A LOCATION-ALLOCATION MODEL AND ALGORITHM FOR
OPTIMALLY LOCATING SHELTERS TO MINIMIZE EVACUATION TIMES

by

Todd B. Carter

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of
MASTER OF SCIENCE

in

Industrial Engineering and Operations Research

APPROVED:

Hanif D. Sherali, Chairman

Antoine G. Hobelka

Jeffrey D. Tew

July, 1989
Blacksburg, Virginia
A LOCATION-ALLOCATION MODEL AND ALGORITHM FOR
OPTIMALLY LOCATING SHELTERS TO MINIMIZE EVACUATION TIMES

by

Todd B. Carter

(ABSTRACT)

Location - allocation models are designed to seek the concurrent location of a set of service facilities and an allocation scheme to satisfy the demands of a set of customers or users of a given system. If the location-allocation model is based on a graph-theoretic formulation, then the demand-fulfilling items will move from a designated origin or origins, through arcs and transshipment nodes, to a set of destinations selected by the model. It is suggested in this research effort that such a modeling structure may be employed to simulate transportation evacuation conditions that may arise in the case of a natural disaster, namely a hurricane. A nonlinear mixed integer mathematical program is formulated to route passengers in automobiles on paths in the transportation network, such that the endangered area is evacuated in the minimum amount of time. One heuristic and two exact, convergent, implicit enumeration algorithms based on the generalized Benders' decomposition method are presented. The algorithms are designed to exploit the
inherent problem structure. Computational experience is provided against a set of realistic test problems formulated on the Virginia Beach network. Potential avenues for further research are also explored.
ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Hanif D. Sherali, who helped conceive this project. He was always available for guidance and support, and his optimism was a wonderful gift for me.

In addition, Dr. Hobeika deserves thanks for his part in conceiving this project.

I would also like to thank my parents, Mr. and Mrs. and my parents-in-law, Mr. and Mrs.

for their support (financial and otherwise) and patience in helping me to see this project through.

The biggest thanks of all goes to my wife who made it all possible. Thank you, Love.
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>ii</td>
</tr>
<tr>
<td>Acknowledgements</td>
<td>iv</td>
</tr>
<tr>
<td>List of Tables</td>
<td>vi</td>
</tr>
<tr>
<td>I. Introduction</td>
<td>1</td>
</tr>
<tr>
<td>II. Model Development and Problem Formulation</td>
<td>4</td>
</tr>
<tr>
<td>III. Literature Review</td>
<td>13</td>
</tr>
<tr>
<td>IV. Model Solution</td>
<td>24</td>
</tr>
<tr>
<td>V. Results</td>
<td>42</td>
</tr>
<tr>
<td>VI. Proposed Further Research</td>
<td>57</td>
</tr>
<tr>
<td>VII. Bibliography</td>
<td>65</td>
</tr>
<tr>
<td>VIII. Appendices</td>
<td>69</td>
</tr>
<tr>
<td>Vita</td>
<td>115</td>
</tr>
</tbody>
</table>
**LIST OF TABLES**

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>I. Heuristic Performance Against Six Test Problems</td>
<td>49</td>
</tr>
<tr>
<td>II. Implicit Enumeration Performance Against Six Problems</td>
<td>52</td>
</tr>
<tr>
<td>III. Scenarios 1, 2, and 3 Using Option 2 Problem</td>
<td>55</td>
</tr>
<tr>
<td>6 Baselines</td>
<td></td>
</tr>
</tbody>
</table>
Chapter I
Introduction

The threat of hurricanes and their potential for devastation is a part of life on the Atlantic Coast of the United States. Nearly every year some community is faced with the potential damage that a hurricane, if it makes its way ashore, can inflict. According to Neil Frank, a previous Director of the National Hurricane Center, it is an accepted fact that a major hurricane will strike a large community within the next ten years. The effects of the hurricane strike will be dear, both in terms of lives lost and property damage.

It is with these facts in mind that many seaside communities have invested in hurricane evacuation plans. The plans attempt to make available all existing knowledge about the storm and the threatened communities' demographics to local officials, who will then decide which communities are in serious danger, and when and to where the residents will be evacuated. One of the greatest tasks in developing a hurricane evacuation plan is to determine where evacuees should go to retreat from the storm's damaging power.

A primary researcher in the field of Evacuation Transportation Planning is Dr. A.G. Hobeika. Dr. Hobeika in conjunction with other researchers has developed over the past
seven years a computer simulation model entitled "TEDSS" - Transportation Evacuation Decision Support System (See Hobeika, Hwang, and Han [1982]). This system serves both to evaluate candidate evacuation plans before a storm’s arrival and to provide real-time management information for an evacuation, should a hurricane hit the coast.

As research and analysis progressed, and as follow-up studies were accomplished on the transportation evacuation plan that was implemented at the City of Virginia Beach [21], it became apparent that there was a need for conducting research on determining locations for potential hurricane evacuation shelters. Since it is typical that many evacuees go to shelters that are still contained in the threatened area, evacuation times can be markedly reduced by opening and staffing shelters in the most convenient locations, provided that all potential locations have been previously deemed safe and adequate. In previous research, the potential to reduce evacuation time by optimally locating shelters had not been closely examined. This thesis is dedicated to studying the problem of locating shelters in order to optimally reduce evacuation times.

We begin by presenting in Chapter 2 a statement of the problem being analyzed along with a mathematical formulation of this problem. The thrust of this new evacuation model is to develop an abstraction of the problem previously simulated by TEDSS, but with the additional consideration of creating
shelter locations as variables in the problem formulation. The idea is then to generate one or more sets of feasible shelter location solutions, which can be further analyzed in more detail by TEDSS (see Hobeika and Sherali, 1987). Chapter 3 contains a review of literature pertinent to this problem. Chapter 4 then presents a heuristic and two algorithms which solve the problem developed in Chapter 2.

Chapter 5 presents computational experience on a set of test problems designed to study the effect on algorithm performance of different factors which can contribute to road congestion during an evacuation procedure. A final chapter discusses avenues for further research in this field. Central to this chapter is a section containing an alternate problem formulation that employs a user-optimal approach as opposed to a system-optimal formulation. The appendices contain the source code for the algorithmic implementation, as well as a section containing hints for using MINOS 5.1 at Virginia Tech.

To summarize, a model has been developed to prescribe shelter locations, as well as traffic flow, for a system oriented emergency evacuation problem. Proposed algorithms include a heuristic solution procedure as well as two efficient branch and bound algorithms. Computational results are provided, along with recommendations for further research.
Chapter II
Model Development and Problem Formulation

In this chapter, we provide a formal description of the problem along with a mathematical formulation. To begin, visualize the area under consideration as a network of federal highways, and state and county roads. Let the road network be represented by a connected digraph $G(N,A)$, with a set of nodes $N$ and a set of directed arcs $A$. Each node $i$ contained in $N$ has a forward star, denoted $F(i)$, of all nodes $j$ in $N$ such that arc $(i,j)$ is an element of $A$: likewise, every node $i$ in $N$ has a reverse star, denoted $R(i)$, of nodes $k$ such that the arc $(k,i)$ is an element of $A$.

Within the node set $N$ is a subset $N_s$, which is defined as the source node set, or the set of all nodes from which evacuees originate. Typically, for any node $i$ in $N_s$, all local demand for service is modeled to be centrally located at node $i$. Let $D_i$ be the volume to be dissipated from node $i$, for every $i$ in $N_s$. We will assume that the rate of dissipation of $D_i$ for each $i$ in $N_s$ over a time horizon $T$ occurs according to a three-segment piecewise linear $S$-curve. Accordingly, compute $D_i^k$ as the rate of dissipation from source node $i$ in $N_s$ during period $k$ as

$$D_i^k = D_i \left( \rho_k/t_k \right), \quad k=1,2,3.$$ (1)
Another subset of nodes is the set of all potential destination nodes, denoted by $N_d$. $N_d$ represents all conceivable destinations for evacuees; so, to that end, define a binary decision variable as $y_j$, where $y_j=1$ if a shelter is chosen to be located at node $j$ in $N_d$, and $y_j=0$ otherwise. Since the potential shelter set is typically a highly diverse lot of public buildings, define $C_j$ as the accommodation capacity for shelter $j$ in $N_d$, should a shelter be located there. In addition, each shelter requires a staff of $s_j$ individuals. To protect against the possibility of understaffing a shelter or shelters, given that $S$ is the total available shelter staff for the evacuation region, we enforce the binary variable $y$ to be a member of the set $Y$, where

$$Y = \{ y=(y_j, j \in N_d) : \sum_{j \in N_d} s_j y_j \leq S, \text{y binary} \}. \quad (2)$$

Just as there are restrictions on the staffing ability for any danger area that is to be evacuated, so also there might be other potential restrictions that the modeler might include. Various possibilities include the total number of shelters to be opened, the available necessary evacuation needs (beds); transportation restrictions, or any number of other vital considerations. To this end, the constraint set $Y$ to which every potential shelter configuration $y$ must
belong, is further modeled as a set of linear inequalities of the form

\[ Y = \{y=(y_j, j \in N_d) : \sum_{j \in N_d} s_{ij} y_j \leq S_i, \ i=1,\ldots,m, \ y \text{ binary}\} \]  

(3)

where we assume that \( s_{ij} \geq 0 \), and \( S_i > 0 \ \forall \ i,j \). The number and actual formulation of these constraints is then a user-determined parameter in the code. We used constraints limiting the number of emergency personnel available for each evacuation destination.

Next, define the decision variables \( x^k_{pq} \) as the flow rate on link \((p,q)\) in \( A \) during period \( k, k=1,2,3 \). The flow rate, as is typical of transportation engineering research, is in units of vehicles per hour. To insure consistency in model formulation, it becomes necessary to represent both \( D_i \) and \( C_i \) in vehicle per hour, and vehicle units, respectively, using a constant vehicle occupancy factor, typically \( 1.5 \). We make the simplifying assumption here that these are steady-state rates which are achieved instantaneously during each of the three periods under consideration. In addition, the model is modified to accommodate the traffic occurring on the network due to evacuees who will not be using the shelters. Many people leaving the endangered area may opt not to use the shelters, but may rather leave the affected region altogether. It is therefore crucial to the success of an accurate model to allow for some superimposed steady-state network flow, for
each arc, that is not a part of the flow involved with getting evacuees from origins to destinations. To this end, define \( \ddot{x}_{pq}^k \) as this superimposed, extraneous flow rate for link \((p,q)\) in \(\mathbb{A}, k=1,2,3\).

To model vehicle travel times, the U.S. Bureau of Public Roads suggests the equation

\[
T_{pq} = t_{pq} \left[1 + \alpha \left(\frac{x_{pq}}{u_{pq}}\right)^\beta \right], \quad \forall (p,q) \in \mathbb{A}.
\]

In this equation, \( T_{pq} \) is the travel time on any link \((p,q)\) and is a function of the steady state volume rate \(x_{pq}^k\); \(t_{pq}\) is the uncongested or free-flow travel time; \(u_{pq}\) is the capacity of link \((p,q)\); and alpha and beta are non-negative model parameters, usually 0.15 and 4.0 respectively.

To put everything together, the steady state flow rate for link \((p,q)\) during evacuation period \(k\) is \(x_{pq}^k + \ddot{x}_{pq}^k\), and the volume of evacuees being guided to shelters that crosses link \((p,q)\) in period \(k\) is \(t_k x_{pq}^k\): then the total time spent by evacuees seeking shelters on the road network is

\[
\sum_{(p,q) \in \mathbb{A}} \sum_{k=1}^3 t_{pq} \left[1 + \alpha \left(\frac{(x_{pq}^k + \ddot{x}_{pq}^k)}{u_{pq}}\right)^\beta \right] t_k(x_{pq}^k).
\] (5a)

An alternate objective function could be represented by

\[
\sum_{(p,q) \in \mathbb{A}} \sum_{k=1}^3 t_{pq} \left[1 + \alpha \left(\frac{(x_{pq}^k + \ddot{x}_{pq}^k)}{u_{pq}}\right)^\beta \right] t_k(x_{pq}^k + \ddot{x}_{pq}^k).
\] (5b)

In this objective formulation, the modeler would attempt to minimize total evacuation-related traffic time on the network.
instead of minimizing evacuation traffic time of persons using the shelter locations.

The units of this function are vehicle hours, and this is the quantity which is to be minimized. The complete formulation can be written now as follows:

\[
\text{Minimize } \sum_{(p,q) \in A} \sum_{k=1}^{3} t_{pq}[1+\alpha((x_{pq}^{k}+x_{pq}^{k})/u_{pq})^{\beta}]t_{k}(x_{pq}^{k})
\]

Subject to: \[ D_{i}^{k} + \sum_{p \in R(i)} x_{pi}^{k} - \sum_{q \in F(i)} x_{iq}^{k} = z_{i}^{k}, \forall i \in N, k=1,2,3 \] \[ \sum_{k=1}^{3} z_{i}^{k}t_{k} \leq C_{i}y_{i}, \forall i \in N \]

\[ y_{i} \equiv (z_{i}^{k}) \geq 0, x_{pq}^{k} \geq 0, (x_{pq}^{k}+x_{pq}^{k}) \leq u_{pq}, \forall (p,q) \in A, k=1,2,3. \]

It is understood that \( D_{ij}^{k} \) is necessarily zero for non-source nodes, and that \( y_{i} \) is defined as zero for non-demand nodes.

Some important remarks are appropriate now. Note that the objective function is convex for nonnegative \( x \)-variables and that all the constraints are linear. Thus, for any \( y \) in \( Y \), problem \( P_{a} \) is a linearly constrained, convex program. The set of constraints given by (7) are known as flow balance equations: one exists for every node \( i \) in \( N \). The variable \( z_{i}^{k} \) in (7) is the net demand accumulation rate at any node \( i \) in \( N \) during period \( k \), \( k=1,2,3 \). Logically, no volume should accumulate at nodes other than the chosen demand nodes of set \( N_{d} \), and this restriction is precisely what constraint set (8) accomplishes. Equations (8) restrict volume from accumulating
at both non-demand nodes and unchosen demand nodes by defining \( y_i \) as zero in these cases. In addition, (8) insures that shelter capacity \( C_i \) is not overextended, by using \( C_i \) as a coefficient for \( y_i \). Finally, the last constraint set (9) includes the shelter configuration, nonnegativity, and bounds restrictions.

It is possible to simplify and improve problem \( P_a \) by using a one-period approximation of the flow rates. Specifically, assume that all flow rates \( x_{pq}^k \) and \( \bar{x}_{pq}^k \), during period \( k \), are proportional to the slope of the piecewise linear function \( p_k/t_k \), which was described earlier. Thus it is possible to infer that the flow pattern over the total evacuation period \( T \) remains unchanged, and that its intensity varies in direct proportion to the evacuation dissipation slope rate. Mathematically, the above assumption states that

\[
x_{pq}^k(t_k/p_k) = x_{pq}, \quad \forall (p,q) \in A, \quad k = 1,2,3
\]  

(10a)

and where we assume that

\[
\bar{x}_{pq}^k(t_k/p_k) = \bar{x}_{pq}, \quad \forall (p,q) \in A, \quad k = 1,2,3.
\]  

(10b)

The objective function (6) may now be collapsed, by using \( p_1 + p_2 + p_3 = 1 \) and inserting (10a) above to obtain

\[
\sum_{(p,q) \in A} \left[ t_{pq}x_{pq} + \left[ \frac{\alpha t_{pq}/(u_{pq})}{\beta} \right] x_{pq}(x_{pq}+\bar{x}_{pq})^{\beta} \sum_{k=1}^{3} (p_k^{\beta+1}/t_k^{\beta}) \right].
\]  

(11)

which may be further simplified to

\[
\sum_{(p,q) \in A} t_{pq}x_{pq} + \tau_{pq}x_{pq}(x_{pq}+\bar{x}_{pq})^{\beta}
\]  

(12)
where

\[ \tau_{pq} = \left[ \alpha_{pq}/u_{pq}^0 \right] \sum_{k=1}^{3} \left[ p_k^{\alpha+1}/t_k^2 \right], \forall (p,q) \in A. \]  

(13)

In addition to the above simplifications, it is also possible to completely eliminate the piecewise linear approximation function from the flow balance equations (7) by employing (1) and (10). It is simple to arrive at the following:

\[ D_i + \sum_{p \in R(i)} x_{pi} - \sum_{q \in F(i)} x_{qk} = \gamma^k \left( t_k/p_k \right), \forall i \in N, k=1,2,3. \]  

(14)

Noting the absence of any time-dependent quantities on the left side of the previous equation, it is clear that the quantity \((\gamma^k t_k)/p_k\) is a constant \(\gamma_i\), for every \(i\) in \(N\). Since \(\gamma_i p_k = \gamma_i^k t_k\), and \(p_1 + p_2 + p_3 = 1\), we get

\[ \gamma_i = \sum_{k=1}^{3} \gamma_i^k t_k, \]  

(15)

and so, the time-dependent constraint set (8) simplifies to

\[ \gamma_i \leq C_i y_i, \forall i \in N. \]  

(16)

Obviously, one could eliminate the \(\gamma_i\) variables from the problem formulation at this point, but the reason not to model them out will become clear in the chapter detailing model solution.

In addition, to determine the upper bound \(v_{pq}\) on the flow rate variables \(x_{pq}\), we note from (9) and (10) that

\[ x_{pq} + \bar{u}_{pq} \leq v_{pq}, \forall k, k=1,2,3 \]  

(17)
asserts

\[(p_k / t_k)(x_{pq} + \bar{x}_{pq}) \leq u_{pq} \quad \forall k, k=1,2,3. \tag{18}\]

This gives

\[x_{pq} \leq (t_k / p_k)u_{pq} - \bar{x}_{pq} \quad \forall k, k=1,2,3. \tag{19}\]

It is then apparent that to allow for all conceivable flow rates, it is necessary to let \(u'_{pq} = (u_{pq})(t_k / p_k) - \bar{x}_{pq}\), where \((t_k / p_k)\) is the minimum slope quantity, from the piecewise linear evacuation dissipation function above, i.e.,

\[(t' / p_k) \leq (t_k / p_k) \quad \forall k, k=1,2,3. \tag{20}\]

Consequently the problem to be solved is as follows:

\[
P: \quad \text{Minimize} \quad \sum_{(p,q) \in A} \left[ t_{pq}x_{pq} + \tau_{pq}x_{pq}(x_{pq} + \bar{x}_{pq})^3 \right] \tag{21}\]

\[
\text{Subject to:} \quad - \sum_{p \in R(i)} x_{pi} + \sum_{q \in F(i)} x_{iq} = \begin{cases} D_i - z_i & \forall i \in N_d \\ D_i & \forall i \in N-N_d \end{cases} \tag{22}\]

\[0 \leq z_i \leq C_i y_i \quad \forall i \in N_d \tag{23}\]

\[y \in Y = \{y \text{ binary: } \sum_{j \in N_d} s_{ij}y_j \leq S_i, i=1,...,m\} \tag{24}\]

\[0 \leq x_{pq} \leq u'_{pq} \quad \forall (p,q) \in A. \tag{25}\]

Observing the problem formulation, it is important to realize that if any \(y\) in \(Y\) is fixed, the problem reduces to a convex cost network flow mathematical program. This problem
class can be easily solved, using the convex simplex method, for example, as in Kennington and Helgason [1980].

In conclusion, the hurricane evacuation problem has been modeled as a nonlinear, mixed integer network flow problem with side constraints. As will be evident in the next chapter, this particular class of problem has not received adequate attention in the literature, leaving little in theoretical constructs behind. However due to some special structures purposefully retained during the modeling phase, this program may be solved quite effectively.
Chapter III

Literature Review

In the previous chapter, it was shown that the general form of the problem in question falls into the category of a nonlinear mixed integer network flow program with side constraints. The problem was also referred to as a location-allocation problem, due to the fact that not only does the model develop an optimal origin-to-destination flow allocation pattern, it also determines the optimal location of shelters out of an acceptable feasible candidate set. This chapter reviews associated research that has been done in the area of location-allocation problems, including discrete, mixed integer programming formulations. The evidence appears to suggest that the particular problem formulation presented earlier has been previously untreated until now.

To begin, first notice that the objective function equation (22) is driven by a system-optimal objective; namely, to minimize total vehicle hours accrued during the evacuation procedure. In the optimization literature, this objective formulation style is referred to as the normative approach; see Steenbrink [1974]. A parallel school of thought for similar problems is known as the descriptive or user-oriented objective approach, which was described by Nash [1951]. Nash's research into non-cooperative game theory asserts, for our purposes here, that there exists an optimal flow balance
such that no network user can reduce travel time by taking an alternate route. The basis for using this approach is realistic: it would seem that drivers often naturally behave in this manner. More research is available on the subject of user-optimal objectives, including papers by Dafermos and Sparrow [1969], LeBlanc [1975], and Frank [1981].

The work by Dafermos and Sparrow is very informative and contains detailed strategies for developing objective function statements for both normative and descriptive problem formulations. A nontrivial example is worked through, and it becomes apparent that, given the right network characteristics, the time formulations yield different optimal flow equilibriums. It is in this work first that the justification for the system optimization approach, for cases in which a traffic pattern is attempted to be regulated by a central authority, is touted as the correct optimization strategy.

LeBlanc et al. [1975] point out that while system-oriented models are more likely to appear in urban transportation network formulations like the one under study here, occasionally some network users may be forced to incur an unnecessarily (from their viewpoint) high travel time in order to decrease the system's objective function value. The authors further state that the above behavior pattern would not typically be found: in other words, drivers, when given a choice, do not take the long way around. However, when
danger is detected in an area, and a central controller begins to direct traffic flows (to varying degrees of success), it is our opinion that the concept of a system-oriented objective is appropriate.

Marguerite Frank [1981] represents a compelling theoretic concept in her paper. The Braess paradox refers to a user-optimized network assignment model which possesses an unusual trait. It happens that, if one inserts a directed arc into this network, thereby creating a third origin-destination path, the cost is raised for every person on the network. Certain necessary conditions that a network must possess to make it a Braess network are presented. It is important, however, to realize that when optimizing and employing a user-oriented objective, certain intuitive dominance relationships may not hold up.

Since user-optimized functions represent another possible outlook on this problem, more time is given to developing a user-optimal objective function and problem formulation in a later chapter. It should be noted that this discussion will proceed in a manner analogous to the previous chapter, but with slightly less detail.

Due to the presence of a central planning authority, the median location approach as opposed to the center location approach was chosen for this application. The median location approach contains as its basis a solution to the minisum location program. The objective of a minisum problem might
be, for example, to locate on a two-dimensional plane the point which minimizes the sum of all weighted distances to the point being located: thus, minisum. The minimax location problem might seek to locate the point on a plane which minimizes the maximum weighted distance to that point: thus minimax.

A comprehensive survey on the field of network location programs is contained in Tansel, Francis, and Lowe [1983a], which charts the evolution and solution procedures to the two classes of problems, and provides excellent references to other authors' related research.

Another such source is the text by Handler and Mirchandani [1979] on network location problems. In a discussion on locating emergency medical services and other related facilities, it is suggested that the minisum approach be adopted so as to minimize the total amount of travel time incurred by all network users. The authors present graph-theoretic approaches to the minisum problem, and offer a short section that discusses heuristic-, primal-, and dual-based procedures for solving the minisum problem. The references and discussions are pertinent only for the location aspect of the problem at hand. No procedures or references are given for a location-allocation formulation.

One of the more referred to works in the field of network location problems is the paper by Kariv and Hakimi [1979]. The paper begins at the basics with mathematical and graph-
theoretic definitions and graduates to questions of algorithmic convergence and problem classification. Among other things it is asserted that the general p-median problem, where p is the number of points under consideration, is NP-hard. Finally, an example is worked and results given, but again, the paper is concerned with the location only, not the location and allocation combination.

The same contention exists with the paper by Hansen et al. [1985]. Although the work contains excellent definitions and good computational experience results, their research was confined strictly to the location of centers and medians on networks. The authors employ a branch and bound algorithm, which is in the spirit of the algorithm presented later in this thesis.

The location-allocation problem formulation is discussed in Francis and White [1974]. This text provides a problem formulation for the discrete-site capacitated class of network location problems, in which a set of binary variables stand for decision to locate while flow variables represent the amount that is to be allocated to the network arcs. Various authors have examined this problem, and a discussion of their findings proceeds now.

Akinc and Khumawala [1977] derived an efficient branch and bound algorithm for a capacitated location problem. Theirs is a mixed integer problem formulation and includes a set of constraints in the style of the constraints presented
earlier in this work (3a). The authors are quick to point out that much research has been done in location theory. They next describe a branch and bound algorithm, specifying bounding procedures, and branching node selection strategies. Facilities are fixed open or closed by the use of a marginal cost test, while upper and lower bounds are computed via free integer variable fixation (establishing a dominance relationship) and relaxation of the integrality restrictions, respectively. Eight branching variable selection strategies are outlined along with their performance on a selection of test problems. The authors make it clear that their procedure could be improved by either primal-dual or out-of-kilter network routines. The paper is descriptive and informative, but does not discuss the possibilities of nonlinearity or side constraints, both of which are crucial to the problem at hand.

Bitran et al. [1981] present the concept of inverse, or dual optimization in their paper. The authors assert that the constrained primal problem can often be solved, after reformulation, as an unconstrained dual problem, and they forward a proposed procedure for reconciling duality gaps through the use of nonlinear multiplier functions. The problem formulation is similar to the previous reference, with integer flow and binary location decision variables. The objective function differs from our problem due to the fact that their findings are presented in the context of a fixed plant location problem. Plant location incurs an initial
fixed cost as a function of the binary location variables, a feature which does not appear in the objective of the problem at hand. Still, computational experience and areas for further research are divulged. The problem bears some semblance of our problem without the nonlinear objective and side constraints.

In a similar, but shorter paper, Barrie Baker [1982] proposes a formulation and solution procedure for the capacitated warehouse location problem CWLP. Baker allows for the fixed costs to be eliminated from the objective, and he also includes side constraints on the flow variables. No allowance is made for side constraints in the binary variables, nor is a complete algorithm presented. Baker does employ a Lagrangian relaxation technique similar to other works to bound the objective function, and he also develops an interesting general constraint inequality that unites the flow and the location variables. In practical cases, the general form decomposes into familiar constraint inequalities such as the necessity of the closed interval \([0,1]\) for \(y\), or the constraint set restricting overflow into destination nodes. No mention is made of the potential for nonlinearities in the objective function.

Christofides and Beasley [1983] extend the CWLP to develop a Lagrangian-based lower bound for the objective function. In addition, the authors attempt to maximize this lower bound by alternately solving a linear programming
relaxation of a constrained knapsack problem, then a zero-one program which sets destinations "open." Penalties are developed to prevent false optimization pivoting, and two reduction tests are presented. In the subgradient search procedure the authors include a frequency factor for recomputing the Lagrange multiplier step size for speedier convergence. This methodology is similar to the one contained in the nonlinear problem solving code for the problem contained in this thesis.

A similar approach is given in Geoffrion and McBride [1978], although it is an earlier work. Geoffrion's formulation includes potential for fixed plant changes, but there is allowance made for the implementation of a side constraint set for the flow and location variables which could be easily converted to a side constraint set for the latter set alone. As before, the authors employ Lagrangian relaxation techniques: they do use a new twist in choosing to dualize both the continuous and binary constraint sets. The result lets the module uncouple the problem into separate problems, one for each destination, which are easily solved. The discussion highlights the ability of this formulation to outperform previous dualizations, particularly in the "tightness" of the lower bound. Penalty functions are resurrected and finally a Gomory cutting plane is compared to the Lagrangian formulation.
Nauss [1978] develops a specialized Lagrangian relaxation to the capacitated facility location problem, but refers to Geoffrion for a discussion of the theory. Nauss uses the concept of projection over the location variable space, a concept that is crucial to the solution of our model. Once again a good branch-and-bound algorithm is detailed, and Nauss is a proponent of LIFO (last-in, first-out) branching techniques for this problem subclass. This is the route chosen for solution of our problem. The author does not pay much attention to branching procedures, but the test results are impressive.

Sherali and Adams [1984] detail a solution for the discrete location-allocation problem via a special linearized reformulation. Section 2 of this work contains the basis for the Generalized Benders' cuts that are used in the implicit enumeration scheme of the model solution procedure presented in the next chapter. The alternation between a master program and subproblem, with the most "violated" constraint creating a Benders' cut each iteration, is shown to have performed favorably compared to existing Lagrangian techniques. The point is made that potential for "complicating" side constraints can necessitate tailoring an algorithm to yield an efficient solution procedure.

Van Roy [1986] attempts to unify the Benders' cut approach with the Lagrangian relaxation approach into a "cross decomposition" algorithm. The strength of this algorithm lies
in its coalescing both primal and dual cuts into a dual master program that attempts to restrict dual multiplier values to a smaller set than possible using either Benders' cuts or Lagrangian relaxation alone. A LIFO branch-and-bound strategy and a simple branching variable selection strategy round out the paper, along with computational results for medium-sized test problems.

Finally Cornuejols et al. [1988] have a comparison paper that surveys various heuristic approaches taken by authors, in an attempt to draw generalities and similarities from among the literature available on the capacitated plant location problem. Various relaxation strategies are ranked, and the authors provide some classical inequalities, along with computational experience in implementing heuristics. As a survey and comparison paper, the work brings together most of the methodology described in this chapter. Still, the overall emphasis has been on linear objective functions and little or no side constraints. The problem at hand contains special side constraints in the location space only, has a highly nonlinear objective function that simulates total system traffic time, and seems to be as yet untreated in the literature.

Practical literature reviewed for a background and the basics of the problem and the solution algorithm in this effort includes the works mentioned above, plus a review of Bazaraa and Jarvis [1977], Bazaraa and Shetty [1979], Fisher
[1981], Geoffrion [1972], and the previous programs and theory presented by Dr. Hobeika under the TEDSS auspices: Hobeika [1985], Hobeika and Sherali [1987]. All of these individual efforts were united into a unique solution for a problem with very interesting features.
Chapter IV
Model Solution

This chapter contains a heuristic approach, as well as two exact algorithms directed at solving the mixed integer math program presented in Chapter II. As was pointed out earlier, the exact algorithms are likely to be computationally intensive due to their NP-hardness (see Garey and Johnson, [1979]). Very important to the verification of the model, however, is the obtainment of optimal solutions: it is by knowing exact answers that one may judge how well various heuristic schemes operate, and perform meaningful sensitivity analyses based on data perturbation.

Both exact algorithms employ Geoffrion's [1972] Generalized Benders' Algorithm as the optimizing tool. Benders' cuts were chosen because of their ability, in this case, to help exploit certain structures inherent to the problem class at hand. The heuristic routine also exploits the special problem structure by developing a strongest surrogate constraint: this is a tight lower bounding linear form for the objective function.

For convenience, recall the problem to be solved as

\[ P: \text{Minimize} \quad \sum_{(p,q) \in A} [t_{pq}x_{pq} + \tau_{pq}x_{pq}(x_{pq} + \bar{x}_{pq})^\beta] + \sum_{i \in N} M_i x_{ai} \]  

Subject to:

\[ \sum_{q \in F(i)} x_{iq} - \sum_{p \in R(i)} x_{pi} + x_{ai} = D_i - z_i, \quad i \in N_d \]  

\[ 0 \leq z_i \leq C_i y_i, \quad \forall i \in N_d \]  

\[ 0 \leq x_{pq} \leq u_{pq}, \quad x_{ai} \geq 0, \quad \forall i \in N \]  

\[ y \in Y \]  

24
Several modifications merit discussion. First, artificial arcs, one for each node, are included in the flow balance equations. Their inclusion provides for a feasible starting solution in every instance, while the large coefficient vector $M$ serves to penalize positive values taken on by these variables. Notice that the $z_i$ variables exist only for terminal nodes, and that all variables carry a nonnegativity restriction. Problem $P$ may be written more compactly in vector form as

$$\begin{align*}
P: \quad & \text{Minimize} & \quad & [F(x) + M \cdot x_a] \\
& \text{Subject to:} & \quad & Ax + Bz + x_a = D \\
& & \quad & 0 \leq z_i \leq C_i y_i, \quad \forall i \in N_d \\
& & \quad & 0 \leq x \leq u', \quad x_a \geq 0. \quad y \in Y.
\end{align*}$$

In the above representation, $B$ is a rectangular matrix composed of an identity matrix of dimension equal to the cardinality of the terminal node set, appended above a zero matrix to match dimensions with $A$. Several constraints have been conglomerated to form (34). Now if problem $P$ is projected onto the $y$-variable space, the program above may be equivalently rewritten as follows. Define

$$\begin{align*}
f(y) \equiv & \quad \text{minimum} & \quad & [F(x) + M \cdot x_a] \\
& \text{Subject to:} & \quad & Ax + Bz + x_a = D \\
& & \quad & 0 \leq z_i \leq C_i y_i, \quad \forall i \in N_d \\
& & \quad & 0 \leq x \leq u', \quad x_a \geq 0.
\end{align*}$$
for any fixed value of \( y \) in \( Y \). Note that because of the presence of the variables \( x_a \), \( f(y) \) is finite for all \( y \) in \( Y \).

It is imperative to notice that (35)-(38) is a separable, convex cost, nonlinear objective, network flow programming problem. Ultimately the projection of problem \( P \) onto the \( y \)-variable subspace takes the form

\[
P: \text{Minimize } \{f(y): y \in Y\}.
\]  

(39)

Since the problem (35)-(38) is convex with linear constraint functions, nonlinear programming duality theory asserts that, for this case,

\[
f(y) = \max_{\pi \geq 0} \{\nu(\pi) - \sum_{i \in N_d} \pi_i C_i y_i\},
\]  

(40)

where \( \pi \) is the Lagrange multiplier vector associated with the constraint set (37), and where

\[
\nu(\pi) = \min \{F(x) + M \cdot x_a + \sum_{i \in N_d} \pi_i z_i\}
\]  

subject to: \( Ax + Bz + x_a = D \)  

(42)

\[
0 \leq z_i \leq C_i, \quad \forall i \in N_d
\]  

(43)

\[
0 \leq x \leq u', \quad x_a \geq 0.
\]  

(44)

Observe that the placement of the dual multiplier products in the objective function has uncoupled the \( z \)- and \( y \)-constraints from each other in (43). When (39) is combined with (40) and (42)-(44), an alternate representation of \( P \) is

\[
P: \text{Minimize } \gamma
\]  

subject to:

\[
\gamma \geq \nu(\pi) - \sum_{i \in N_d} \pi_i C_i y_i, \quad \forall \pi \geq 0
\]  

\( y \in Y \).

(47)
where $\nu$ is finite and is given by (43) for any $\pi \geq 0$.

The proposed heuristic exploits the special problem structure by generating a strongest surrogate constraint. Let $\bar{P}$ represent a continuous relaxation of the integer restrictions on $P$. Hence in $\bar{P}$, $y$ is a continuous variable on the closed interval $[0,1]$, and $\bar{Y}$ is as in (3) with the binary restriction removed. The first step of the heuristic is to solve this continuous relaxation $\bar{P}$ using the software package MINOS 5.1. The all-artificial basis is initially set up and MINOS works toward a feasible optimal solution. Let the optimal Lagrangian multipliers associated with the constraint set $-z_i \geq -C_i Y_i$, $i \in N_d$ in (33) be represented by the vector $\pi$, and let $\bar{v}$ represent the optimal objective function value. Also, denote the primal optimal solution to $\bar{P}$ as $(\bar{x}, \bar{y}, X_a, Y)$. The primal-dual solution taken as a complete set satisfies the Karush-Kuhn-Tucker necessary conditions for optimality (41)-(44) (see Bazaraa and Shetty, [1979]), if the dual multipliers $\pi$ are left as $\bar{\pi}$, and if the duals to the redundant inequality constraints (43) are set to zero. Now

$$\bar{v} = f(\bar{y})$$

and

$$\nu(\bar{\pi}) = \bar{v} + \sum_{i \in N_d} \bar{\pi}_i \bar{y}_i \equiv \bar{v} + \sum_{i \in N_d} \bar{\pi}_i C_i \bar{y}_i$$

(49)

Thus the strongest surrogate constraint

$$\gamma \geq \nu(\bar{\pi}) - \sum_{i \in N_d} \bar{\pi}_i C_i y_i$$

(50)
can be developed, and is valid, with the constraint representing a lower bounding linear form for \( P \). The name comes from the fact that the linear program

\[
\text{minimize}_{y \in \mathcal{Y}} \left[ \nu(\bar{\pi}) - \sum_{i \in \mathcal{N}_d} \bar{\pi}_i C_i y_i \right]
\] (51)

has the solution vector \( \bar{y} \), due to satisfaction of the Karush-Kuhn-Tucker necessary and sufficient conditions by the primal-dual solution set to \( \bar{P} \). Thus, with the single constraint (50) and the \( y \) in \( \bar{Y} \) conditions, the program (45)-(47) recovers the optimum value for \( \bar{P} \).

The proposed heuristic operates as follows.

**Option 1: Heuristic**

**Step 1:**

Solve the nonlinear convex program \( \bar{P} \) to obtain a solution \( \bar{Y} \), and generate the strongest surrogate constraint as described above. If \( \bar{Y} \) is a binary vector, terminate with this as the prescribed solution \( \hat{y} \), since \( \bar{P} \)'s solution, due to its dominance over \( P \), is co-optimal for \( P \). Else, there must exist at least one fractional \( \bar{Y}_i \)-variable value. Proceed to Step 2.

**Step 2:**

Construct the solution vector \( \hat{y} \), according to \( \hat{y}_i = |\bar{y}_i| \) \( \forall_i \). The characteristics of the constraint set (3) guarantee that \( \hat{y} \) is an element of the set \( Y \). Obviously, the solution to \( P \) can only improve with an increase in any \( \hat{y}_i \): this is the logical basis for the heuristic scheme. Note that once a \( \hat{y}_i \).
is fixed at one, it remains so until termination of the heuristic. This fact, along with the fact that there are only a finite number of terminal nodes, guarantees an exit from the heuristic procedure in a finite number of steps.

Step 3:

Divide the indices \( y_j, j \) in \( N_d \), of variables which are not unity in \( y \) into two subsets of \( N_d \). Let \( N_1 = \{ j \in N_d : \bar{y}_j \text{ fractional} \} \), and \( N_2 = \{ j \in N_d : \bar{y}_j = 0 \} \). Take the constraints

\[
-\sum_i s_{ki} y_i \geq -s_k, \quad k=1,\ldots,m
\]

(52)

defining \( Y \), examine the optimal dual multipliers

\[
\bar{\theta}_k \geq 0, \quad k=1,\ldots,m
\]

(53)
say, associated with these in the solution of \( \bar{F} \), and construct a surrogate constraint

\[
\sum_i \alpha_i y_i \leq \alpha_0,
\]

(54)

where

\[
\alpha_i = (\sum_k \bar{\theta}_k s_{ki}), \quad \forall i \in N_d, \quad \text{and} \quad \alpha_0 = (\sum_k \bar{\theta}_k s_k).
\]

(55)

Next, compute a ratio

\[
\frac{\bar{C}_i}{\alpha_i}, \quad \forall i \in \{N_1 \cup N_2\}.
\]

(56)

Let the ratio in (56) equal "infinity" if \( \alpha_i \) is zero. Create a candidate list \( L \) of all indices in the union of sets \( N_1 \) and \( N_2 \), and arrange the indices in \( N_1 \) in decreasing order of the
ratio (56), followed by the indices from $N_2$ ordered similarly. Test indices, one by one, to see if by fixing $\hat{y}_i = 1$, the new $\hat{y}$ vector can remain in $Y$. Round up to $\mu$ $y$-variables, $\mu$ being a frequency parameter determining how often to recompute the strongest surrogate constraint.

**Step 4:**

If less than $\mu$ variables can be rounded up at this step, continue to Step 5. Otherwise, fix the variables $y_i$ which are unity in $\hat{y}$ at 1 in Problem $\tilde{P}$, and resolve this problem to derive a revised solution $\tilde{y}$. If this new $\tilde{y}$ is all integer, terminate with this as the prescribed solution $\hat{y}$. Otherwise define the vector $\hat{y}$ according to $\hat{y}_i = \lfloor y_i \rfloor \forall i$, and return to Step 3.

**Step 5:**

Compute $f(\hat{y})$ via (35)-(38) and terminate.

This is the proposed heuristic. Note that once the $y$-variables are fixed at 1, they never become unfixed again. In addition, Step 5 updates $f(\hat{y})$ a final time, incorporating all $\hat{y}_i$ that were set to 1 in the final trip to Step 3.

The parameter $\mu$ can be varied to speed up solution time, as well as to possibly shuffle or repermute unfixed indices. A setting of $\mu = 1$ returns control to MINOS after each variable is fixed. The program then computes a new objective value and attempts to round up any remaining unfixed variables. The heuristic is known as Option 1 in the program, and the exact
algorithmic solution routines are known as Option 2 and Option 3.

Option 2 is the first of the two implicit enumeration techniques designed to terminate with an optimal solution. This option alternates between the problem

$$\text{Minimize } \{f(y): y \in \mathcal{Y}\}$$  \hspace{1cm} (57)

where

$$f(y) = \max_{\pi \geq 0} \{\nu(\pi) - \sum_{i \in \mathcal{N}_d} \pi_i C_i y_i\} \geq$$

$$[\nu(\pi^q) - \sum_{i \in \mathcal{N}_d} \pi^q_i C_i y_i], \quad \forall \pi^q \geq 0, \text{ where } q = 1, \ldots, Q, \hspace{1cm} (58)$$

and a linear programming cut-generating subproblem. The index Q represents successive iterations through MINOS, in which sets of dual multipliers $\pi^q$ corresponding to the constraint set (33) are explicitly enumerated and used to develop constraints on the objective function. The algorithm, as stated previously, utilizes a LIFO depth-first enumeration strategy in which the branch and bound algorithm proceeds along a path of the enumeration tree until fathoming occurs, whence it backtracks to the most recently created active node. A LIFO method affords a straightforward bookkeeping procedure, and the enumeration tree is easily updated. The LIFO scheme should be coordinated with a heuristic for producing good quality solutions early in the enumeration process, to hopefully allow for more node fathoming, and hence, lesser enumeration.
Option 2 consists of an initialization procedure and four main processing steps.

**Option 2: First Algorithmic Solution Routine**

**Initialization:**

Begin by generating the heuristic solution and the strongest surrogate constraint, which are the terminal outputs from Option 1. The solution \( y^* = \hat{y} \) is the incumbent solution vector, and

\[
\nu^* = f(\hat{y}).
\]  

(59)

is the incumbent objective value. The dual solution vector

\[
\pi = \pi^1
\]  

(60)

is saved as the first dual multiplier vector and the index \( Q \) is initialized at 1. The partial solution vector \( PS \) is initialized as empty since no branching or setting of variables has occurred yet. Three sets that merit definition now are:

\[
I^+ = \{i \in N_d: y_i = 1 \text{ in } PS\}
\]

\[
I^- = \{i \in N_d: y_i = 0 \text{ in } PS\}
\]

\[
I' = N_d - (I^+ \cup I^-),
\]

with

\[
Y(PS) = Y \cap \{y: y_i = 1, i \in I^+; y_i = 0, i \in I^-\}.
\]  

(62)

Denote the continuous relaxation of \( Y(PS) \) as \( \overline{Y}(PS) \).
**Step 1 - Node Analysis:**

The node analysis step is a combination of three logical tests which attempt to first fathom the node under consideration, or if that is unsuccessful, to logically set some $y$-variables to values that they must take in order for the current node to remain active.

**Logical Test 1:** If

$$
\gamma_i = [\nu(q) - \sum_{i \in I^{+}} x_i^q c_i] \geq \nu^*
$$

(63)

for any $q \in [1, \ldots, Q]$, then the node can be fathomed by proceeding to Step 4. This follows since allowing all free variables to take on their maximum value will not push the strongest surrogate constraint value below the incumbent value. Hence, it is impossible for the node to produce a better solution.

**Logical Test 2:** Logical test 2 is known as the cancellation-one test. It proceeds as follows. If

$$
\gamma_i + x_i^q c_k \geq \nu^*,
$$

(64)

for any $k$ in $I^f$ and for any $q \in [1, \ldots, Q]$, then we set $y_k = 1$ and increment the partial solution vector by "soft" $+k$. The logic behind the cancellation-one test is that without $y_k$ set to one, there is no way that the lower bound of the type (63) available from the strongest surrogate constraint can remain less than the current incumbent value. Thus this variable must take on the value of 1. A "soft" variable fixation, as
opposed to a "hard" variable fixation, simply means that the variable represented by that index will become a free variable once again when (if) the node is fathomed or becomes inactive. A hard-fixed variable, in contrast, is one that is fixed by the branching process, and does not become free once the node resulting from the corresponding branch is fathomed. Rather, the variable in question is complemented, and is then denoted as a soft-fixed variable for the parent node of that branch. Logical Test 3: The cancellation-zero test operates with the constraints \( y \) in \( Y \) by asserting that if setting \( y_k = 1 \) for some \( k \) \( \in \Gamma \), we can detect that

\[
Y(PS) \cap \{ y_k = 1 \} = \emptyset, \tag{65}
\]

then we can fix \( y_k = 0 \) and increment the partial solution vector by a soft "—k". This set is detected logically by testing if the fixing of a variable \( y_k = 1 \) violates some constraint of the set \( Y \) with other free variables equal to 0. If so, then putting \( y_k = 1 \) obviously makes the problem infeasible and the program is justified in fixing that particular variable \( y_k \) to 0.

Note that if an infeasibility in \( Y(PS) \) is detected by the logical tests on its constraints, perhaps by virtue of some variables being fixed at 1 by Logical Test 2, then the node is fathomed by proceeding to Step 4. It should also be noted here that, as the logical tests are done in a serial fashion, the potential ramifications of the variable-fixing and
fathoming procedures might have an effect on previously executed tests. It is for this reason that, if action is taken by any of the tests, all tests are cyclically repeated. The procedure continues until no variables are fixed in a complete run of all logical tests, or until the current node is fathomed.

**Step 2 - Lower and Upper Bound Computations**

Following the logical tests described above, it is possible and probable that nodes which cannot improve upon the incumbent solution will survive Step 1. To this end, a linear program

\[
\bar{\nu} = \nu(\bar{x}) - \max \left\{ \sum_{i \in \tilde{N}_d} \bar{\pi}_i C_{iy} : y \in \tilde{Y}(PS) \right\}
\]

is solved. Let \( \bar{y} \) denote the optimal solution obtained. It should be clear that the solution to the linear program above, because of the relaxed constraint set, provides a lower bound for the current node's value. Hence, if

\[
\bar{\nu} \geq \nu^*,
\]

the node is fathomed by proceeding to Step 4. If the value

\[
\bar{\nu} < \nu^*,
\]

but \( \bar{y} \) contains at least one fractional component, execution proceeds to Step 3. However, if the solution vector \( \bar{y} \) is integral, the program returns to MINOS 5.1 to compute \( f(\bar{y}) \). Then, if
In any case, the evaluation of solution $f(\tilde{y})$ via (35-38) yields a new set of dual multipliers $\bar{\pi}$. These values are retrieved and another constraint of the type (46) with $\pi = \bar{\pi} = \pi^{Q+1}$ is generated and stored for use in the logical tests at Step 1. Let $Q_{\text{max}}$ be the maximum allowable number of constraints that the programmer is keeping; then if $Q_{\text{max}}$ constraints already exist, the new constraint replaces the constraint generated from the $y$-solution with the highest objective value with respect to $f(\cdot)$ (except for the strongest surrogate constraint, which is never replaced), thus keeping the $Q_{\text{max}}$ "tightest" constraints active. Otherwise, we add this constraint and increment $Q$ by 1. The last substep inside Step 2 utilizes the obtained integer solution $\hat{y} \in Y$ to possibly update the incumbent solution, which, if successful, should reduce node analysis time. If there are no more fractional $y$-variables, the node is fathomed by proceeding to Step 4. Otherwise, we continue to Step 3.

**Step 3 - Branching Variable Selection**

Calculate, for all free variables, $\phi_i = \text{fractionality of } \bar{y}_i = \min [\hat{y}_i, (1-\hat{y}_i)]$, where $\hat{y}$ is the solution to (66) at Step 2, and determine
Increment the partial solution vector PS by a hard +k. The first term in the lexicographic argument set can be visualized as the index's potential for affecting the strongest surrogate constraint based lower bound. The second and third terms provide a tiebreaking procedure according to a similar discriminating prioritization. Return to Step 1.

**Step 4 - Fathoming:**

Find the most recent (right-most) hard element in the partial solution vector PS. If there are none, stop: enumeration is complete. Else, complement the element (turning a zero into a one or vice versa), make it soft, and delete all entries to the right. Release the deleted indices to the free index list for future fixing/branching. Return to Step 1.

Option 3, the second of the implicit enumeration schemes proposed for solution of the problem, is similar to Option 2. The central difference is that Option 3 requires two LP relaxations to be solved in every subproblem iteration, taking more computational time, but potentially further constraining the problem. The algorithm follows.

**Option 3: Second Algorithmic Solution Routine**

**Initialization:**

Proceed in the same manner as with Option 2.
Step 1 - Node Analysis:

Proceed in the same manner as with Option 2.

Step 2 - Lower and Upper Bound Computations:

As in Option 2, solve the linear program (66). If (67) is true, then fathom the node by proceeding to Step 4. If (67) is false, determine \( f(\tilde{y}) \) through MINOS 5.1, whether \( \tilde{y} \) is integral or fractional. Denote the optimal dual variable vector \( \pi \) associated with

\[
-z_i \geq -C_i \tilde{y}_i, \quad i \in N_d, \quad \text{as } \pi, \quad (72)
\]

and let

\[
\tilde{v} = f(\tilde{y}). \quad (73)
\]

Compute

\[
\nu(\pi) = \tilde{v} + \sum_{i \in \tilde{N}_d} \pi_i C_i \tilde{y}_i. \quad (74)
\]

and construct the valid constraint

\[
\gamma \geq \nu(\pi) - \sum_{i \in \tilde{N}_d} \pi_i C_i y_i. \quad (75)
\]

Additionally, if the vector \( \tilde{y} \) is binary, and if \( f(\tilde{y}) < \nu^* \) then set

\[
\nu^* = \tilde{v} \quad \text{and} \quad y^* = \tilde{y}. \quad (76)
\]
In any case, next solve the following linear program with the two Generalized Benders' cuts as shown below:

Minimize \( \gamma \) \hspace{1cm} (77)

Subject to: \[ \gamma \geq \nu(\pi) - \sum_{i \in N_d} \bar{\pi}_i C_i y_i \] \hspace{1cm} (78)

\[ \gamma \geq \nu(\bar{\pi}) - \sum_{i \in \bar{N}_d} \bar{\pi}_i C_i y_i \] \hspace{1cm} (79)

\[ y \in Y(PS). \] \hspace{1cm} (80)

and denote the optimal solution as \((\hat{\gamma}, \hat{y})\). \hspace{1cm} (81)

If \( \hat{y} \) is binary, recompute \( f(\hat{y}) \) and update the incumbent solution, if necessary. If \( \hat{\gamma} \geq \nu^* \) then fathom the node by proceeding to Step 4. Otherwise, denote the optimal dual multiplier obtained from (78) as \( \hat{\lambda} \geq 0 \). \hspace{1cm} (82)

and observe by duality that \( (1 - \hat{\lambda}) \geq 0 \) \hspace{1cm} (83)

is the dual multiplier associated with (79). If \( Q < Q_{\text{max}} \), store the cut generated in (75) denoting \( \pi^{Q+1} = \bar{\pi} \), and increment \( Q \) by 1. Otherwise replace the cut with the largest objective value \( f(\hat{y}) \) with the new cut (75), but again leaving the strongest surrogate constraint as is. Proceed to Step 3.

Some observations merit discussion at this point. Note that Option 3 digresses from Option 2 first by mandating a return to MINOS 5.1 and by using the output to develop a
second cut (75) to go along with the Benders' cut (49). Simply put, Option 3 takes advantage of the updated \( \gamma \)-vector \( \hat{\gamma} \) to produce a new set of optimal dual multipliers which can be used, in a Lagrangian fashion, to bound \( \gamma \) from below. Note that \( \hat{\gamma} \geq \hat{\nu} \) obtained in (66), thereby providing a tighter lower bound than in Option 2. While the production of a second cut and the generation of a tighter lower bound takes effort in a computational sense, the possibility for payoff, in the form of a node (or several) being fathomed, is great. Secondly, the use of the surrogate constraint

\[
\gamma \geq \hat{\nu} - \sum_{i \in N_d} \hat{\nu} C_i y_i, \quad (84)
\]

where

\[
\hat{\nu} = \hat{\lambda} \nu(\pi) + (1 - \hat{\lambda}) \nu(\overline{\pi}) \quad (85)
\]

and where

\[
\hat{\pi} = \hat{\lambda} \overline{\pi} + (1 - \hat{\lambda}) \pi \quad (86)
\]

becomes useful for the branching variable selection strategy described next.

**Step 3 - Branching Variable Selection**

As in Option 2, if there are no more free variables upon which to branch, fathom the node by proceeding to Step 4. Otherwise, if the solution \( \hat{\gamma} \) found from solving (77-80) is binary, then determine

\[
k = \arg\max_{i \in \mathcal{I}} \{ \hat{\pi}_i, C_i \} . \quad (87)
\]
where $\hat{x}$ is defined in (86).

Increment the partial solution vector by a hard +k, and return to Step 1. For the case in which $\hat{y}$ is fractional, let $\phi_i$ represent the fractionality of $\hat{y}_i$ in $\hat{y}$ as before, and determine

$$k = \arg\max_{i \in I', (\phi_i > 0)} \{\phi_i C_i \hat{x}_i\}.$$  \hspace{1cm} (88)

Increment PS by a hard +k and return to Step 1.

**Step 4 - Fathoming**

Proceed in the same manner as with Option 2.

Intuitively, both the exact options and the heuristic option take advantage of both the special problem structure and computational information from previous steps. Note that for a termination of the exact algorithms, all possible branches of the enumeration tree must be either explored or justifiably fathomed. The result is an insured optimal solution.
Chapter V
Computational Experience & Results

This chapter contains the results of the trial runs of the programs MAKEMPS and RMIP that were developed as a part of our research effort. The chapter also includes a brief description of the features of TEDSS, the transportation evacuation decision support system which will eventually incorporate many of the routines that are developed in this research.

As was stated earlier, TEDSS is a real-time computer decision support system which was developed to aid emergency managers in the decision making process concerning evacuation planning for natural or man-made disasters. TEDSS is comprised of four separate modules: the system control module, the database module, the simulation model module, and the graphic display module. Input includes area demographics, the highway network topology and traffic control strategies, the safe areas to which endangered people should be evacuated, and finally, the vehicle loading curves which represent the anticipated public response to an evacuation order. The output from a TEDSS run includes an origin-destination trip table, selected evacuation paths, congested links (arcs) and network performance statistics, and the total network clearance time.

TEDSS is written in FORTRAN77 like MAKEMPS and RMIP, except for the graphic display module, which was coded in
BASIC. Minimum hardware requirements for TEDSS include an IBM-PC or compatible with 640k of RAM, and an 8087 or newer microprocessor. The market for TEDSS has been envisioned as the local or city government which cannot afford a very large computer system. TEDSS, like RMIP, uses a mathematical analytical model to estimate potential evacuation times under emergency conditions. The time is then employed as the driving force behind the model's simulation of a traffic assignment pattern to a given highway network. The advantages of using a decision support tool such as TEDSS are many, including the ability to estimate the proper time to issue evacuation orders, judicious allocation of emergency response personnel, and the deployment of emergency supplies or other resources to the most needed areas.

The third module of TEDSS is the simulation model, and it is here that the results from our research effort will eventually be incorporated. The current version of TEDSS contains three distinct simulation models; namely, the MACROVAC model, the MASSVAC2 model, and the HEUPRAE model. The MACROVAC model is essentially a quick-response tool used to generate the required evacuation times, and it does so without employing any complicated simulation or data manipulation routines. The MASSVAC2 model is the planning tool used to investigate potential evacuation strategies in order to develop a total evacuation plan for the municipality. Finally, the HEUPRAE model operates as the real-time
simulation tool used for altering the master evacuation plan as more disaster-related information becomes available to the decision maker. It is the second submodule that the findings contained in this thesis will impact the most.

Four techniques are employed in the current MASSVAC2 submodule. They include the shortest-path algorithm, the traffic assignment algorithm, the vehicle discharge method, and disaster response factors. The shortest path algorithm used by TEDSS is Dijkstra's algorithm. Dijkstra's algorithm is quite simple; the basic operation is described as $D(k,j) = \min[D(K,i) + D(i,j)]$, where

$$D(k,j) = \text{length of the shortest path from node } k \text{ to node } j,$$
$$D(k,i) = \text{length of the shortest path from node } k \text{ to node } i,$$
$$D(i,j) = \text{length of the link from node } i \text{ to node } j.$$

By suitably iterating this equation through various link-path structures, a solution path can be efficiently determined.

The traffic assignment algorithm in TEDSS is Dial's probabilistic multipath traffic assignment algorithm. Dial's method is a two-pass Markovian process where probabilities of trips using certain network links are calculated in the first pass, and the traffic assignment is conducted in the second pass. It is necessary to know four facts about each node $i$ in order to assign traffic between an origin node $O$ and destination $D$:

$p(i) = \text{the shortest path from } O \text{ to } i,$
$q(i) = \text{the shortest path from } i \text{ to } D,$
I(i) = the set of all links (arcs) with initial node i, 
Fr(i) = the set of all links (arcs) with terminal node i. 
With this information available, trips can then be assigned 
with a backward-pass method to complete the algorithm. 

The third technique is the vehicle discharge method. Two 
separate methods are employed, depending on the category of 
the highway. For macroscopic simulation, the Laporte linear 
regression model modified by Kalevala is employed to determine 
flow on expressways. The method first finds the density of 
the link then it uses the equation Flow(Q) = Density(K) * 
Speed(U) to determine the vehicle discharge. If the 
intersections contain a signal then the equation DA = 
[V1/(V1+V2)] * 1800 is used, where 
DA = discharge volume per lane on major road 1, 
V1 = critical lane volume at major road 1, 
V2 = critical lane volume at minor road 2. 
At a microscopic level the technique is identical except for 
two- and four-way stop sign intersections: the gap delay 
technique is used to determine the number of vehicles to be 
discharged at a two-way stop, while a maximum of 900 vehicles 
per hour are allowed through a four-way stop. 

Lastly, disaster response factors, such as the panic 
factor, suitability factor, and the loading factor are used 
as risk factors in an attempt to model evacuee response 
behavior. As the time for an evacuation order draws near, the 
panic factor is used to eliminate certain links on the
network. The suitability factor is a weight used to model the nearness of the disaster to the affected link (arc). The loading factor captures the accessibility of links (arcs) to shelters.

All four of the abovementioned techniques are useful strategies for simulating the difficult problem of a mass evacuation from a populous area. RMIP is a different approach than the other four techniques with its emphasis on primal-dual optimization techniques and its variety of options within the optimization method. The following are tables reporting the computational experience of several trial applications on the real data used to calculate evacuation times for the City of Virginia Beach, Virginia. The particular region that was used is designated Region 2 in the TEDSS studies and in Hobeika and Sherali [1987]. This area is commonly known as the Princess Anne District.

To adequately test the three options of RMIP, six test problems were run first. Results of this test were used to choose the option that was used for the second phase of code testing. In the second phase of testing, a single selected option was run against several specialized scenarios in order to test the robustness of the code.

The first phase of testing began by running the heuristic known as Option 1 against the six phase 1 test problems. The phase 1 test problems employed 5, 10, and 15 available shelter locations and tested each number of shelter locations against
2 sets of shelter capacities, namely 100 and 900. The region studied consisted of a total of 15 available destination nodes, 9 origin nodes, with 74 total nodes, and with 118 arcs. While some data was available from previous demographic data collection, other information had to be developed. The actual network structure, consisting of the arcs and nodes, is a real region of Virginia Beach. The arc lengths are real as is the rest of the arc data. Total supply for the network was modeled at 75% of capacity for problems 1, 3, and 5, and at 60% of capacity for problems 2, 4, and 6. However, node capacities were not available and had to be estimated. Similarly, three constraints were constructed to represent the set $Y$. These constraints were visualized as regional, subregional, and staffing type constraints. For the regional constraint, the inequality developed was of the form $\sum_{i \in N_d} y_i \leq W$, with $W=4$ for problems 1 and 2, $W=9$ for problem 3, $W=8$ for problem 4, $W=12$ for problem 5, and $W=11$ for problem 6. The subregional constraint was developed due to the large concentration of potential shelter locations in the south central area of Region 2. Potential shelters 10 through 14 are very close to each other, so a constraint $y_{10} + y_{11} + y_{12} + y_{13} + y_{14} \leq 3$ was included. Finally, a staffing constraint was created to model the scarcity of some particular resource in the central area of Region 2. The constraint was modeled as $25y_4 + 15y_6 + 20y_7 + 5y_8 \leq 40$. In addition, the superimposed flow parameter $\bar{x}$ was taken as 15% of the maximum arc flow for
each arc \((p,q)\) in \(A\). This number is often used for preloading conditions by TEDSS researchers. Finally, the parameter \(\mu\) was varied between 1, 5, and 10 to actually create 18 runs. The abbreviations that merit definition for Table 1 are:

- NO. DEST. = Number of potential destinations for evacuees;
- CAP. = Capacity of each destination node;
- NO. ITER. = Number of MINOS 5.1 executions;
- CPU = CPU time, in seconds, on the IBM 3090 mainframe;
- NO. SHELT. = Number of shelters opened from ones available;
- OBJ. = Percentage of objective value determined from Options 1 and 2.

The results follow in Table 1.
### TABLE I

HEURISTIC PERFORMANCE AGAINST SIX TEST PROBLEMS

<table>
<thead>
<tr>
<th>PROB. NO.</th>
<th>FREQ.</th>
<th>DEST.</th>
<th>CAP.</th>
<th>ITER.</th>
<th>CPU</th>
<th>SHELT.</th>
<th>OBJ.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>5</td>
<td>100</td>
<td>2</td>
<td>0.84</td>
<td>4</td>
<td>100</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>5</td>
<td>100</td>
<td>1</td>
<td>0.76</td>
<td>4</td>
<td>100</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td>5</td>
<td>100</td>
<td>1</td>
<td>0.76</td>
<td>4</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>5</td>
<td>900</td>
<td>3</td>
<td>1.07</td>
<td>4</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>5</td>
<td>900</td>
<td>1</td>
<td>0.96</td>
<td>4</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>5</td>
<td>900</td>
<td>1</td>
<td>0.96</td>
<td>4</td>
<td>100</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>10</td>
<td>100</td>
<td>3</td>
<td>0.89</td>
<td>9</td>
<td>102</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>10</td>
<td>100</td>
<td>1</td>
<td>0.80</td>
<td>9</td>
<td>102</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>10</td>
<td>100</td>
<td>1</td>
<td>0.80</td>
<td>9</td>
<td>102</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>10</td>
<td>900</td>
<td>4</td>
<td>1.17</td>
<td>8</td>
<td>103</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>10</td>
<td>900</td>
<td>1</td>
<td>1.02</td>
<td>8</td>
<td>103</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>10</td>
<td>900</td>
<td>1</td>
<td>1.02</td>
<td>8</td>
<td>103</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>15</td>
<td>100</td>
<td>4</td>
<td>1.14</td>
<td>12</td>
<td>104</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>15</td>
<td>100</td>
<td>1</td>
<td>1.05</td>
<td>12</td>
<td>104</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>15</td>
<td>100</td>
<td>1</td>
<td>1.05</td>
<td>12</td>
<td>104</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>15</td>
<td>900</td>
<td>5</td>
<td>1.48</td>
<td>11</td>
<td>106</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>15</td>
<td>900</td>
<td>2</td>
<td>1.26</td>
<td>11</td>
<td>106</td>
</tr>
<tr>
<td>6</td>
<td>10</td>
<td>15</td>
<td>900</td>
<td>1</td>
<td>1.21</td>
<td>11</td>
<td>106</td>
</tr>
</tbody>
</table>
As the test runs show, the problem is fairly insensitive to variations in the frequency parameter \( \mu \). It is hypothesized that with larger problems, an increase in the number of binary variables would in turn make \( \mu \) a more valuable parameter.

In the next test, the two implicit enumeration algorithms Option 2 and Option 3 were run against the same six problems. Note that since both implicit enumeration algorithms generate the heuristic solution produced by Option 1 as their starting feasible solution, the implicit enumeration algorithms must necessarily run at least as long as the heuristic routine. It is our experience that sometimes the heuristic itself produced an optimal solution and that the implicit enumeration algorithms merely verified optimality.

Certain parameters were held constant during the test runs. Since no apparent benefit was gained by increasing \( \mu \) from 5 to 10, \( \mu \) was held steady at 5, thus recomputing the continuous relaxation solution after every five successful binary variable roundups. The number of arcs and number of nodes for the transshipment portion of the network were not varied: the different problems were generated by the deletion (or addition) of potential shelter locations. In order to provide a moderately dense network, thus increasing congestion, the total number of system users remained the same as in the heuristic test. The \( y \) in \( Y \) constraints used were
the same as for the first set of test problems, and a value of 10 for $Q_{\text{max}}$ was employed throughout all test runs.

Abbreviations are:

OPT.PRO = Option and problem number (from Table 1);

NODES = Number of enumeration tree nodes generated in the program execution;

F.E. = Fathoming efficiency, a ratio of the number of visits to the fathoming step divided by the number of nodes generated;

N.GEN. = Number of enumeration tree nodes generated by the time the optimal solution is determined;

NO.CUTS = Number of Generalized Benders' cuts generated in the program execution.

Results are printed in Table II.
<table>
<thead>
<tr>
<th>OPT. NO.</th>
<th>PRO</th>
<th>DEST</th>
<th>CAP.</th>
<th>NODES</th>
<th>F.E.</th>
<th>N.GEN.</th>
<th>SHELT.</th>
<th>CUTS</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>5</td>
<td>100</td>
<td>1(^1)</td>
<td>1.00</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>0.80</td>
<td></td>
</tr>
<tr>
<td>3.1</td>
<td>5</td>
<td>100</td>
<td>1(^1)</td>
<td>1.00</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>0.80</td>
<td></td>
</tr>
<tr>
<td>2.2</td>
<td>5</td>
<td>900</td>
<td>1(^1)</td>
<td>1.00</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>1.04</td>
<td></td>
</tr>
<tr>
<td>3.2</td>
<td>5</td>
<td>900</td>
<td>1(^1)</td>
<td>1.00</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>1.04</td>
<td></td>
</tr>
<tr>
<td>2.3</td>
<td>10</td>
<td>100</td>
<td>4</td>
<td>0.50</td>
<td>2</td>
<td>8</td>
<td>4</td>
<td>2.59</td>
<td></td>
</tr>
<tr>
<td>3.3</td>
<td>10</td>
<td>100</td>
<td>4</td>
<td>0.50</td>
<td>2</td>
<td>8</td>
<td>7</td>
<td>2.84</td>
<td></td>
</tr>
<tr>
<td>2.4</td>
<td>10</td>
<td>900</td>
<td>5</td>
<td>0.60</td>
<td>3</td>
<td>7</td>
<td>5</td>
<td>2.91</td>
<td></td>
</tr>
<tr>
<td>3.4</td>
<td>10</td>
<td>900</td>
<td>5</td>
<td>0.40</td>
<td>4</td>
<td>7</td>
<td>8</td>
<td>3.14</td>
<td></td>
</tr>
<tr>
<td>2.5</td>
<td>15</td>
<td>100</td>
<td>9</td>
<td>0.33</td>
<td>6</td>
<td>12</td>
<td>9</td>
<td>5.19</td>
<td></td>
</tr>
<tr>
<td>3.5</td>
<td>15</td>
<td>100</td>
<td>8</td>
<td>0.38</td>
<td>5</td>
<td>12</td>
<td>13</td>
<td>5.75</td>
<td></td>
</tr>
<tr>
<td>2.6</td>
<td>15</td>
<td>900</td>
<td>11</td>
<td>0.36</td>
<td>7</td>
<td>11</td>
<td>10</td>
<td>5.92</td>
<td></td>
</tr>
<tr>
<td>3.6</td>
<td>15</td>
<td>900</td>
<td>12</td>
<td>0.33</td>
<td>6</td>
<td>11</td>
<td>16</td>
<td>6.44</td>
<td></td>
</tr>
</tbody>
</table>

\(^1\) = MINOS Returned Integer Answer.
Table II shows a slight advantage to Option 2 when running the test problems. It appears that although roughly the same number of nodes were generated, the extra cuts generated did not seem to pay off in reducing the enumeration time. Rather, the additional cuts cost time in the amount of time needed to compute, test, and store the necessary information. The main difference between Option 2 and Option 3 requires that 2 linear programming relaxations must be solved in every subproblem iteration, and it would seem that in problems of this magnitude, the extra work is not paying off. It is hypothesized that with larger problems, the gap between the two options would close, with Option 3 potentially outperforming Option 2. However, for problems of this scale, Option 2 performed better. For this reason, Option 2 was selected as the option for phase 2 problem testing.

In the second phase of problem testing, the data of problem 6 was used (15 destinations, 900 capacity) and various scenarios that would be of practical use to a modeler were developed, with some interesting results. Phase 2 consisted of 6 more problems developed out of 3 possible scenarios, each with 2 sets of characteristic values.

For Scenario 1, certain links deemed as critical to the evacuation traffic flow were presumed unusable so that traffic was necessarily routed around these links. The links were chosen for their criticality in evacuating residents chiefly by their location and typical load experience provided by the
tests in the phase 1 test runs. Each scenario deleted 5 critical links, with the results printed in Table III.

Scenario 2 was developed by presuming that the disaster would strike sometime when the resort areas near the beaches were crowded with tourists not typically accounted for in the evacuation planning. The tourists were also assumed to be seeking shelter, thus becoming additional evacuees. Two separate loads were considered: a typical weekend load with a 10% link overload, and a holiday weekend load with a 25% link overload. The results are provided in Table III.

The final problems under Scenario 3 were developed by assuming an inordinate amount of traffic on the roads known to lead out of town in an attempt to simulate a typical business afternoon when rush hour traffic increases for the working commuters. The commuters are assumed not to be using the evacuation shelters; rather, they contribute to an additional highway load similar to \( \bar{x} \). Once again, two distinct loads were modeled, and the same percentage overloads as in Scenario 2 were used for this Scenario. Results are shown in Table III.
<table>
<thead>
<tr>
<th>SCENARIO</th>
<th>DEST.</th>
<th>CAP.</th>
<th>NODES</th>
<th>F.E.</th>
<th>N. GEN.</th>
<th>SHELT.</th>
<th>CUTS</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>BASELINE</td>
<td>15</td>
<td>900</td>
<td>11</td>
<td>0.36</td>
<td>7</td>
<td>11</td>
<td>10</td>
<td>5.92</td>
</tr>
<tr>
<td>1.1</td>
<td>15</td>
<td>900</td>
<td>13</td>
<td>0.30</td>
<td>8</td>
<td>11</td>
<td>11</td>
<td>6.19</td>
</tr>
<tr>
<td>1.2</td>
<td>15</td>
<td>900</td>
<td>12</td>
<td>0.33</td>
<td>7</td>
<td>11</td>
<td>12</td>
<td>6.11</td>
</tr>
<tr>
<td>2.1</td>
<td>15</td>
<td>900</td>
<td>14</td>
<td>0.28</td>
<td>9</td>
<td>11</td>
<td>11</td>
<td>7.21</td>
</tr>
<tr>
<td>2.2</td>
<td>15</td>
<td>900</td>
<td>15</td>
<td>0.33</td>
<td>10</td>
<td>11</td>
<td>13</td>
<td>7.48</td>
</tr>
<tr>
<td>3.1</td>
<td>15</td>
<td>900</td>
<td>13</td>
<td>0.39</td>
<td>9</td>
<td>11</td>
<td>12</td>
<td>7.18</td>
</tr>
<tr>
<td>3.2</td>
<td>15</td>
<td>900</td>
<td>15</td>
<td>0.40</td>
<td>11</td>
<td>11</td>
<td>12</td>
<td>7.45</td>
</tr>
</tbody>
</table>
It is apparent that all three Scenarios did impact the CPU time by either rerouting traffic along links which were not optimal in the phase 2 problem, or by requiring more analysis before optimality could be verified. Overall, it appears beneficial to use the Option 2 choice to generate a single cut in addition to the strongest surrogate constraint at each node, and to use them together when executing logical tests during implicit enumeration. The heuristic performed very well on the data set, giving answers no greater than 107% of the optimal objective for all problems. It is suggested that larger problems be run to determine how well the heuristic performs against a more difficult set of problems. The phase 2 and phase 3 runs also contain interesting information. It would appear in the phase 2 runs that shelter capacity does not greatly affect solution time. Rather, the number of potential destinations seems to be the driving variable. The solution times did rise with the larger number of destinations, but not drastically. This suggests that the algorithm developed has a capability which far exceeds the scope of the test problems employed in this study.
Chapter VI
Proposed Further Research

The transportation engineering field offers many research opportunities for operations research analysts, mathematicians, and civil engineers. There is still a wide variety of problems, modeling formulations, and methodologies that require investigation in this field. This chapter serves to propose several enhancements that could be implemented in conjunction with the problem formulation presented earlier, and in particular, a user-equilibrating formulation is also derived.

One option that could be considered for solving the current formulation of our problem is another extension of the Generalized Benders'-based implicit enumeration algorithm. It is possible to retain all (or most) cuts generated by the earlier heuristic procedure and the new algorithm: in effect, every iteration of the program through MINOS 5.1 could generate one constraint of the form in (50). Suppose that the heuristic iterates $Q$ times and that these $Q$ Generalized Benders' cuts are stored in a matrix format. Now when the implicit enumeration branch and bound routine iterates and updates the incumbent solution (as in Steps 2 of Options 2 and 3), the corresponding $y$ vector $\bar{y}$ can be used to generate another generalized Benders' cut through the subproblem, which
may be stored and used as a linear lower bounding constraint of the form (50).

It is possible also to provide for the possibility of "throwing" away of these generated inequalities. The modeler could empirically decide on a prechosen maximum value of cuts to keep active. Disqualification and subsequent removal of cuts could be based on the value of the objective at the current incumbent solution; in other words, if the number of cuts exceeds the prescribed value, say R, then each time a new cut is generated, it can replace the least binding cut at the current incumbent solution. This cut purging step would serve to streamline data storage requirements and to speed up execution since fewer computations would need to be made.

Whether cuts are "thrown" or not, the next step of such an algorithm would be to develop a strongest surrogate constraint by dualizing the set of Benders' constraints with a dual multiplier vector \( \delta \) corresponding to a linear programming relaxation of the problem. The same logic that produced (83) asserts that the sum of the \( \delta \) multipliers equals 1.0; in effect, the strongest surrogate cut

\[
\gamma \geq \sum_{j=1}^{R} \delta_j \nu_j - \sum_{i=1}^{R} \left[ \sum_{j=1}^{R} \delta_j \pi_{ij} \right] C_i y_i
\]

(89)

is merely a weighted combination of the active cut set. In addition, the constraint set \( y \) in \( Y \) could be surrogated to develop the form

\[
\alpha \cdot y \leq \beta.
\]

(90)
using the optimal dual variables. Then, the 0-1 knapsack problem

\[
\tilde{\nu} = \sum_{j=1}^{R} \delta_j \nu(x^j) - \max \sum_{j=1}^{R} \delta_j \pi^j C_i y_i
\]

subject to

\[
\alpha \cdot y \leq \beta
\]

\[
0 \leq y \leq 1, \text{ integer}
\]

could be solved. Due to the utilization of the optimal dual variables, note that a solution to (91)-(93) with a relaxation of integrality in the conditions (93) produces a lower bound to the master program. The binary knapsack (91)-(93) can be solved to a predetermined level to arrive at another lower bound \( \tilde{\nu} \) for the problem (91), where

\[
\tilde{\nu} \leq \tilde{\nu} \leq \tilde{\nu}. \quad (94)
\]

If

\[
\tilde{\nu} \geq \nu^*, \quad (95)
\]

the current node may be fathomed, since it will not produce a new incumbent. If (95) is false, we may continue with the branching variable selection as before.

The second major potential for further research examined in this thesis describes the alternate problem formulation discussed previously in Chapter III. As was stated earlier, the user-equilibrating approach constitutes a parallel
modeling effort in the solution of the transportation evacuation network flow problem class.

As before, let

\( D_i \) = demand emanating from a node \( i \) in \( N_s \).

\( j \) in \( N_d \) = a potential shelter location.

\( C_j \) = capacity of shelter \( j \), \( j \) in \( N_d \).

\( u_i \) = upper bound on link (arc) flows.

\( Y \) = constraint set for shelter locations \( y \), where

\[ y_j = 1, \text{ if a shelter is located at node } j, \]

\[ 0, \text{ otherwise, for all } j \in N_d \]

\( x_{pq} \) = flow on link \((p,q)\).

Let \( k = 1, \ldots, K \) represent all possible origin - to - destination paths from source to potential destination nodes. This set could be designed to include all self-node paths for the cases in which node \( j \) is an element of both \( N_s \) and \( N_d \).

The path list could be based on the cross product \( N_s \times N_d \) to insure that all paths are enumerated. In addition, let \( B_i = \left[ k \in [1, \ldots, K] : \text{ path } K \text{ starts from node } i \right] \) , \( i \in N_s \), and \( T_j = \left[ k \in [1, \ldots, K] : \text{ path } k \text{ ends at node } j \right] \) , \( j \in N_d \). Denote \( p_k \) as a vector with cardinality equal to the arc set \((p,q)\) in \( A \), which is defined as a binary vector for each path \( k \) with 1 for an arc used in the path, and 0 otherwise. Define

\[ c_{pq}(x_{pq}) = t_{pq} + \tau_{pq}(x_{pq} + \overline{x}_{pq})^\beta \]  

(96)

as the (per user) arc cost coefficient. Let \( \lambda_k \) = flow on path \( k \).
The problem may now be formulated as

\[
\begin{align*}
\text{minimize} & \quad \sum_{(p,q) \in A} \int_0^{T_{pq}} cpq(f) \, df \\
\text{subject to:} & \quad \sum_k p_k \lambda_k = (x) \\
& \quad \sum_{k \in B_i} \lambda_k = D_i, \quad \forall i \in N, \\
& \quad \sum_{k \in T_j} \lambda_k \leq C_j y_j, \quad \forall j \in N_d, \\
& \quad \lambda_k \geq 0, \quad k = 1, \ldots, K, \quad 0 \leq x \leq u, \quad y \in Y.
\end{align*}
\]

Note that the total system cost is given by

\[
\sum_{(p,q) \in A} cpq(x_{pq}) \cdot x_{pq}.
\]

Also here, given an optimal solution \((x^*, \lambda^*, y^*)\), and assuming above that (1) \(N_d\) corresponds only to the particular \(y_j\)’s which are unity, and (2) the paths \(k=1, \ldots, K\) also correspond to the particular \(y_j\)’s which are unity without loss of generality, we obtain the equivalent problem

\[
\begin{align*}
\text{minimize} & \quad \sum_{(p,q) \in A} \int_0^{T_{pq}} cpq(f) \, df \\
\text{subject to:} & \quad \sum_k p_k \lambda_k = (x) \\
& \quad \sum_{k \in B_i} \lambda_k = D_i, \quad \forall i \in N, \\
& \quad \sum_{k \in T_j} \lambda_k \leq C_j, \quad \forall j \in N_d \\
& \quad \lambda_k \geq 0, \quad 0 \leq x \leq u.
\end{align*}
\]

Denote

\((\pi_{pq}), (\alpha_i), \text{ and } (\beta_j)\)
as dual variables corresponding to constraints (104), (105), and (106), respectively, and let \( \theta_{pq} \geq 0 \) be the dual variables associated with

\[-x_{pq} \geq -u_{pq} \tag{109}\]

in (107). Then, we obtain

\[c_{pq}(x_{pq}) - \pi_{pq} + \theta_{pq} \geq 0. \tag{110}\]

with the left side equal to 0 if \( x_{pq} > 0 \), for each arc \((p,q)\) in \( A \). In addition,

\[\sum_{(p,q) \in A} \pi_{pq} \cdot \alpha_{i(k)} - \beta_{j(k)} \geq 0, \tag{111}\]

and is equal to 0 if \( \lambda \) is strictly positive, for all used paths \( k=1, \ldots, K \); where \( p_k = [(p,q) \in A: \text{arc } (p,q) \text{ is part of path } k] \); \( i(k) = \text{origin node in } N_s \text{ for path } k \); and \( j(k) = \text{terminal node in } N_d \text{ for path } k \). Therefore, the result follows that if

\[\lambda_{k_1} > 0, \ \lambda_{k_2} > 0, \ i(k_1) = i(k_2), \ j(k_1) = j(k_2). \tag{112}\]

(111) asserts that

\[\sum_{(p,q) \in P_{k_1}} \pi_{pq} = \sum_{(p,q) \in P_{k_2}} \pi_{pq}. \tag{113}\]

Since both \( \lambda_{k_1} \) and \( \lambda_{k_2} \) are positive, then from (111) all corresponding arcs on the paths must contain positive flow, and with this result, (110) produces

\[\alpha_{i(k)} + \beta_{j(k)} = \sum_{(p,q) \in P_{k_1}} [c_{pq}(x_{pq}) + \theta_{pq}] = \sum_{(p,q) \in P_{k_2}} [c_{pq}(x_{pq}) + \theta_{pq}]. \tag{114}\]
So, for any path \( k \), one can derive from (110) and (111), that

\[
\sum_{(p,q) \in P_k} [c_{pq}(x_{pq}) + \theta_{pq}] \geq \sum_{(p,q) \in P_k} \pi_{pq} \geq \alpha_i(k) + \beta_j(k),
\]

verifying Wardrop's user equilibrium equation when the upper bounding constraints in (107) are dualized and accommodated into the link cost function (See Frank [1981]).

The problem formulation presented above utilizes a user-oriented objective formulation for the development of the cost function and a path formulation instead of an arc formulation. As an alternative objective function for (97), a system objective formulation (102) could be employed. Additionally, upper bounds could be included in the constraint set to bound the user's cost for any given path. The user's cost on a path \( k \) is \( p_k \cdot c(x) \) where \( c(x) \) is the cost vector \( (c_{pq}(x_{pq}), \text{for } (p,q) \in A) \). It may benefit the model to stipulate the requirements that

\[
p_k \cdot c(x) \leq R_i, \quad \forall k \in S_r, \quad \forall i \in N_s, \text{ if } \lambda_k > 0;
\]

where \( R_i \) is an upper bound on the cost for all system users originating from node \( i \) in \( N_s \). The constraint could be modeled in a standard fashion using binary variables (See Taha [1975] for example).

Either of the two formulations described above, by taking into consideration path flows as well as arc flows, and developing a new type of objective equation, would be more user-oriented than the model that was developed and tested for
this research. The potential for further research of this type is great, and deserves to be examined with more scrutiny.
Chapter VII

Bibliography


Appendix A

Using MINOS 5.1 in a FORTRAN Program
Appendix A

Using MINOS 5.1 in a FORTRAN Program

Using MINOS 5.1 is a fairly straightforward process, but there are several points contained in this appendix that will save future MINOS 5.1 users much time and frustration if they take the time to read and understand my comments. The two most important documents that a MINOS 5.1 user needs are (1) "Using MINOS 5.1 at Virginia Tech", and (2) "MINOS 5.1 User's Guide". The first document is a 10-page handout prepared by the people at User Services and can be picked up there. The second document is a 120-page handbook written by the people at the Stanford's Systems Optimization Laboratory who developed the software. Both documents are helpful, but it is imperative that one reads and understands "Using MINOS 5.1 at Virginia Tech": the reasons should become clear in this appendix.

MINOS 5.1 can be used as a stand-alone piece of software, in which a data set is entered and a solution is developed, or it can be imbedded in a FORTRAN program to be used interactively when called upon by the FORTRAN program. This second use of MINOS is more powerful and it is this use which this appendix concerns. In either case, the user must prepare three files before executing MINOS 5.1: (i) the input, execution and output option file, (ii) the problem structure file, and (iii) the file definitions file.
The first file is described very well in the User's Guide and is known as the SPECS file. All specifications have default values, so it is only necessary to specify values on those parameters which the user is changing. The second file is known as the MPS file and is also described in detail in the User's Guide. The MPS format largely follows the industry standard, but it is wise to consult the User's Guide for potential conflicts before submitting an MPS file from previous research/software efforts. The third file is necessary only at Virginia Tech. It is called MINOS FILES, and its purpose is to make file interfacing between your USERID and the MINOS 5.1 package a simple matter. This file is described in detail in the User Services handout.

No matter whether the user wants to use MINOS 5.1 as a stand-alone program or whether the user is imbedding MINOS 5.1 in a FORTRAN code, the execution of MINOS 5.1 is still controlled by an exec called RUNMINOS. RUNMINOS is written in REXX. It may be necessary to alter RUNMINOS to suit some particular needs, particularly in the file definition section of the exec. If this is true, simply receive a copy of RUNMINOS at an A disk (from the MINOS disk) and alter it, or have User Services help you.

When MINOS 5.1 is used inside a FORTRAN program, the programmer must remember to do these things:

1. Declare MINOS1 "external" in the code;

2. Assign a value to NWCORE,
(3) Dimension Z(NWCORE);
(4) Compile with VS FORTRAN Version 2 compiler.

These things are a must to get MINOS 5.1 to operate in a program. In addition, there are several things that the user can do to speed up execution of MINOS 5.1 in a FORTRAN program. When using MINOS 5.1 in a FORTRAN program, the user does not begin execution by the LOAD (START) command sequence: rather, the RUNMINOS exec is used. When control is passed over to MINOS 5.1, the software package begins outputting diagnostics to the terminal. The diagnostics can be disabled, or at least abbreviated, by correct parameter specification in the SPECS file, as described in the User's Guide. Also, the frequency with which the solution is saved can be adjusted here.

Another trick that can speed up the total modeling-verification process is to write a program to create the MPS file. This is not difficult, and an example called MAKEMPS is given in the next appendix. The input to MAKEMPS is unformatted data and the output is a correct (for MINOS 5.1) MPS file; this is a real timesaver. If the user has imbedded MINOS 5.1 in a FORTRAN program, the user will probably be reading from the solution file and possibly altering the MPS file for future iterations, so it is necessary to be able to create a "fresh" MPS file at will. Another important point to remember is that MINOS 5.1 reads and writes sequential files. This means that if MINOS 5.1 has in fact been imbedded
in a FORTRAN program, then to reread or rewrite a file, the file must first be rewound using the FORTRAN command 'REWIND'. It is also a good idea, although not absolutely necessary, to 'CLOSE' these file(s) before 'REWIND'ing them.

Before diving into MINOS 5.1, try the examples that are provided on the MINOS disk, and be sure to acquire copies of the documents mentioned earlier. A last piece of advice: learn about the three necessary files, their purposes, and their capabilities. Every ten minutes spent studying their structure and syntax will save an hour or more in frustration and valuable programming/computer time!
Appendix B

Codes for FORTRAN Programs MAKEMPS and RMIP
Appendix B

Codes for FORTRAN Programs MAKEMPS and RMIP

C      THIS PROGRAM IS DESIGNED TO TRANSFORM OUR CONDENSED DATA INTO A
C      CORRECTLY FORMATTED AND READY TO USE MPS FILE FOR THE MINOS 5.1
C      SOFTWARE PACKAGE.
C
C
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
C
THE FIRST INTEGER STATEMENT IS TO SPECIFY NEEDED PARAMETERS.
C
C
INTEGER NODES,NOFARC,NTERM,NYCON
PARAMETER (NODES=57,NOFARC=118,NTERM=15,NYCON=4)
C
C
NODES=TOTAL # NODES: NOFARC=TOTAL # OF ARCS, EXCEPT ARTIFICIAL:
C NTERM=TOTAL # OF TERMINAL NODES: NYCON=# OF Y-IN-Y CONSTRAINTS:
C NREC=RECORD #: NIND= AN INDEX: NTO & NFROM= TO & FROM ARC INDICES:
C NCOL=A CODED VARIABLE #.
C
C
THIS INTEGER STATEMENT SPECIFIES THE INTEGER ARRAYS.
C
C
INTEGER NARCS(NODES),KFARCS(NOFARC)
C
C
CHARACTER*1 E,G,L
DIMENSION REAL ARRAYS

DIMENSION D(NODES), C(NTERM), U(NOFACT), S(NYCON), YINY(NYCON, NTERM)

DATA INPUT SECTION

READ (8,*) (NARCS(I), I=1, NODES)
READ (8,*) (KFARCS(I), I=1, NOFACT)
READ (8,*) (D(I), I=1, NODES)
READ (8,*) (C(I), I=1, NTERM)
READ (8,*) (U(I), I=1, NOFACT)
READ (8,*) (S(I), I=1, NYCON)
DO 2 I=1, NYCON
    READ (8,*) (YINY(I, J), J=1, NTERM)
2 CONTINUE
READ (8,*) E, G, L

OPEN A DIRECT ACCESS FILE TO CONTAIN OUR MPS FILE

OPEN (UNIT=2, FILE='MPSFILE', ACCESS='DIRECT', FORM='FORMATTED', 
$       STATUS='NEW', RECL=61)

INPUT INTO NEW FILE-- NAME HEADER FIRST, THEN ROW SECTION

NREC=1
WRITE (2,500,REC=NREC)
NREC=NREC+1
NROW=1000

WRITE (2,510,REC=NREC)
NREC=NREC+1

DO 10 I=NROW+1,NROW+NODES
   WRITE (2,520,REC=NREC) E,I
   NREC=NREC+1
10 CONTINUE

DO 20 I=1001+NODES,1000+NODES+NTERM
   WRITE (2,520,REC=NREC) G,I
   NREC=NREC+1
20 CONTINUE

DO 30 I=1001+NODES+NTERM,1000+NODES+NTERM+NYCON
   WRITE (2,520,REC=NREC) L,I
   NREC=NREC+1
30 CONTINUE

INPUT COLUMN HEADER AND DATA NOW

WRITE (2,530,REC=NREC)
NREC=NREC+1

C

NIND=0

DO 40 I=1,NODES
   IF (NARCS(I) .EQ. 0) GO TO 40
   DO 50 J=1,NARCS(I)
      NFROM=I*1000
      NTO=KFARCS(NIND+J)
      NCOL=1000000+NFROM+NTO
      WRITE (2,540,REC=NREC) NCOL,1000+I,1000+NTO
      NREC=NREC+1
   50      CONTINUE
   NIND=NIND+NARCS(I)
40 CONTINUE

C

DO 60 I=1001,1000+NODES
   WRITE (2,550,REC=NREC) I,I
   NREC=NREC+1
60 CONTINUE

C

DO 70 I=1001,1000+NTERM
   WRITE (2,560,REC=NREC) NODES-NTERM+I,NODES-NTERM+I,NODES+I
   NREC=NREC+1
70 CONTINUE

C
DO 80 J=1001,1000+NTERM
    WRITE (2,570,REC=NREC) NODES-NTERM+J,NODES+J,C(J-1000)
    NREC=NREC+1
    DO 90 I=1001,1000+NYCON
        WRITE (2,570,REC=NREC) NODES-NTERM+J,NODES+NTERM+I,
                      $  YINY(I-1000,J-1000)
        NREC=NREC+1
    90 CONTINUE
80 CONTINUE

C
C   RHS HEADER AND CARDS COME NEXT
C
C
    WRITE (2,580,REC=NREC)
    NREC=NREC+1

C

    DO 100 I=1001,1000+NODES
        WRITE (2,590,REC=NREC) I,D(I-1000)
    NREC=NREC+1
100 CONTINUE

C

    DO 105 I=1000+NODES+1,1000+NODES+NTERM
        WRITE (2,595,REC=NREC) I
    NREC=NREC+1
105 CONTINUE

    DO 110 I=1001,1000+NYCON
80
WRITE (2,590,REC=NREC) NODES+NTERM+I,S(I-1000)
NREC=NREC+1
110 CONTINUE
C
C  NEXT ARE THE BOUNDS HEADER AND DATA CARDS
C  
WRITE (2,600,REC=NREC)
NREC=NREC+1
C
NIND=0
DO 120 I=1,NODES
  IF (NARCS(I) .EQ. 0) GO TO 120
  DO 130 J=1,NARCS(I)
    NFROM=1000*I
    NTO=KFARCS(NIND+J)
    NCOL=1000000+NFROM+NTO
    WRITE (2,610,REC=NREC) NCOL,U(NIND+J)
    NREC=NREC+1
  130 CONTINUE
NIND=NIND+NARCS(I)
120 CONTINUE
C
DO 140 I=1001,1000+NTERM
  WRITE (2,620,REC=NREC) NODES-NTERM+I
  NREC=NREC+1
140 CONTINUE
WRITE (2,630,REC=NREC)
C
C CLOSE SEQUENTIAL FILE NOW.
C
C CLOSE (UNIT=2)
WRITE (6,640) NREC
C
C FORMAT STATEMENTS AND TERMINATION.
C
500 FORMAT ('NAME',10X,'FILE')
510 FORMAT ('ROWS')
520 FORMAT (1X,A1,2X,'ROW',I4)
530 FORMAT ('COLUMNS')
540 FORMAT (4X,'X',I7,2X,'ROW',I4,7X,'1.0',8X,'ROW',I4,6X,'-1.0')
550 FORMAT (4X,'XART',I4,2X,'ROW',I4,7X,'1.0')
560 FORMAT (4X,'Z',I4,5X,'ROW',I4,7X,'1.0',8X,'ROW',I4,6X,'-1.0')
570 FORMAT (4X,'Y',I4,5X,'ROW',I4,3X,F8.2)
580 FORMAT ('RHS')
590 FORMAT (4X,'RHS',7X,'ROW',I4,3X,F8.2)
595 FORMAT (4X,'RHS',7X,'ROW',I4,7X,'0.00')
600 FORMAT ('BOUNDS')
610 FORMAT (1X,'UP',1X,'BOUND',5X,'X',I7,2X,F8.2)
620 FORMAT (1X,'UP',1X,'BOUND',5X,'Y',I4,9X,'1.0')
630 FORMAT ('ENDATA')
640 FORMAT ('THE NUMBER OF RECORDS IS', I4)

STOP

END
THIS PROGRAM IS DESIGNED TO UTILIZE AN EARLIER CREATED MPS FILE TO INPUT TO THE MINOS OPTIMIZER. THE HEURISTIC FOR OUR STRONGEST SURROGATE CONSTRAINT BOUND THEN follows. Finally, the coding for options 2 and 3, the implicit enumeration codes, follows. The choice of options is made in the parameter statement under the variable name 'nopt'.

IMPLICIT DOUBLE PRECISION (A-H,O-Z)

DECLARE ALL EXTERNAL FUNCTIONS

EXTERNAL DSVRGP,MINOS1,DDLPRS,TIMEON,TIMECK

PROGRAM PARAMETERS

PARAMETER (NODES=57,NOFARC=118,NTERM=15,NYCON=4,NWCORE=100000, $ NPICE=3,NREC=556,MU=1,NOPT=1)

NODES=TOTAL # NODES: NOFARC=TOTAL # OF ARCS, EXCEPT ARTIFICIAL:
NTERM=TOTAL # OF TERMINAL NODES: NYCON=# OF Y-IN-Y CONSTRAINTS:
NWCORE=WORKING SPACE FOR MINOS:
NPICE=NUMBER OF LINES IN THE PIECEWISE APPROXIMATION TO THE ARC LOAD CURVE: MU=A FREQUENCY PARAMETER.
NOPT=OPTION FOR STOPPING COMPUTATION AFTER HEURISTIC ROUTINE:
C NOPT=1,2,OR 3 DEPENDING ON THE OPTION CHOSEN.

C
C DIMENSION REAL ARRAYS
C

DIMENSION PK(NPIECE),TK(NPIECE),PI(NTERM),Z1(NTERM),Y(NTERM),
$ YHAT(NTERM),RA(NTERM),RB(NTERM),IPERM(NTERM),Z(NWCORE),
$ ALPHA(NTERM),BIGM(NOFARC),T(NOFARC),TAU(NOFARC),
$ XBAR(NOFARC),U(NOFARC),C(NTERM),S(NYCON),
$ YINY(NYCON,NTERM),FEE(NTERM),DIST(NOFARC)

DIMENSION YSTAR(NTERM),YTEST(NTERM),YNEW(NTERM),
$ C3(NTERM+1),Z3(NTERM),FFS(NOFARC),TESMAT(10,NTERM+1),
$ PIHAT(NTERM),A(NYCON+2,NTERM+1),BL(NYCON+2),
$ BU(NYCON+2),IRTYPE(NYCON+2),YLB(NTERM+1),VALPI(NTERM),
$ YUB(NTERM+1),YSOL(NTERM+1),DSOL(NYCON+2),THETA(NYCON)

C
C DIMENSION INTEGER ARRAYS
C

INTEGER IVECT(NTERM),IPS(2,NTERM)

C
C DECLARE COMMON VARIABLES
C

COMMON /COMII/ IBETA

COMMON /COMRI/ BETA,F1,BIGM,T,TAU,XBAR

C
C INPUT DATA NEEDED FOR OBJECTIVE & BOUNDS COMPUTATIONS

C

READ (9,*) ALPH,BETA,IBETA
READ (9,*) (PK(I), I=1,NPIECE)
READ (9,*) (TK(I), I=1,NPIECE)
READ (9,*) (BIGM(I), I=1,NODES)
READ (9,*) (DIST(I), I=1,NOFARC)
READ (9,*) (FFS(I), I=1,NOFARC)
READ (9,*) (XBAR(I), I=1,NOFARC)
READ (9,*) (U(I), I=1,NOFARC)
READ (9,*) (C(I), I=1,NTERM)
READ (9,*) (S(I), I=1,NYCON)
DO 2 I=1, NYCON
    READ (9,*) (YINY(I,J), J=1, NTERM)
2 CONTINUE

C

C INTERMEDIATE CALCULATIONS

C

CALL TIMEON

DO 10 I=1, NOFARC
    SLOPE=0.0
    DO 20 J=1, NPIECE
        SLOPE=SLOPE+PK(J)**(IBETA+1)/TK(J)**IBETA
    20 CONTINUE
    T(I)=DIST(I)/FFS(I)
TAU(I)=ALPH*T(I)*SLOPE/U(I)**IBETA

10 CONTINUE
NCYC=0
C
C CALL OPTIMIZER
C
1000 MUVAL=0
REWIND (5)
REWIND (10)
CLOSE (10)
CALL MINOS1(Z,NWCORE)
C
C EXTRACT THE DUAL VARIABLES AND Z1,Y VALUES FROM SOLUTION FILE.
C
DO 30 I=1,NODES+14
   READ (10,*)
30 CONTINUE
DO 40 I=1,NTERM
   READ (10,5500) PI(I)
   IF (ABS(PI(I)) .LT. 0.001) PI(I)=0.0
40 CONTINUE
DO 45 I=1,NYCON
   READ (10,5500) THETA(I)
   IF (ABS(THETA(I)) .LT. 0.001) THETA(I)=0.0
45 CONTINUE
DO 50 I=1,NOFARC+NODES+5
    READ (10,*)
50 CONTINUE

DO 60 I=1,NTERM
    READ (10,5510) Z1(I)
60 CONTINUE

DO 70 I=1,NTERM
    READ (10,5510) Y(I)
70 CONTINUE

C
C DEVELOP STRONGEST SURROGATE CONSTRAINT LOWER BOUND
C
NCYC=NCYC+1
IF (NCYC .GT. 1) GO TO 84
DCON=0.0
DO 80 I=1,NTERM
    DCON=DCON+PI(I)*(Z1(I)-C(I)*Y(I))
    TESMAT(1,I)=PI(I)
80 CONTINUE

TESMAT(1,NTERM+1)=F1
SSCLB=F1+DCON
IF (XSTOP .EQ. 1.0) GO TO 1499

C
C INTEGER SOLUTION TEST LOOP
C
84 NCOUNT=0
DO 85 I=1,NTERM
   IF ((Y(I) .GE. 0.001) .AND. (Y(I) .LE. 0.999)) THEN
      NCOUNT=NCOUNT+1
      GO TO 86
   ELSE
      CONTINUE
   END IF
85 CONTINUE
86 IF (NCOUNT .EQ. 0) GO TO 1499
C
C ROUND Y'S DOWN AND CALCULATE ROUNDING UP RULE FOR CANDIDATE Y'S.
C
NCON1=0
NCON2=0
DO 90 I=1,NTERM
   ALPHA(I)=0.0
90 CONTINUE
DO 100 I=1,NTERM
   IF (Y(I) .GE. 0.999) THEN
      YHAT(I)=1.0
      RA(I)=0.0
      OPEN (UNIT=2,FILE='MPSFILE',ACCESS='DIRECT',FORM=
$        'FORMATTED', STATUS='OLD',RECL=61)
      WRITE (2,5520,REC=NREC-1-NTERM+I) 1000+NODES-NTERM+I
ELSE IF (Y(I) .GT. 0.001) THEN
    YHAT(I)=0.0
    NCON1=NCON1+1
    THETAS=0.0
    DO 110 K=1,NYCON
        THETAS=THETAS+THETA(K)*YINY(K,I)
    110 CONTINUE
    ALPHA(I)=THETAS
    IF (ALPHA(I) .EQ. 0.0) THEN
        RA(I)=-1.0D12-I
    ELSE
        RA(I)=PI(I)*C(I)/ALPHA(I)
    END IF
ELSE
    YHAT(I)=0.0
    NCON2=NCON2+1
    THETAS=0.0
    DO 120 K=1,NYCON
        THETAS=THETAS+THETA(K)*YINY(K,I)
    120 CONTINUE
    ALPHA(I)=THETAS
    IF (ALPHA(I) .EQ. 0.0) THEN
        RA(I)=1.0/(1.0D12+I)
    ELSE
        RA(I)=PI(I)*C(I)/ALPHA(I)
END IF
END IF

100 CONTINUE

C
C SORT THE RATIO ARRAY AND STORE THE PERMUTATION VECTOR.
C

N=NTERM
DO 130 I=1,NTERM
    IPERM(I)=I
130 CONTINUE

C
CALL DSVRGP(N,RA,RB,IPERM)

C
C NOW SORTED ARRAY = RB, AND PERMUTATION IS STORED IN IPERM.
C
C BEGIN TRYING TO ROUND UP Y-VALUES ONE AT A TIME.
C

DO 140 K=1,NTERM
    IF (YHAT(IPERM(K)) .EQ. 1.0) GO TO 140
    YHAT(IPERM(K))=1.0
    DO 150 I=1,NYCON
        XSUM=0.0
        DO 160 J=1,NTERM
            XSUM=XSUM+YHAT(J)*YINY(I,J)
        160 CONTINUE
        IF (XSUM .GT. S(I)) GO TO 140
CONTINUE

MUVAL=MUVAL+1

OPEN (UNIT=2, FILE='MPSFILE', ACCESS='DIRECT', FORM=
'S FORMATTED', STATUS='OLD', RECL=61)

WRITE (2,5520,REC=NREC-1-NTERM+IPERM(K)) 1000+NODES-NTERM+
$ IPERM(K)

IF (MUVAL .EQ. MU) GO TO 1000

CONTINUE

IF (MUVAL .EQ. 0) GO TO 1499

IF (MUVAL .GT. 0) THEN
  XSTOP=1.0
  GO TO 1000
END IF

1499 IF (NOPT .EQ. 1) GO TO 8000

C

C THIS BEGINS THE IMPLICIT ENUMERATION SECTION OF THE PROGRAM.

C

DO 210 I=1,NTERM
  YSTAR(I)=YHAT(I)
  IVECT(I)=2

210 CONTINUE

NUM=0

NFATH=0

NBRAN=1

NGENO=1
NGBCUT=0
NUMFR= NTERM
VSTAR=F1
GAMMA= SSCLB
NSTOP=0
NQ=1

C

C BEGIN LOGICAL TESTS.
C

1600 DO 251 M=1,NQ
   NCOUN=0
   PIZEE=0.0
   DO 220 I=1,NTERM
       PIZEE=PIZE+TESMAT(M,I)*Z1(I)
   220 CONTINUE
   QOFNU=VSTAR+PIZE
   PICEE=0.0
   DO 230 I=1,NTERM
       IF (IVECT(I) .EQ. 0) GO TO 230
       PICEE=PICEE+TESMAT(M,I)*C(I)
   230 CONTINUE
   GAMMAQ=QOFNU-PICEE
   IF (GAMMAQ .GE. VSTAR) THEN
       CALL FATHOM (IPS,NCOUN,NUMFR,NUM,NSTOP,NFATH,IVECT)
       IF (NSTOP .EQ. 1) GO TO 8000
GO TO 1600
END IF

C
C CANCELLATION-ONE TEST
C

PICEE=0.0
DO 250 I=1,NTERM
   IF (IVECT(I) .EQ. 2) THEN
      IF (GAMMAQ+TESMAT(M,I)*C(I) .GE. VSTAR) THEN
         NUMFR=NUMFR-1
         NCOUN=NCOUN+1
         NUM=NUM+1
         IPS(1,NUM)=I
         IPS(2,NUM)=2
         IVECT(I)=1
      END IF
   END IF
END IF
250 CONTINUE

251 CONTINUE
   IF (NUMFR .EQ. 0) GO TO 2000
C
C
C CANCELLATION-ZERO TEST
C

DO 260 I=1,NTERM
   YTEST(I)=0.0

260 CONTINUE
260 CONTINUE
    IF (NUM .EQ. 0) GO TO 271

    DO 270 I=1,NTERM
        IF (IVECT(I) .EQ. 1) YTEST(I)=1.0
    270 CONTINUE

271 DO 280 I=1,NYCON
    XSUM=0.0
    DO 290 J=1,NTERM
        IF (IVECT(J) .EQ. 2) THEN
            IF (XSUM+YINY(I,J) .GT. S(I)) THEN
                NUMFR=NUMFR-1
                NCOUN=NCOUN+1
                NUM=NUM+1
                IPS(1,NUM)=-J
                IPS(2,NUM)=2
                IVECT(J)=0
            END IF
        END IF
    290 CONTINUE

    280 CONTINUE

1999 IF (NCOUN .GE. 1) GO TO 1600

    IF (NOPT .EQ. 3) GO TO 3000

C

C STEPS 2-7, OPTION 2.

C
C USING DDLPRS TO SOLVE LP FOR BOUND COMPUTATION.

2000 M=NYCON
   LDA=M
   NVAR=NTERM
   DO 320 I=1,NTERM
      C3(I)=PI(I)*C(I)
   320 CONTINUE

   DO 330 I=1,NYCON
      IRTYPE(I)=1
   330 CONTINUE

   DO 335 I=1,NTERM
      YLB(I)=0.0
      YUB(I)=1.0
   335 CONTINUE

   DO 336 I=1,NTERM
      IF (((IVECT(I) .EQ. 0) .OR. (PI(I)*C(I) .LE. 0.001)) THEN
         YUB(I)=0.0
      ELSE IF (IVECT(I) .EQ. 1) THEN
         YLB(I)=1.0
      END IF
   336 CONTINUE

CALL DDLPRS (M,NVAR,YINY,LDA,BL,S,C3,IRTYPE,XLB,XUB,OBJ,YSOL,
DSOL)  
C
RETURN OBJECTIVE VALUE TO MAIN PROGRAM
C
NGBCUT=NGBCUT+1

DO 2001 I=1, NTERM
    WRITE (21,*) YSOL(I)

2001 CONTINUE

TESNU=QOFNU+OBJ
IF (TESNU .GE. VSTAR) THEN
    CALL FATHOM (IPS, NCOUN, NUMFR, NUM, NSTOP, NFATH, IVEC)
    IF (NSTOP .EQ. 1) GO TO 8000
    GO TO 1600
END IF

DO 340 I=1, NTERM
    YNEW(I)=YSOL(I)
    WRITE (21,*) YSOL(I)

340 CONTINUE

GO TO 8000

DO 350 I=1, NTERM
    IF ((YNEW(I) .GT. 0.001) .AND. (YNEW(I) .LT. 0.999)) GO TO 2900

350 CONTINUE

OPEN (UNIT=2, FILE='MPSFILE', ACCESS='DIRECT', FORM='FORMATTED',
     STATUS='OLD', RECL=61)
DO 360 I=1, NTERM
  IF (YNEW(I) .LE. 0.001) THEN
    WRITE (2, 5530, REC=NREC-1-NTERM+I) 1000+NODES-NTERM+I
  ELSE
    WRITE (2, 5520, REC=NREC-1-NTERM+I) 1000+NODES-NTERM+I
  END IF
360 CONTINUE
CLOSE(2)
REWIND(5)
REWIND(10)
CLOSE(10)
CALL MINOS1(Z, NWCORE)
C
C EXTRACT SOLUTION FROM PRESPECIFIED SOLUTION FILE
C
NQ=NQ+1
DO 370 I=1, NODES+14
  READ (10, *)
370 CONTINUE
DO 380 I=1, NTERM
  READ (10, 5500) VALPI(I)
  IF (ABS(VALPI(I)) .LT. 0.001) VALPI(I)=0.0
380 CONTINUE
IF (F1 .GE. VSTAR) GOTO 2900
VSTAR=F1
NGENO=NBRAN

DO 390 I=1,NOFARC+NODES+TERM+NYCON+5
   READ (10,*)

390 CONTINUE

DO 400 I=1,NTERM
   READ (10,5520) YSTAR(I)

400 CONTINUE

IF (NQ .LE. 10) THEN
   DO 401 I=1,NTERM
      TESMAT(NQ,I)=VALPI(I)

401 CONTINUE

TESMAT(NQ,NTERM+1)=F1
ELSE
   FHIGH=TESMAT(2,NTERM+1)
   DO 402 I=3,10
      IF (TESMAT(I,NTERM+1) .GT. FHIGH) THEN
         FHIGH=TESMAT(I,NTERM+1)
         NIND=I
      END IF

402 CONTINUE

IF (FHIGH .GT. F1) THEN
   DO 403 I=1,NTERM
      TESMAT(NIND,I)=VALPI(I)

403 CONTINUE

TESMAT(NIND,NTERM+1)=F1
END IF
END IF
DO 405 I=1,19+2*NODES+2*NTERM+NOFARC+NYCON
  READ (10,*)
405 CONTINUE
DO 410 I=1,NTERM
  READ (10,5520) YSTAR(I)
410 CONTINUE

C
C BRANCHING VARIABLE SELECTION STRATEGY
C

2900 IF (NUMFR .EQ. 0) THEN
    CALL FATHOM (IPS,NCOUN,NUMFR,NUM,NSTOP,NFATH,IVECT)
    IF (NSTOP .EQ. 1) GO TO 8000
    GO TO 1600
END IF
DO 420 I=1,NTERM
  IF (IVECT(I) .EQ. 2) THEN
    IF (YSTAR(I) .LE. 0.5) THEN
      FEE(I)=YSTAR(I)
    ELSE
      FEE(I)=1.0-YSTAR(I)
    END IF
  END IF
420 CONTINUE
DO 425 I=1,NTERM
   IF (IVECT(I) .EQ. 2) GO TO 426
425 CONTINUE
426 NARG=I
   IF (NUMFR .EQ. 1) GO TO 431
   DO 430 J=I+1,NTERM
      IF (IVECT(J) .EQ. 2) THEN
         IF (FEE(J)*PI(J)*C(J) .GT. FEE(NARG)*PI(NARG)*C(NARG))
            NARG=J
            GO TO 430
      ELSE IF (FEE(J)*PI(J)*C(J) .EQ. FEE(NARG)*PI(NARG)*C(NARG)) THEN
         IF (FEE(J) .GT. FEE(NARG)) THEN
            NARG=J
            GO TO 430
         ELSE IF (FEE(J) .EQ. FEE(NARG)) THEN
            IF (PI(J)*C(J) .GT. PI(NARG)*C(NARG)) THEN
               NARG=J
               GO TO 430
            END IF
         END IF
      END IF
   END IF
430 CONTINUE
431 NBRAN=NBRAN+1
    NUM=NUM+1
    IPS(1,NUM)=NARG
    IPS(2,NUM)=1
    NUMFR=NUMFR-1
    IVECT(NARG)=1
    GO TO 1600

C

C STEPS 2-7, OPTION 3
C
C FIRST SOLVE LP BOUND PROBLEM AS IN OPTION 2.
C
C USING DDLPRS TO SOLVE LP FOR BOUND COMPUTATION.
C
3000 M=NYCON
    LDA=M
    NVAR=NTERM
    DO 460 I=1,NTERM
       C3(I)=-PI(I)*C(I)
460 CONTINUE
    DO 470 I=1,NYCCN
       IRTYPE(I)=1
470 CONTINUE
    DO 475 I=1,NTERM
       YLB(I)=0.0
DO 476 I=1,NTERM
   IF (IVECT(I) .EQ. 0) THEN
      YUB(I)=0.0
   ELSE IF (IVECT(I) .EQ. 1) THEN
      YLB(I)=1.0
   END IF
476 CONTINUE

CALL DDLPRS (M,NVAR,YINY,LDA,BL,S,C3,IRTYPE,YLB,YUB,OBJ,YSOL,
$   DSOL)

C
C RETURN OBJECTIVE VALUE TO MAIN PROGRAM
C

NGBCUT=NGBCUT+1
TESNU=QOFNU+OBJ
IF (TESNU .GE. VSTAR) THEN
   CALL FATHOM (IPS,NCOUN,NUMFR,NUM,NSTOP,NFATH,IVECT)
   IF (NSTOP .EQ. 1) GO TO 8000
   GO TO 1600
END IF
OPEN (UNIT=2,FILE='MPSFILE',ACCESS='DIRECT',FORM='FORMATTED',
$   STATUS='OLD',RECL=61)
DO 480 I=1,NTERM
   WRITE (2,5550,REC=NREC-1-NTERM+I) 1000+NODES-NTERM+I,
$ \ YSOL(I) $

480 CONTINUE

CLOSE(2)

REWIND(5)

REWIND(10)

CLOSE(10)

CALL MINOS1 (Z,NWCore)

C

C EXTRACT SOLUTION VALUE AND DUAL MULTIPLIERS FROM SOL'N FILE.

C

DO 490 I=1,NODES+14

READ (10,*)

490 CONTINUE

DO 500 I=1,NTERM

READ (10,5500) PIHAT(I)

500 CONTINUE

DO 510 I=1,NOFARC+NODES+NYCON+5

READ (10,*)

510 CONTINUE

DO 520 I=1,NTERM

READ (10,5510) Z3(I)

520 CONTINUE

DO 530 I=1,NTERM

READ (10,5510) YNEW(I)
530 CONTINUE
C
SUM=0.0
DO 540 I=1, NTERM
    SUM=SUM+PIHAT(I)*C(I)*YNEW(I)
540 CONTINUE

VEEPI=F1+SUM
C
C A BINARY Y-VECTOR IMPLIES A NEW INCUMBENT SOLUTION. OF COURSE
C THE SOLUTION VALUE MUST ALSO BE LESS THAN THE INCUMBENT.
C
IF (F1 .LT. VSTAR) THEN
    DO 550 I=1, NTERM
        IF((YNEW(I) .GE. 0.001).AND.(YNEW(I) .LE. 0.999)) GO TO 570
    550 CONTINUE
    VSTAR=F1
    NGENO=NBRAN
    DO 560 I=1, NTERM
        YSTAR(I)=YNEW(I)
    560 CONTINUE

570 END IF
C
C SOLVE CUT LP USING IMSL SUBROUTINE DDLPRS.
C
M=NYCON+2
NVAR = NTERM + 1

DO 580 J = 1, NVAR - 1
   A(1, J) = PI(J) * C(J)

580 CONTINUE

A(1, NVAR) = 1.0

DO 590 J = 1, NVAR - 1
   A(2, J) = PIHAT(J) * C(J)

590 CONTINUE

A(2, NVAR) = 1.0

DO 600 I = 1, NYCON
   DO 610 J = 1, NVAR - 1
      A(I + 2, J) = YINY(I, J)
   610 CONTINUE

A(I + 2, NVAR) = 0.0

600 CONTINUE

LDA = M

BL(1) = QOFNU

BL(2) = VEEP1

DO 620 I = 1, NYCON
   BU(I + 2) = S(I)

620 CONTINUE

DO 630 I = 1, NVAR - 1
   C3(I) = 0.0

630 CONTINUE

C3(NVAR) = 1.0
IRTYPE(1)=2
IRTYPE(2)=2
DO 640 I=1,NYCON
   IRTYPE(I+2)=1
640 CONTINUE
DO 650 I=1,NTERM
   YLB(I)=0.0
   YUB(I)=1.0
650 CONTINUE
DO 660 I=1,NTERM
   IF (IVECT(I) .EQ. 0) THEN
      YUB(I)=0.0
   ELSE IF (IVECT(I) .EQ. 1) THEN
      YLB(I)=1.0
   END IF
660 CONTINUE
C
CALL DDLPRS (M,NVAR,A,LDA,BL,BU,C3,IRTYPE,YLB,YUB,GAMMA,YSOL,
            DSOL)
C
NGBCUT=NGBCUT+1
IF (GAMMA .GE. VSTAR) THEN
   CALL FATHOM (IPS,NOUN,NUMFR,NUM,NSTOP,NFATH,IVECT)
   IF (NSTOP .EQ. 1) GO TO 8000
GO TO 1600
END IF

C

BRANCHING VARIABLE SELECTION STRATEGY.

C

IF (NUMFR .EQ. 0) THEN

CALL FATHOM (IPS,NCOUN,NUMFR,NSTO,NFATH,IVECT)

IF (NSTOP .EQ. 1) GO TO 8000

GO TO 1600

END IF

DO 670 I=1,NTERM

IF (((YSOL(I) .GE. 0.001) .AND. (YSOL(I) .LE. 0.999)) THEN

GO TO 690

END IF

670 CONTINUE

C

STRATEGY 1- FOR AN INTEGER Y-VECTOR.

C

DO 675 I=1,NTERM

IF (IVECT(I) .EQ. 2) GO TO 676

675 CONTINUE

676 NARG=I

IF (NUMFR .EQ. 1) GO TO 720

DO 680 J=I+1,NTERM

IF (IVECT(J) .EQ. 2) THEN

IF (((DSOL(1)*PI(J)+DSOL(2)*PIHAT(J))*C(J) .GT.
$ (DSOL(1)*PI(NARG)+DSOL(2)*PIHAT(NARG)*C(NARG)))$ THEN

NARG=J
END IF
END IF

680 CONTINUE

GO TO 720

C
C FOR FRACTIONAL Y-VECTOR BRANCHING SELECTION.
C

690 DO 700 I=1,NTERM
    IF (IVECT(I) .EQ. 2) THEN
        IF (YSOL(I) .LE. 0.5) THEN
            FEE(I)=YSOL(I)
        ELSE
            FEE(I)=1.0-YSOL(I)
        END IF
    END IF
END IF
END IF

700 CONTINUE

DO 705 I=1,NTERM
    IF (IVECT(I) .EQ. 2) GO TO 706
END IF

706 NARG=I

IF (NUMFR .EQ. 1) GO TO 720

DO 710 J=I+1,NTERM
    IF (IVECT(J) .EQ. 2) THEN
IF (FEE(J)*C(J)*(DSOL(1)*PI(J) + DSOL(2)*PIHAT(J)) .GT. FEE(NARG)*C(NARG)*(DSOL(1)*PI(NARG) + DSOL(2)*PIHAT(NARG))) THEN
    NARG=J
END IF
END IF

710 CONTINUE

C
C UPDATING INTEGER VECTORS AND STATISTICS.
C
720 NUM=NUM+1
    NBRAN=NBRAN+1
    IPS(1,NUM)=NARG
    IPS(2,NUM)=1
    IVECT(NARG)=1
    NUMFR=NUMFR-1
    GO TO 1600
C
C FORMAT STATEMENTS AND TERMINATION
C
8000 CALL TIMECK(NTIME)
    WRITE (11,5560) 'NODES', 'NOFARC', 'NTERM', 'NBRAN', 'NGENO', 'NGBCUT'
    WRITE (11,5570) NODES, NOFARC, NTERM, NBRAN, NGENO, NGBCUT
    WRITE (11,5580) 'ELTIME'
    WRITE (11,5590) NTIME
IF (NOPT .GT. 1) WRITE (11,5580) 'FATEFF'

IF (NOPT .GT. 1) WRITE (11,5600) REAL(NFATH/NBRAN)

5500 FORMAT (88X,E16.6)
5510 FORMAT (24X,E16.6)
5520 FORMAT (1X,'FX',1X,'BOUND',5X,'Y',I4,9X,'1.0')
5530 FORMAT (1X,'FX',1X,'BOUND',5X,'Y',I4,9X,'0.0')
5540 FORMAT (56X,E16.6)
5550 FORMAT (1X,'FX',1X,'BOUND',5X,'Y',I4,9X,D12.10)
5560 FORMAT (A5,2X,A6,2X,A5,2X,A5,2X,A5,2X,A6)
5570 FORMAT (I5,2X,I6,2X,I5,2X,I5,2X,I5,2X,I6)
5580 FORMAT (1X,A6)
5590 FORMAT (I6)
5600 FORMAT (E7.2)

STOP

END

C

C SUBROUTINE FUNOBJ EVALUATES THE OBJECTIVE FUNCTION AND ITS
C GRADIENT AT A GIVEN VALUE OF X.
C
C SUBROUTINE FUNOBJ(MODE,N,X,F,G,NSTATE,NPROB,Z,NWCORE)
C
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
INTEGER NOD,NOFAR
C
PARAMETER (NOD=57, NOFAR=118)

COMMON /COMI1/ IBETA
COMMON /COMR1/ BETA,F1,BIGM,T,TAU,XBAR

DIMENSION X(N),G(N),Z(NWCore),BIGM(NOFAR),T(NOFAR),
$ TAU(NOFAR),XBAR(NOFAR)

C

C THESE ENTRY POINTS WILL NEVER BE USED BUT THIS IS ONE WAY TO
C SPECIFY THE NAMES SO THAT THERE WILL BE NO UNRESOLVED REFERENCES
C AT LOAD TIME.
C
C
C ENTRY FUNCON
ENTRY MATMOD
C
C OBJECTIVE FUNCTION EVALUATION
C

F=0.0
PART1=0.0
PART2=0.0
DO 9100 I=1,NOFAR
  PART1=PART1+T(I)*X(I)+TAU(I)*X(I)*(X(I)+XBAR(I))**IBETA
9100 CONTINUE

DO 9110 I=1,NOD
PART2=PART2+BIGM(I)*X(NOFAR+I)

9110 CONTINUE

    F=PART1+PART2
    F1=F

C

C GRADIENT EVALUATION
C

DO 9120 I=1,NOFAR

    G(I)=T(I) + BETA*TAU(I)*X(I)*(X(I)+XBAR(I))**(IBETA-1) +
    $ TAU(I)*X(I)*XBAR(I)**BETA

9120 CONTINUE

    DO 9130 I=1,NOD

    G(NOFAR+I)=BIGM(I)

9130 CONTINUE

C

C RETURN TO MAIN PROGRAM NOW.
C
RETURN
END

C

C SUBROUTINE FATHOM WILL FATHOM THE UNPRODUCTIVE NODES
C

SUBROUTINE FATHOM (IPS,NCOUN,NUMFR,NUM,NSTOP,NFATH,IVECT)
C
IMPLICIT DOUBLE PRECISION (A-H,O-Z)

INTEGER NTER

PARAMETER (NTER=15)

INTEGER IPS(2,NTER),NCOUN,NUMFR,NUM,NSTOP,IVECT(NTER),NFATH

NFATH=NFATH+1

WRITE (13,*) NFATH

NSTOP=0

IF (NUM .EQ. 0) THEN

NSTOP=1

GO TO 9299

END IF

9200 IF (IPS(2,NUM) .EQ. 2) THEN

IF (IPS(1,NUM) .LT. 0) THEN

NUMFR=NUMFR+1

IVECT(-IPS(1,NUM))=2

NUM=NUM-1

IF (NUM .EQ. 0) THEN

NSTOP=1

GO TO 9299

END IF

GO TO 9200
ELSE
NUMFR=NUMFR+1
IVECT(IPS(1,NUM))=2
NUM=NUM-1
IF (NUM .EQ. 0) THEN
   NSTOP=1
   GO TO 9299
END IF
GO TO 9200
END IF
ELSE
   IPS(1,NUM)=-IPS(1,NUM)
   IPS(2,NUM)=2
   IVECT(-IPS(1,NUM))=0
END IF
C
C RETURN TO MAIN PROGRAM NOW
C
C 9299 RETURN
END
The vita has been removed from the scanned document