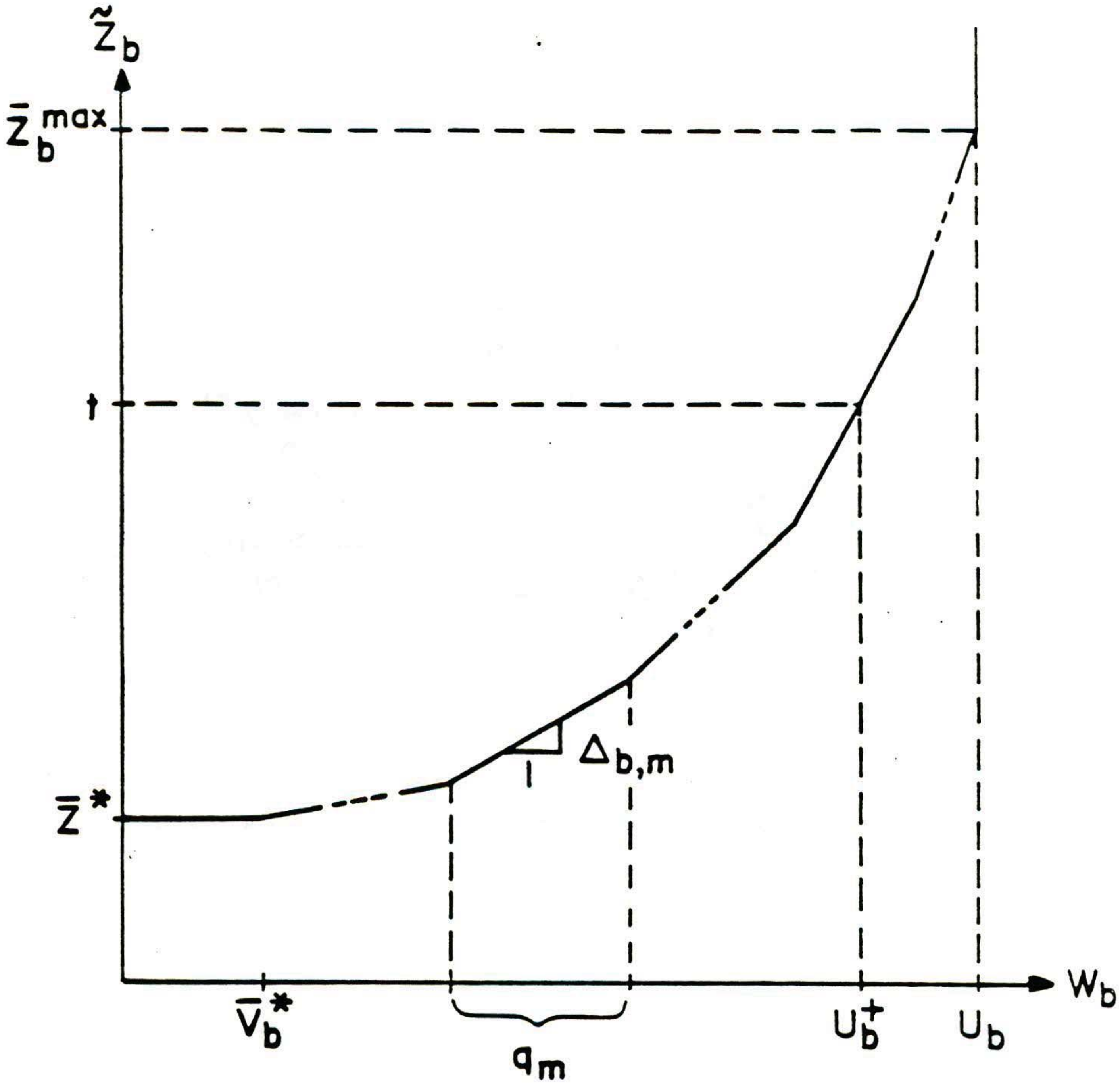


Figure 5

Determination of Improved Capacity Parameter for Link b Using Algorithm G_1

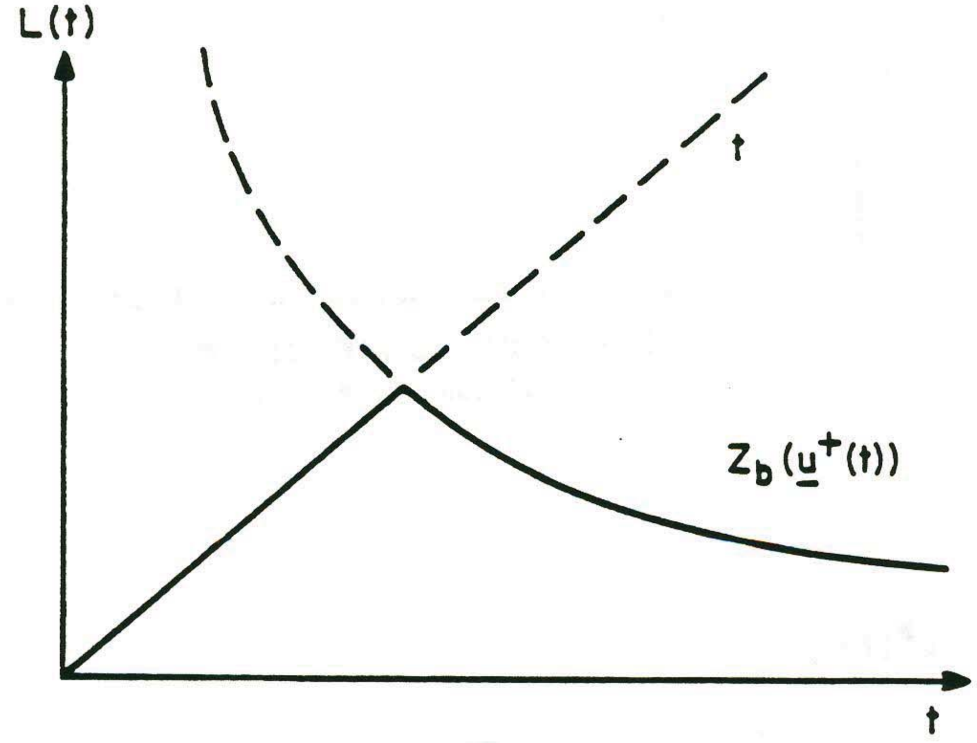
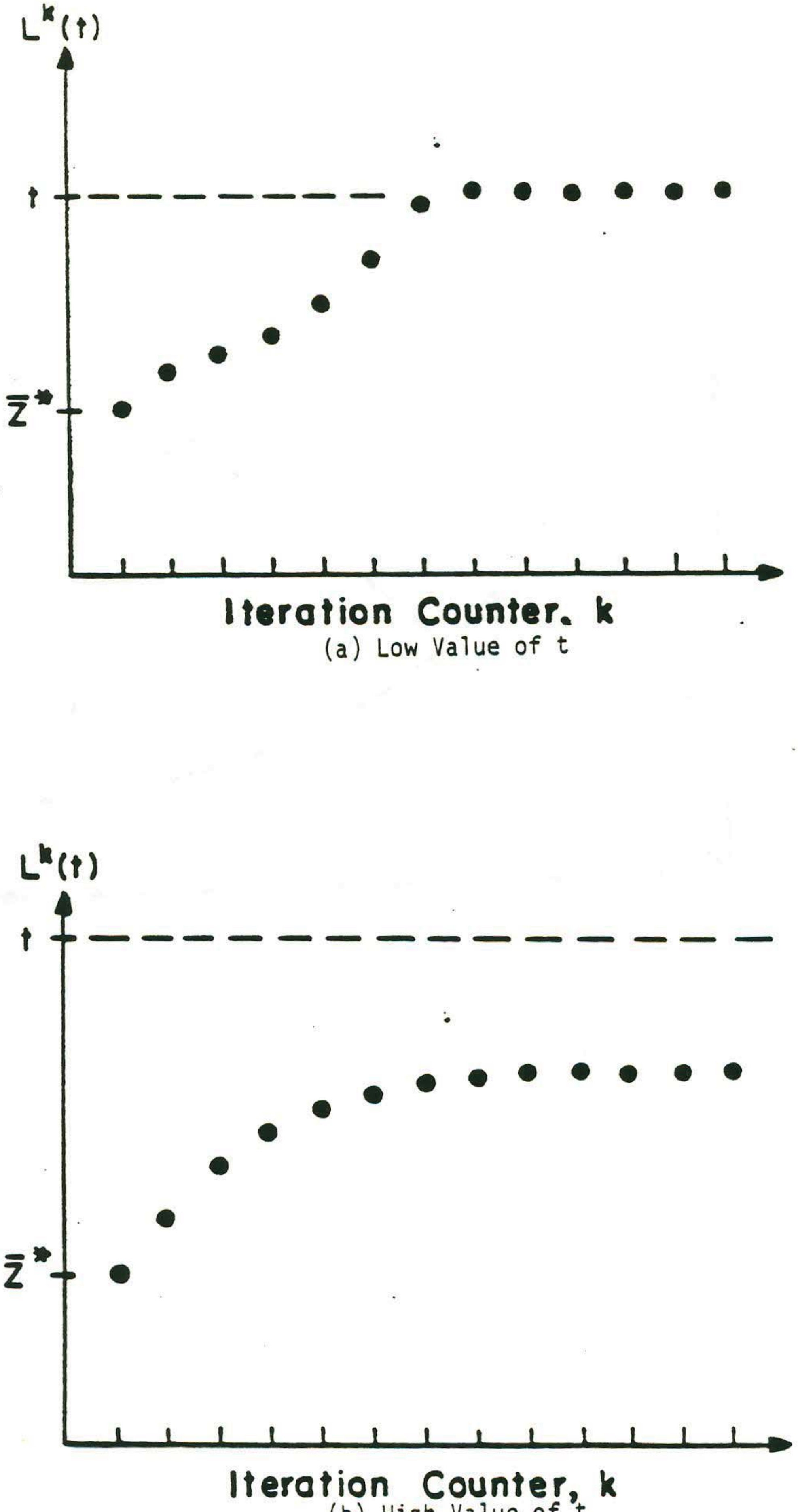


Figure 6

Lower Bound as a Function of Target Value t



(a) Low Value of t

(b) High Value of t

Figure 7

Lower Bound as a Function of Iteration Counter

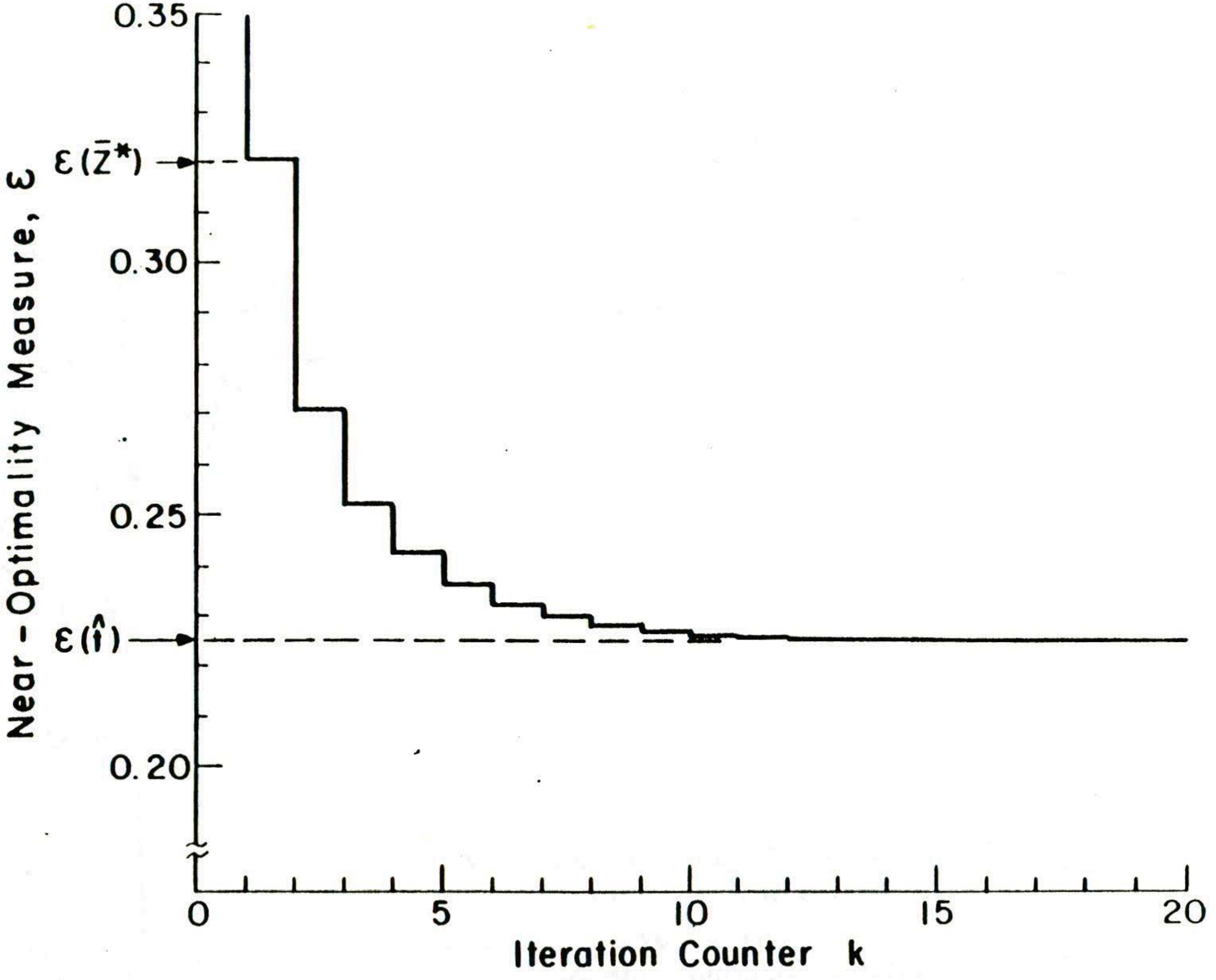
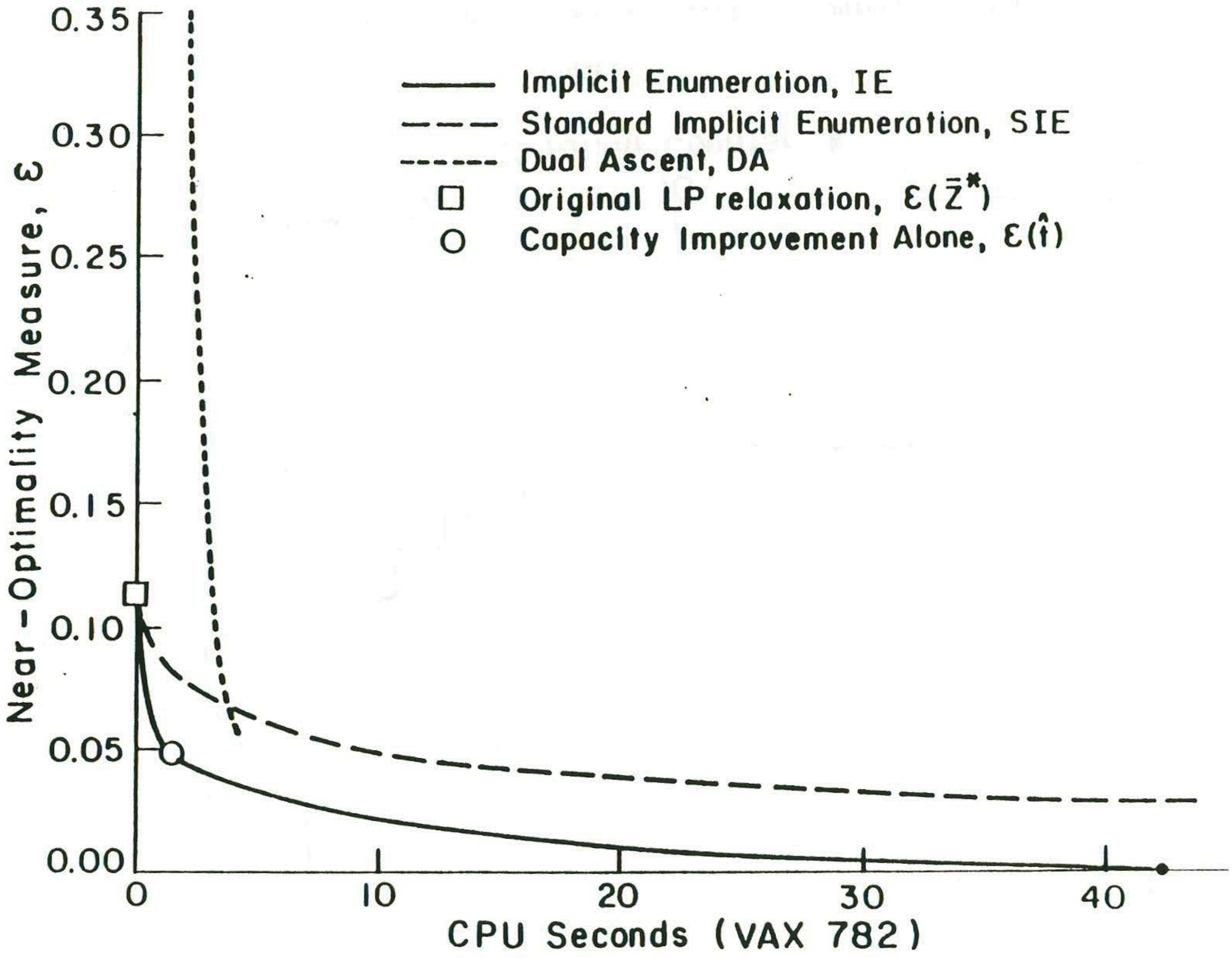


Figure 8

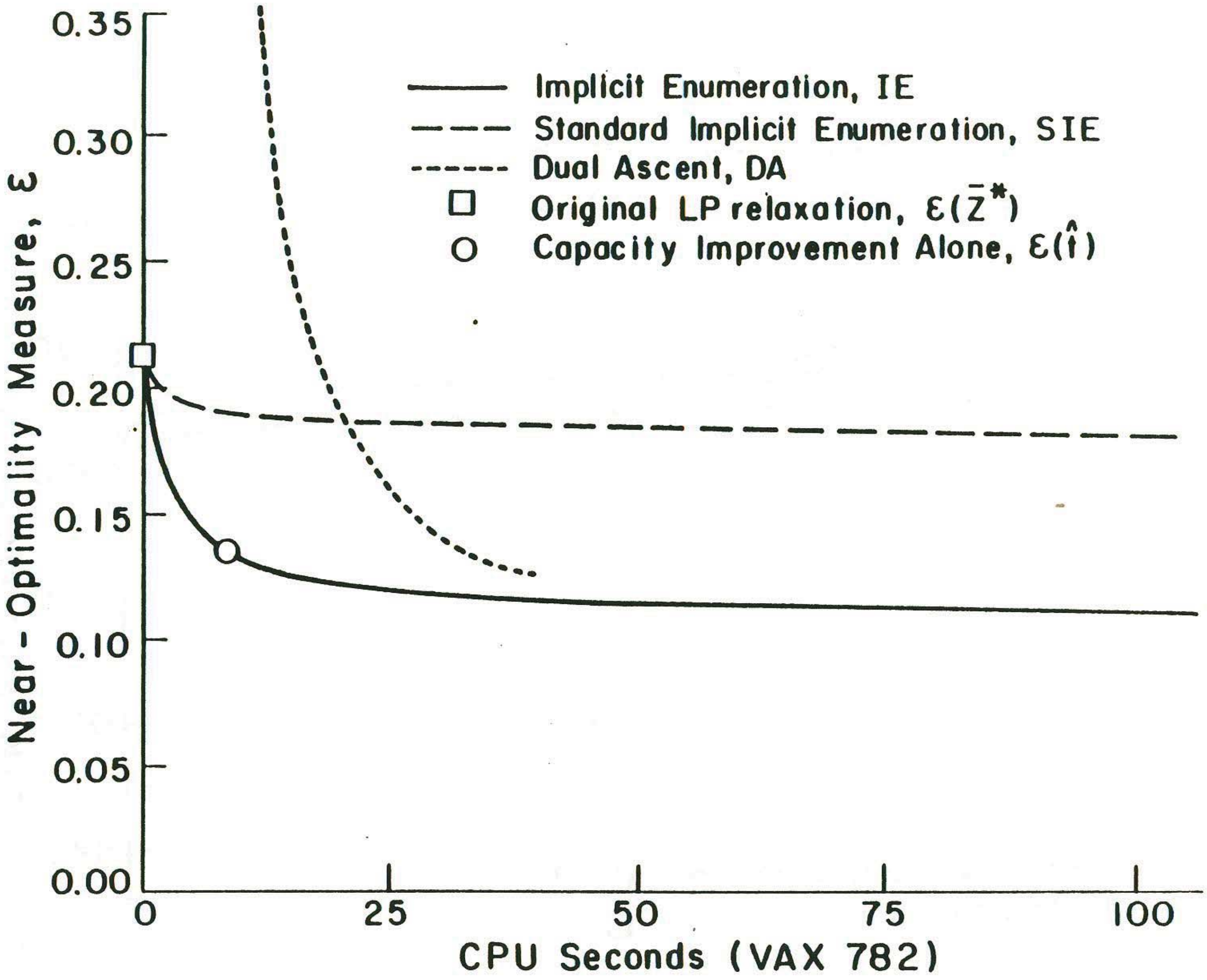
Near-Optimality Measure, ε, Versus Iteration Counter, k, for Capacity Improvement Algorithm, G_2



(a) $|\underset{\sim}{N}_E| = 10$, $|\underset{\sim}{N}_B| = 2$

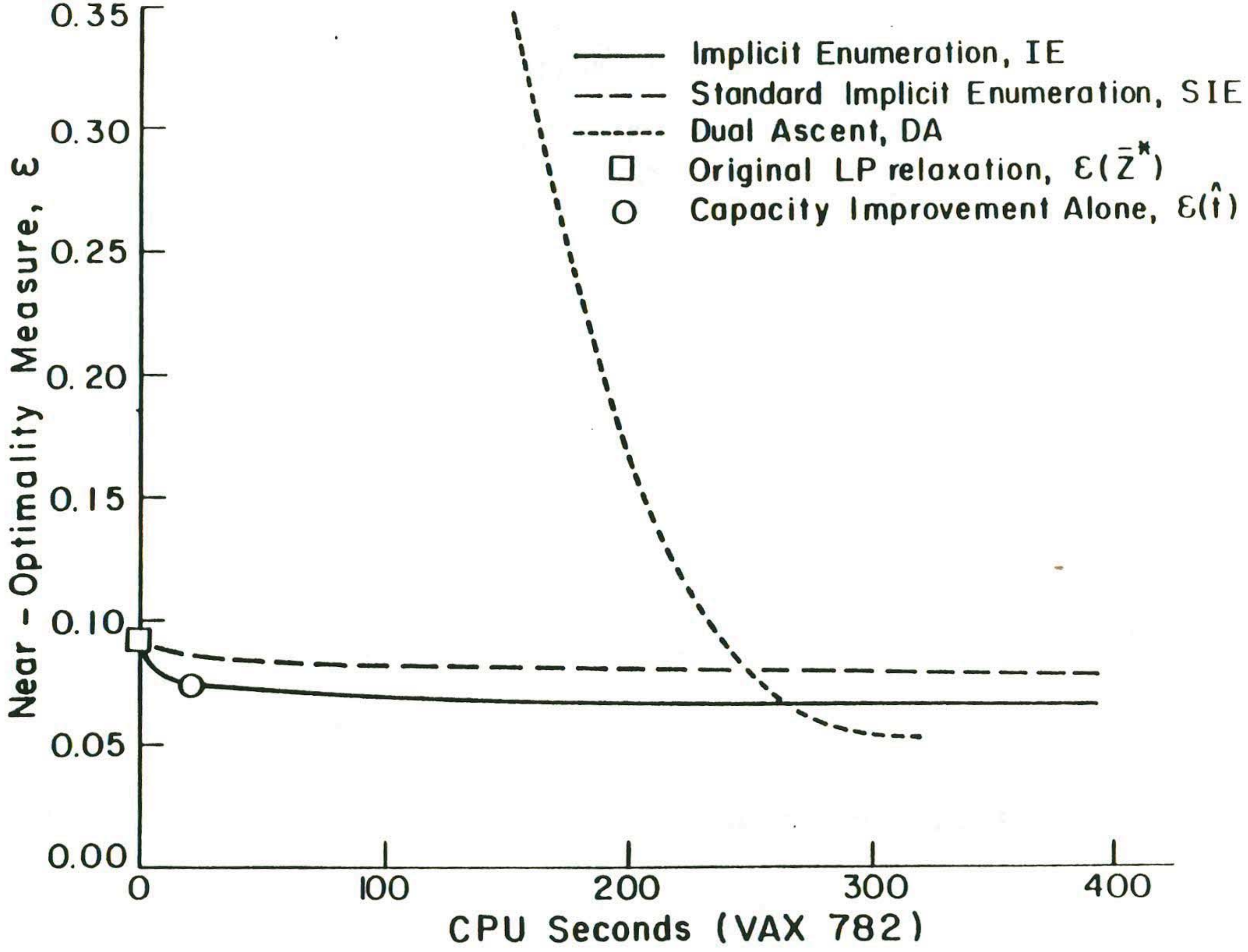
Figure 9

Near-Optimality Measure, ε, Versus CPU Time for Four Trial Networks of Various Sizes



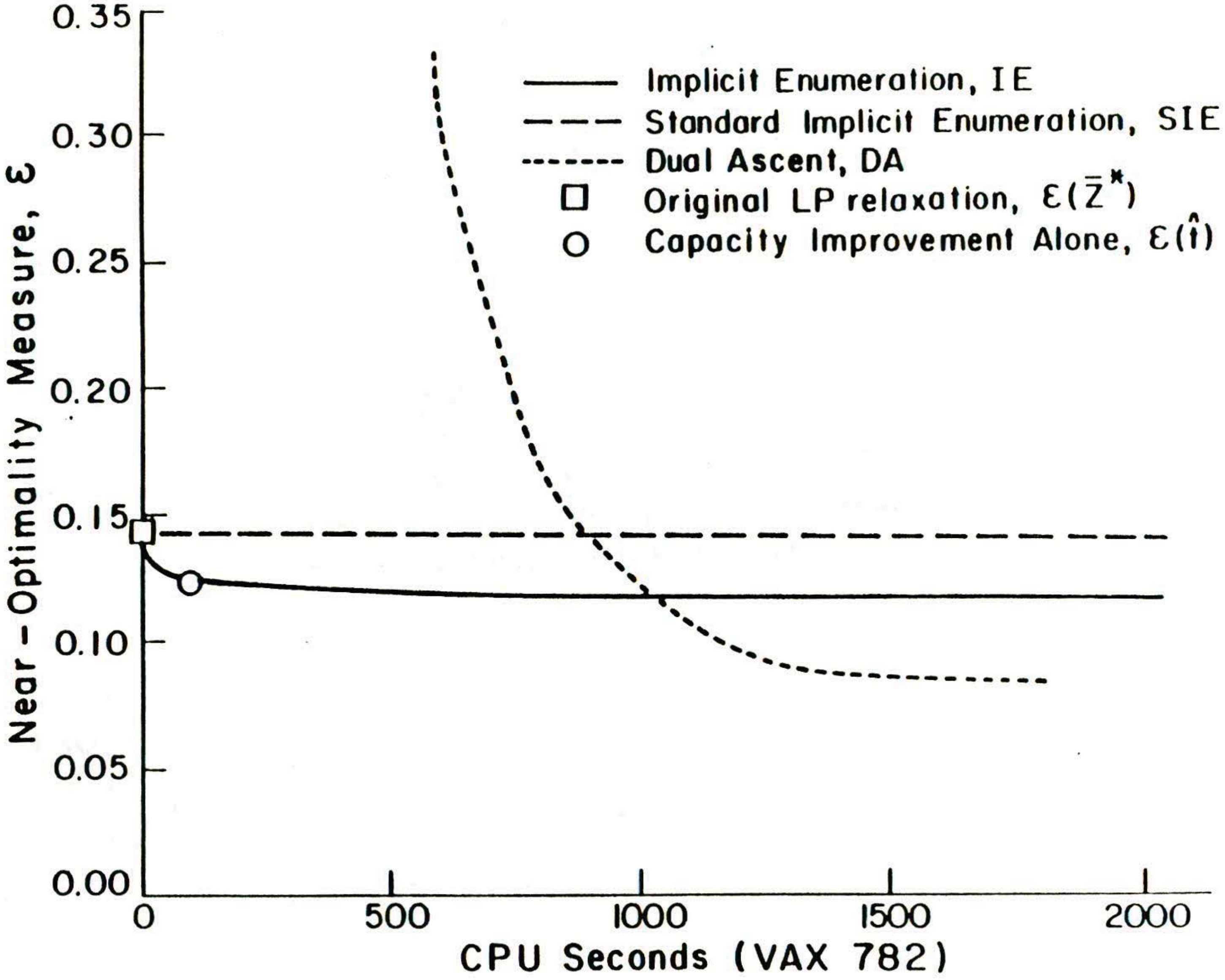
(b) $|\underset{\sim}{N}_E| = 10$, $|\underset{\sim}{N}_B| = 6$

Figure 9 (continued)



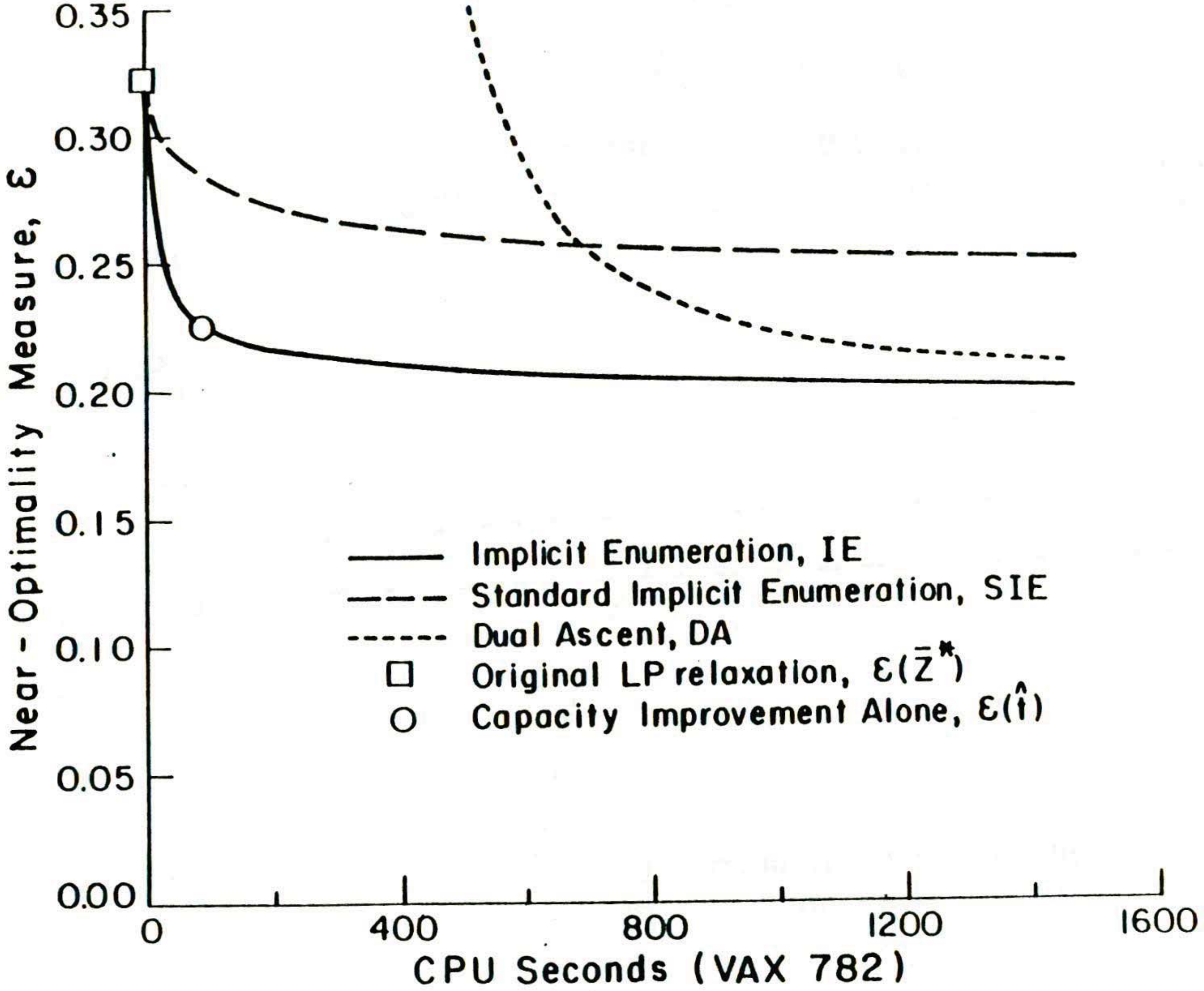
(c) $|\underset{\sim}{N}_E| = 40$, $|\underset{\sim}{N}_B| = 2$

Figure 9 (continued)



(d) $|\underset{\sim}{N}_E| = 40$, $|\underset{\sim}{N}_B| = 6$

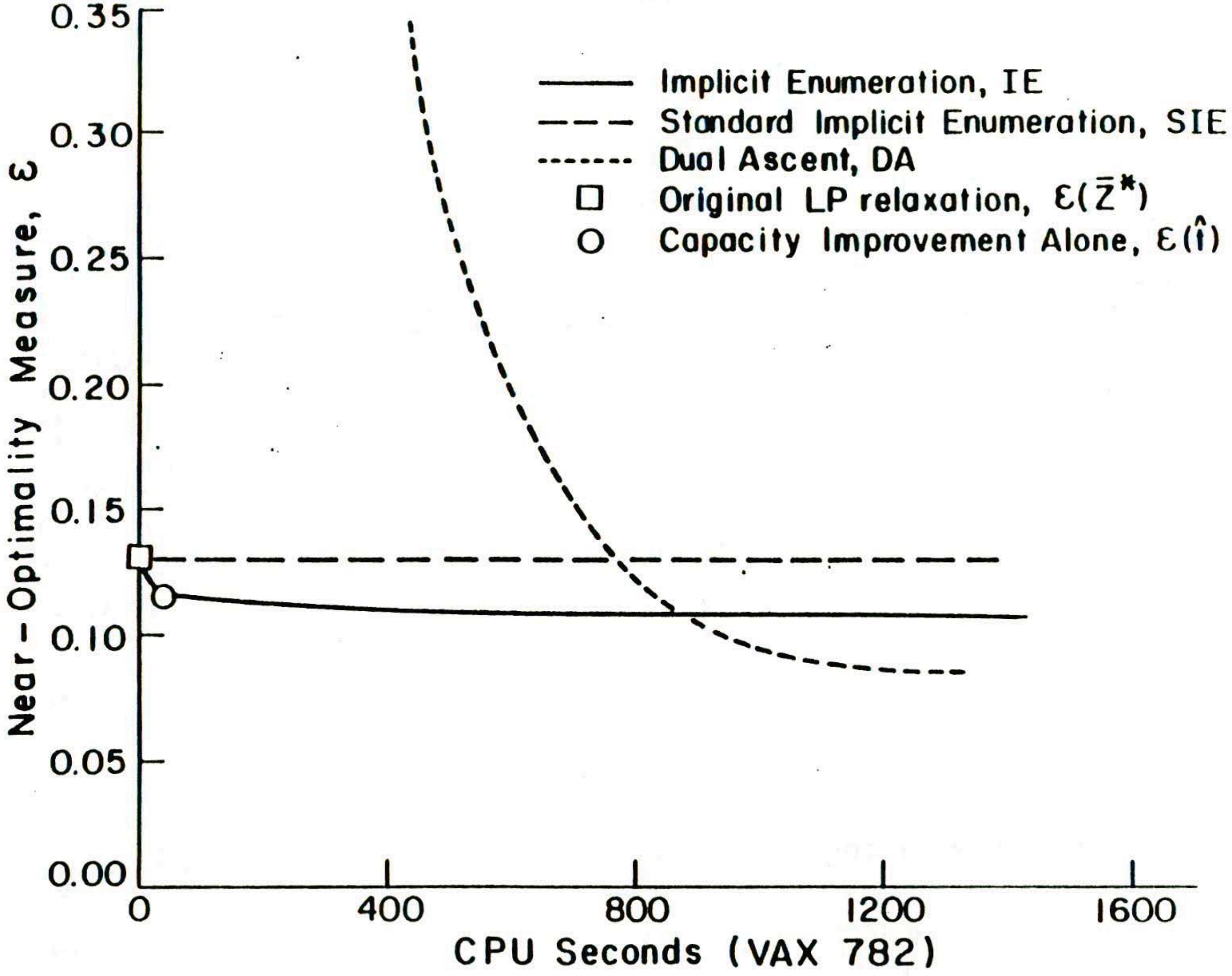
Figure 9 (continued)



(a) Low Flow Level

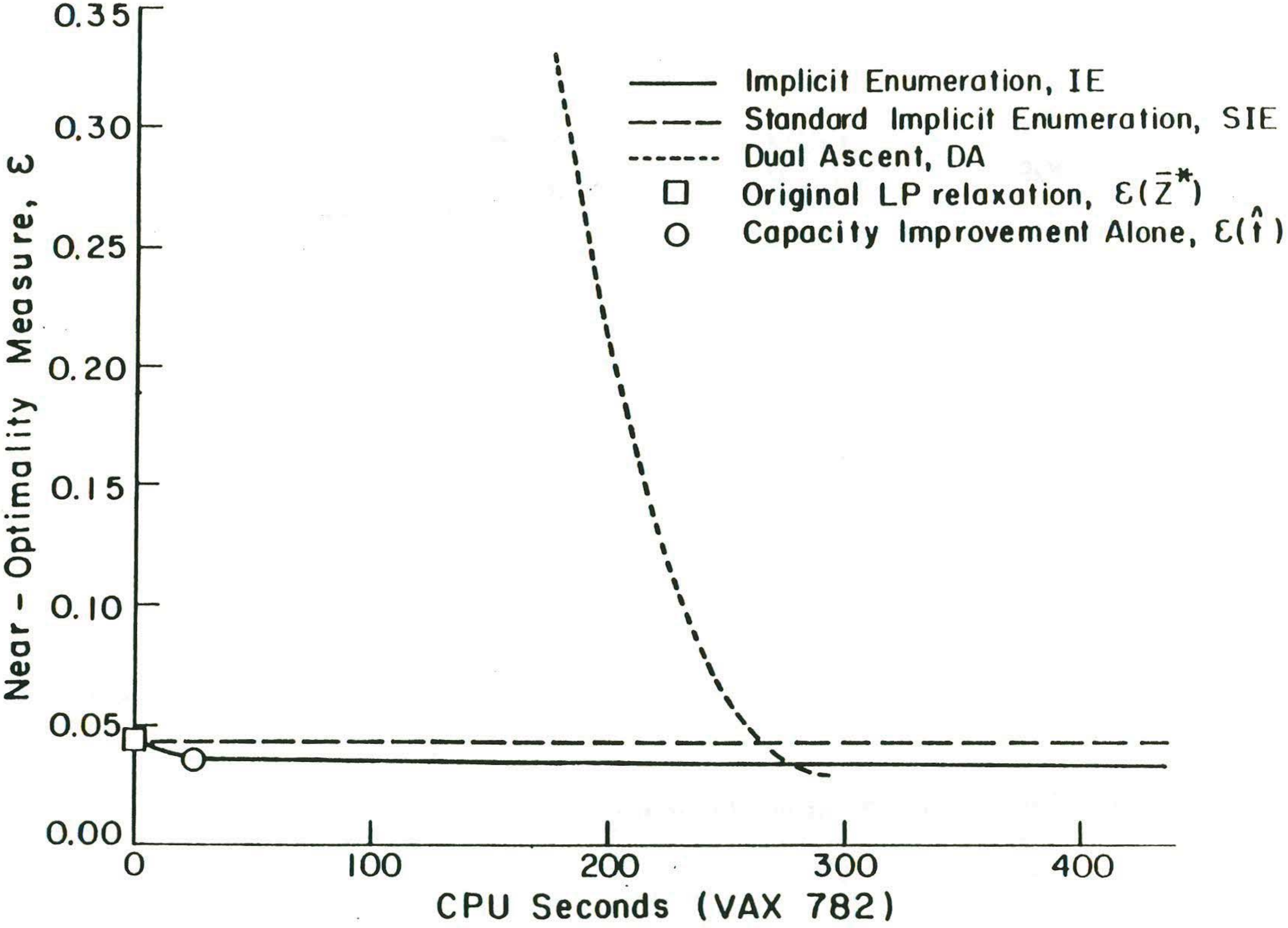
Figure 10

Near-Optimality Measure, ε, Versus CPU Time for Largest Trial Network Using Three Flow Levels.



(b) Medium Flow Level

Figure 10 (continued)



(c) High Flow Level

Figure 10 (continued)

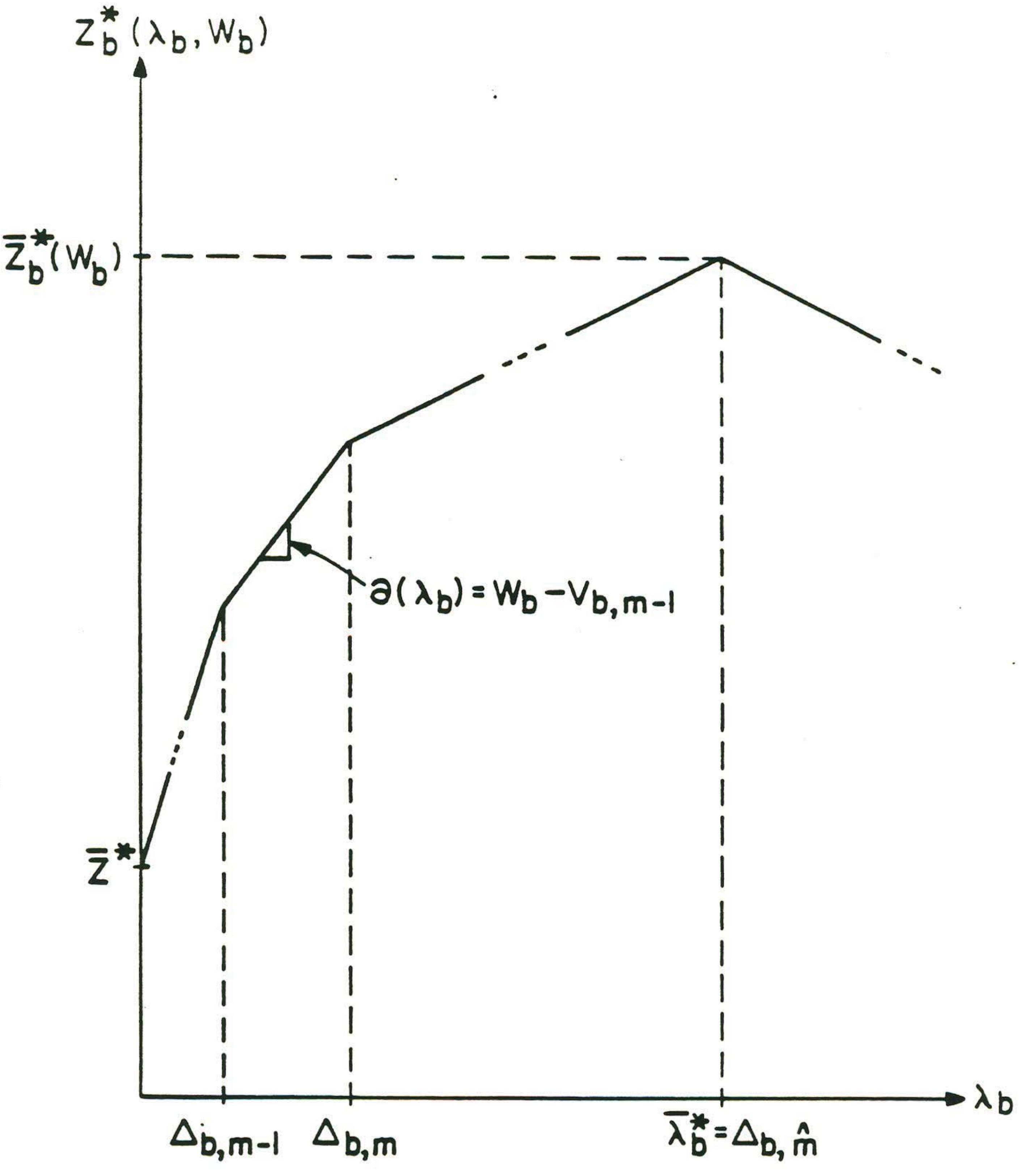


Figure A1

Objective Function Value of Program $\bar{R}_b^*(\lambda_b, w_b)$ as Function of λ_b