REPORT NO. DOT-TSC-OST-77-39

COMPUTER CODE FOR TRANSPORTATION NETWORK DESIGN AND ANALYSIS

R.P. Harvey
D.W. Robinson

CONTROL ANALYSIS CORPORATION

800 Welch Road
Palo Alto, CA 94304

INTERIM REPORT

MAY 1977

Prepared for

U.S. DEPARTMENT OF TRANSPORTATION

OFFICE OF THE SECRETARY

Office of the Assistant Secretary for
Systems Development and Technology

Office of Systems Engineering
Washington D.C. 20590

NOTICE

This document is disseminated under the sponsorship of the Department of Transportation in the interest of information exhange. The United States Government assumes no liability for its contents or use thereof.

NOTICE

The United States Government does not endorse products or manufacturers. Trade or manufacturers' names appear herein solely because they are considered essential to the object of this report.

Technical Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
DOT-TSC-OST-77-39		

4. Title and Subtitle	5. Report Date
COMPUTER CODE FOR TRANSPORTATION NETWORK DESIGN AND ANALYSIS	May 1977
	6. Performing Organization Code

7. Author(s)	8. Performing Organization Report No.
R. P. Harvey and D. W. Robinson	DOT-TSC-OST-76-62

9. Performing Organization Name and Address	10. Work Unit No. (TRAIS)
Control Analysis Corporation* 800 Welch Road Palo Alto CA 94304	11. Contract or Grant No. DOT-TSC-1059

12. Sponsoring Agency Name and Address	13. Type of Report and Period Covered
Urban Mass Transit Administration Office of Transportation Planning Methods and Support Washington, D.C. 20590	Interim Report January 1976-October 1976
	14. Sponsoring Agency Code

15. Supplementary Notes

*Under contract to: U.S. Department of Transportation
Transportation Systems Center
Kendall Square
Cambridge MA 02142

16. Abstract

This document describes the results of research into the appliction of the mathematical programming technique of decomposition to practical transportation network problems. A computer code called CATNAP (for Control Analysis Transportation Network Analysis Program) has been developed in the course of this study; this code has the capability to solve the following problems.

1. The traffic assignment problem with fixed demands.
2. The transportation network design problem with or without a budget constraint.
3. The optimal staging problem for transportation network investments over a fixed time horizon.

In this report we describe the basic structure and algorithms employed in CATNAP and give actual numerical results obtained in some representative sample problems. These results indicated that CATNAP is an improvement over existing transportation network codes, particularly for solving the network design problem.

17. Key Words	18. Distribution Statement
Transportation Network Traffic Assignment Budget Constraint Design and Analysis	DOCUMENT IS AVAILABLE TO THE U.S. PUBLIC THROUGH THE NATIONAL TECHNICAL INFORMATION SERVICE, SPRINGFIELD, VIRGINIA 22161

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	125	

Form DOT F 1700.7 (8-72) Reproduction of completed page authorized

PREFACE

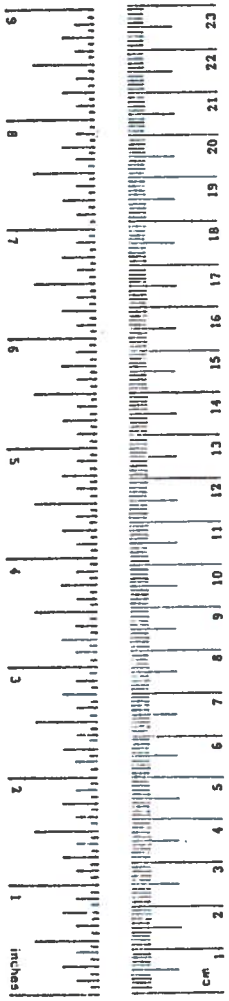
The research herein reported was funded by the Transportation Advanced Research Program (TARP) under the auspices of the Office of the Secretary, U.S. Department of Transportation. Supplemental funds were provided by the Urban Mass Transportation Administration. Technical review is the responsibility of the Special Studies Branch, Research Division, Transportation Systems Center. The objective of the TARP program is to stimulate basic scientific research in areas that are of major importance to the U.S. Department of Transportation. This particular project is intended to develop computerizable algorithms that will permit network analysis techniques to be applied to very large networks.

This report describes the implementation, testing and performance of some of the algorithms presented in an earlier Control Analysis Corporation report.

METRIC CONVERSION FACTORS

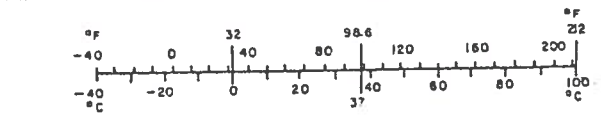
Approximate Conversions to Metric Measures

Symbol	When You Know	Multiply by	To Find	Symbol
		LENGTH		
in	inches	2.5	centimeters	cm
ft	feet	30	centimeters	cm
yd	yards	0.9	meters	m
mi	miles	1.6	kilometers	km
		AREA		
in^2	square inches	6.5	square centimeters	cm^2
ft^2	square feet	0.09	square meters	m^2
yd^2	square yards	0.8	square meters	m^2
mi^2	square miles	2.6	square kilometers	km^2
	acres	0.4	hectares	ha
		MASS (weight)		
oz	ounces	28	grams	g
lb	pounds	0.45	kilograms	kg
	short tons (2000 lb)	0.9	tonnes	t
		VOLUME		
tsp	teaspoons	5	milliliters	ml
Tbsp	tablespoons	15	milliliters	ml
fl oz	fluid ounces	30	milliliters	ml
c	cups	0.24	liters	l
pt	pints	0.47	liters	l
qt	quarts	0.95	liters	l
gal	gallons	3.8	liters	l
ft^3	cubic feet	0.03	cubic meters	m^3
yd^3	cubic yards	0.76	cubic meters	m^3
		TEMPERATURE (exact)		
°F	Fahrenheit temperature	5/9 (after subtracting 32)	Celsius temperature	°C



Approximate Conversions from Metric Measures

Symbol	When You Know	Multiply by	To Find	Symbol
		LENGTH		
mm	millimeters	0.04	inches	in
cm	centimeters	0.4	inches	in
m	meters	3.3	feet	ft
m	meters	1.1	yards	yd
km	kilometers	0.6	miles	mi
		AREA		
cm^2	square centimeters	0.16	square inches	in^2
m^2	square meters	1.2	square yards	yd^2
km^2	square kilometers	0.4	square miles	mi^2
ha	hectares (10,000 m^2)	2.5	acres	
		MASS (weight)		
g	grams	0.035	ounces	oz
kg	kilograms	2.2	pounds	lb
t	tonnes (1000 kg)	1.1	short tons	
		VOLUME		
ml	milliliters	0.03	fluid ounces	fl oz
l	liters	2.1	pints	pt
l	liters	1.06	quarts	qt
l	liters	0.26	gallons	gal
m^3	cubic meters	35	cubic feet	ft^3
m^3	cubic meters	1.3	cubic yards	yd^3
		TEMPERATURE (exact)		
°C	Celsius temperature	9/5 (then add 32)	Fahrenheit temperature	°F



CONTENTS

CONTENTS (Cont'd)

ILLUSTRATIONS

TABLES

1. INTRODUCTION AND SUMMARY

1.1 BACKGROUND

This study is concerned with solving problems which involve network models of transportation systems. Such network models tend to be quite large in practice, so that it is often computationally desirable to separate the resulting problems into simpler or smaller parts before solving them; this separation is done by means of so-called decomposition techniques.

A survey of applicable transportation network problems together with proposed decomposition algorithms for their solution have been presented in the report "The Application of Decomposition to Transportation Network Analysis" by G.B. Dantzig, S.F. Maier and Z.F. Lansdowne (reference [1]). The current report describes the implementation, testing and performance of some of these new algorithms in a computer code called CATNAP (Control Analysis Transportation Network Analysis Program). A fuller explanation of the mathematics involved in the algorithms and a summary of previous research in this area are presented in the earlier report.

The CATNAP code can solve the following types of transportation network problems:

a) Traffic assignment with fixed demands,

b) Network design with or without a budget constraint,

c) Optimal staging of network investments over time.

In this section, we briefly describe each of these problems and demonstrate the capabilities of CATNAP by summarizing the results of using the code to solve a representative problem of each type. In the remainder of the report a more detailed approach is adopted: Section 2 gives the mathematical formulation of each problem and describes the solution algorithm employed; Section

3 discusses the organization, operation and use of CATNAP; and Section 4 sets forth the actual numerical results obtained from the sample problems. The appendices contain detailed information (input formats, etc.) for using CATNAP.

1.2 TRAFFIC ASSIGNMENT PROBLEM

The objective of the traffic assignment problem is to distribute a given set of interzonal trip requirements over the links of a network in an optimum manner. This may have the goal of minimizing the total travel time for all users of the network in the face of congestion (system optimal assignment), or it may be required to find a flow pattern such that no individual traveler can decrease his time by selecting an alternate route, given that all others remain on their present paths (user equilibrium assignment); the solutions to these two problems are generally quite different, though the solution techniques used are essentially the same. Although we use the words "travel time" here and throughout this report, we recognize that there are other cost functions which may be appropriate in some applications: fuel comsumption for example, or number of accidents.

The main difficulty in solving the traffic assignment problem is that the average time for a given user to travel along a link typically depends on the number of other users of the link. Changing total link flow to alter this average time is very involved since total flow is composed of components from many different origins, and these components (termed "commodities" in this report) must be kept distinct to satisfy the trip demands in the network.

Our solution technique for this problem makes use of an algorithm for finding the shortest path through a network; when all trip demands are assigned to the links lying on the shortest path between the respective origin and destination, the resulting flow pattern is clearly feasible since all demands

are satisfied. The algorithm (which is described more fully in Section 2.2.2) generates a series of such feasible flow patterns and combines them with an existing flow pattern in order to reduce total travel time; this iterative procedure converges to the optimum solution.

A major advantage of our approach (which is basically an application of the Frank-Wolfe Algorithm ref [9]) is that individual commodity flows need not be explicitly retained; the trial flow patterns generated by the method consist only of total flows. This means that very large problems can be solved on a computer using only high-speed main memory without relying on slower peripheral storage devices (tape or disk). A further advantage is that it is easy to compute a lower bound on the optimum solution value, so that the method offers a built-in stopping criterion. Finally, the algorithm can accommodate a wide variety of nonlinear travel time functions; in fact, nonlinear differentiable functions work better here than the commonly used piecewise linear approximations.

Considerable computational experience with the Frank-Wolfe Algorithm has generally confirmed the above advantages; one shortcoming has been noted, however. This is the slow convergence of the method to the final optimum solution. The first few trial flow patterns typically improve the starting solution substantially, but later iterations change the solution very little. Perhaps 15 to 20 iterations are needed in large problems to get within 3 to 5 percent of the optimum, but reaching 1 percent may require 40 to 50 iterations. See Section 4.2 for an example of this behavior.

In summary, quite large traffic assignment problems (say, 5000 nodes and 15000 links) are within the capability of CATNAP. Solutions within 3 percent of the optimum can be obtained with a fairly modest expenditure of computer resources.

1.3 NETWORK DESIGN PROBLEM

An efficient way of solving the network design problem is the major new capability offered by CATNAP. This problem involves the improvement of links in a transportation network through investment; the goal may be either to minimize total travel time subject to a budget constraint or else to minimize a weighted combination of investment and travel time.

This problem is difficult to solve because improvement of one link typically affects flow on many other links; adjusting the flow pattern to even a single improvement thus involves the same kind of difficulties as the traffic assignment problem. Furthermore, the effects of different investments interact with one another thus adding another degree of complexity to the problem. The usual method used has been to perform a series of traffic assignment problems with differing manually determined investment "scenarios"; this offers no guarantee that an optimum solution will even be approached.

The method employed in CATNAP to solve the network design problem is to transform the problem into an equivalent traffic assignment problem in the case when there is no budget constraint, and a sequence of traffic assignment problems when there is a budget constraint. The Frank-Wolfe method described in the previous section is then used to solve the problems. The link congestion functions for those links with investment opportunities now become more complex. There is a component corresponding to traffic congestion, and a component associated with the investment level. This composite function results from determining the optimal investment for each link as a function of the total flow on the link. It reflects the optimal trade-off between increase of investment and decrease of congestion using a given current conversion factor.

For the network design problem with no budget constraint the determination of the optimum link investment depends on minimizing a weighted combination of travel time and investment dollars. The weighting factor for the investment expresses the conversion between dollars and travel delay and appears directly in the objective function. This problem requires about the same effort to solve as a traffic assignment problem.

The other form of the network design problem (with budget constraint) is somewhat more difficult to solve. In this case, the problem is solved with a sequence of values for the weighting factor which now plays the role of a Lagrange multiplier. This multiplier is successively adjusted until the sum of all the link investments is close to the required total budget. It has been found that this procedure requires approximately seven different multiplier values, and takes about three times as long to solve as a comparable traffic assignment problem.

CATNAP provides a large number of user options for solving the network design problem, including solution tolerances, output reports and post-optimality adjustments. Besides being more flexible, the code is several orders of magnitude faster than the best previous effort in this area, which used a linear programming formulation of the problem with piecewise linear approximations to the actual nonlinear curves. It is estimated, that the present version of CATNAP can solve a network design problem for a 2000-node, 6000-arc network in less than 30 minutes of processor time on an IBM 370/168 computer.

1.4 INVESTMENT STAGING PROBLEM

When a major improvement project is to be undertaken in an existing transportation network, it will often take a long time to complete. In this case, it may well be desirable to improve some links earlier than others;

the investment staging problem thus seeks the optimum sequence in which to improve the links at given stages over a fixed-time horizon.

Many formulations of this problem are intractably large for practical transportation networks because they require the simultaneous solution of several network design problems which are linked together. The formulation used in CATNAP, and described in Section 2.4.1, allows the investment staging problem to be solved as a sequence of essentially independent design problems with a resultant saving of problem size and solution time.

Several heuristic methods have been devised for this problem. While CATNAP procedures give uniformly better solutions than the other techniques, the solutions are close enough that the somewhat greater computer execution time required by CATNAP may be a factor favoring the heuristic approaches.

1.5 POTENTIAL APPLICATIONS

The CATNAP code is intended primarily to assist a transportation planner in making the best use possible of limited funds for the improvement of a large transportation system. Because the interactions of traffic flows in such systems are so complex, it is believed that a mathematical solution from CATNAP will be a very valuable aid to the planner's intuition, which alone can be misleading.

The main application area for CATNAP as it is presently configured is for urban highway planning. The use of network models is fairly well established in this area and planners are familiar with the mathematical evaluation of improvement proposals. CATNAP cannot explicitly allow for city politics, pollution, planning boards and many other constraints which are very real to an urban planner. For this reason, it is recommended that the initial CATNAP improvement plan be treated as a "best case" and that the other considerations

be imposed in later runs using the code's bound-setting capability (see equation (19) and Section 3.2). This interactive use of CATNAP is made both convenient and economical by the code's restart mechanism which allows subsequent runs to begin the solution where the initial run left off.

Some modifications to CATNAP will be needed to enable the code to solve the modal split problem, i.e., selecting which transit method (of several) each commuter chooses and determining the resulting traffic patterns. The modifications would involve both the data input modules of CATNAP and the link congestion functions used, but the code itself is flexible enough to solve this important urban problem without major reworking.

CATNAP will require the same type of modifications (i.e., data input and congestion functions) to be applied in areas other than urban planning. The use of network models is not nearly so well established in railroad or airport applications (to give two examples) but there is no reason that CATNAP could not be applied in these areas. For example, many rail links have low speed limits (called "slow orders") due to poor track maintenance. CATNAP could be used to determine an optimum plan to minimize travel time by spending maintenance dollars to remove the slow orders.

CATNAP has been designed as a highly efficient and convenient computer program. Its speed and computer memory requirements (see Section 3.5) are comparable to those of the UTPS code UROAD and its modular design facilitates any necessary modifications to deal with different classes of problems.

1.6 POTENTIAL EXTENSIONS

CATNAP, although originally envisioned as an exploratory tool, is both efficient and flexible enough to be used as a production code. The code is organized in a highly modular fashion and improvements are relatively easy to carry out.

Several possible extensions and modifications are outlined in Section 3.6 which could be implemented in CATNAP. They fall into one or more of the following four categories:

a) Broaden the capabilities of the code; for example, allow the use of alternative travel time and investment functions, "negative" improvements in network design, or several independent budget constraints.

b) Increase the size of problem which can be handled by such techniques as dynamic core allocation and external storage for trip table data.

c) Improve the efficiency of the code by exploring algorithm variations in the Frank-Wolfe procedure, the golden section search technique, the shortest path algorithm and the network design subproblem solution method.

d) Enhance user convenience by making the input and output options more flexible.

2. PROBLEM FORMULATIONS AND ALGORITHMS

2.1 INTRODUCTION

In this section, we describe the problems which the CATNAP code is designed to solve, and we summarize the solution techniques which are employed. This discussion here is intended to be fairly specific with respect to what has actually been implemented and tested, in contrast to reference [1] which sets forth a general framework for applying decomposition techniques to transportation network problems. See Section 3.6 for an account of some areas in which the work reported here can be extended with a relatively small amount of effort.

For each of the three major problem types (traffic assignment, network design and investment staging), we give an annotated mathematical formulation of the problem together with some implementation details. This is followed by a description of the algorithm used in CATNAP to solve the problem. A more general account of these matters is given in the earlier report [1].

2.2 TRAFFIC ASSIGNMENT PROBLEM

The links* in practical transportation networks are subject to congestion as the volume of traffic increases. For this reason, the determination of the best routes to be taken by all the users of a network is a difficult problem: the delay encountered on a given link depends on the total flow of traffic, which means that interactions between travelers from different origins must be explicitly accounted for. Finding a traffic routing which minimizes total time (the so-called "system optimal" problem), or for which no user can

*The terms "link" and "arc" are used interchangeably in this report.

find an alternate route with a lower time while keeping the routes for other travelers fixed (the "user equilibrium" problem), constitutes what is known as the traffic assignment problem.

The decomposition approach taken to this problem allows one to determine for each origin in the network a trial solution that is independent of the flows from all the other origins. The interactions between different origins are then managed by seeking an optimal combination of the trial solution with an existing solution. The procedure is then repeated for a new trial solution until a satisfactory answer is obtained.

The traffic assignment algorithm described here is at the heart of the CATNAP code, and it is used extensively in the solution of network design problems (see Section 2.3). In the remainder of this section, we give a mathematical formulation of the traffic assignment problem and a brief description of the solution method.

2.2.1 Problem Formulation

The number of distinguishable flow commodities in the traffic assignment problem is equal to the number of origins, the number of destinations, or the number of origin-destination pairs, depending upon how it is formulated. In the following formulation, we distinguish commodities by origin node.

$$\text{MINIMIZE} \quad Z = \sum_{j \in A} T_j(f_j) \ , \tag{1}$$

with respect to: f_j^r and f_j , $j \in A$, $r = 1, \ldots, R$.

Subject to

$$\sum_{j \in W_i} f_j^r - \sum_{j \in V_i} f_j^r = h_i^r \ , \qquad (i \in N;\ r = 1, \ldots, R), \tag{2}$$

$$f_j = \sum_{r=1}^{R} f_j^r \ , \qquad (j \in A) \ , \tag{3}$$

$$f_j^r \geq 0 \ , \qquad (j \in A;\ r = 1, \ldots R), \tag{4}$$

where

A = the set of links in the network ,

f_j = total flow on link j ,

$T_j(f_j)$ = total travel cost on link j when flow is f_j ,

f_j^r = flow on link j from origin r ,

R = the number of origin nodes in the network,

$$h_i^r = \begin{cases} -O_{ij} & \text{if } i \text{ is a destination node,} \\ \sum_{j \in A} O_{rj} & \text{if } i = r \ , \\ 0 & \text{otherwise,} \end{cases}$$

O_{ij} = the number of trips originating at origin i and terminating at destination j ,

N = the set of nodes in the network,

V_i = the set of links terminating at node i ,

W_i = the set of links originating at node i .

The definition of cost functions $T_j(\cdot)$ depends on whether a user equilibrium or system optimal traffic assignment problem is being considered. In accordance with the results summarized in [1], we first define the average cost function

$C_j(f_j)$ = average (per unit) time for travelers on link j when the flow is f_j.

The objective function component for link j for system optimal assignment is

$$T_j(f_j) = C_j(f_j) \cdot f_j \quad , \tag{5}$$

and for user equilibrium assignment is

$$T_j(f_j) = \int_0^{f_j} C_j(z)\, dz \quad . \tag{6}$$

The average cost function $C_j(\cdot)$ may be chosen to have many different functional forms as long as it is non-negative, continuous and a non-decreasing function of f_j. In CATNAP, a simple polynomial has been chosen which leads to the travel cost function $T_j(\cdot)$ commonly used by the Federal Highway Administration (FHWA). The FHWA curve for system optimal assignment is given by

$$T_j(f_j) = t_j f_j \left[1 + r \left(\frac{f_j}{CA_j}\right)^k\right] \quad , \tag{7}$$

where

t_j = free-flow travel time for link j (positive),

CA_j = capacity parameter for link j (positive),

r = positive constant (FHWA uses 0.15),

k = positive constant (FHWA uses 4) .

From (6), the FHWA curve for user equilibrium assignment is

$$T_j(f_j) = t_j f_j \left[1 + \frac{r}{k+1} \left(\frac{f_j}{CA_j}\right)^k\right] \quad . \tag{8}$$

Note that although the objective function value for the user equilibrium problem is computed from (8), the actual travel time is still given by (7). Note further that the only difference between (7) and (8) is the factor $1/(k + 1)$ in the congestion term, and that this may be accounted for by simply multiplying the link capabity CA_j by a factor of $(k + 1)^{1/k}$. (This factor is 1.495 for $k = 4$.) The behavioral implication of this analysis is that under the user equilibrium objective individual travelers are approximately 50 percent less congestion-averse than a system optimal traffic assignment would indicate.

Previous researchers have often used a piecewise linear cost function for link travel time, and CATNAP also retains this capability mainly for use in certain network design applications. If there are M_j linear segments associated with the curve for link j, we replace (3) with

$$\sum_{m=1}^{M_j} X_j^m = \sum_{r=1}^{R} f_j^r , \qquad (j \in A), \qquad (3')$$

and we add the upper bound constraints

$$0 \leq X_j^m \leq K_j^m , \qquad (j \in A;\ m = 1, \ldots, M_j) . \qquad (9)$$

In practice, we take $K_j^{M_j}$ to be infinite. Now if the slope segment m in the cost curve for link j is C_j^m, we replace the objective function (1) by

$$\text{MINIMIZE} \quad Z = \sum_{j \in A} \sum_{m=1}^{M_j} C_j^m \, X_j^m \qquad (1')$$

This formulation may be used for either user equilibrium or system optimal problems by an appropriate choice of K_j^m and C_j^m. Referring to the FHWA curve (7), the segment lengths K_j^m and the slopes C_j^m depends on the capacity parameters and the free-flow travel time parameters, respectively; this may be seen from Eqs. (29) and(30) in Section 2.3.1. Throughout this report we assume that X_j^m's are ordered so that $C_j^m < C_j^n$ for $m < n$.

Although the new problem (1'), (2), (3'), (4), (9) is a linear program it is no easier to solve by our method than the nonlinear program (1) through (4); furthermore, the new problem is so much larger than the nonlinear one that it is beyond the capability of general-purpose linear programming codes except for very small networks (see Section 4.3). It is thus anticipated that the main use of this formulation will be for network design situations as described in Section 2.3, and that most links in the network will use the appropriate FHWA curve [(7) or (8)].

2.2.2 Solution Algorithm

The traffic assignment problem solution technique must begin with an initial feasible flow pattern; i.e., a set of f_j^r values satisfying (2) and (4). A simple way to find such a pattern is to assign all trips between each origin and destination to the arcs lying on the path with the smallest total free-flow travel time. For a given origin, the determination of a set of minimum cost paths to all the destinations is called the shortest path problem; an initial feasible solution to the traffic assignment problem may thus be found by solving a shortest path problem for each origin in the network using the parameter t_j (free-flow travel time) as the cost for each link j.

For the shortest path problem, the basic algorithm due to Dijkstra [2] is used with a binary-tree structure. This approach offers several important computational advantages over other common implementations of the Dijkstra method. For a complete description of the method together with an analysis of its performance, see reference [5].

Once a feasible flow pattern (referred to as F1) has been found, a secon feasible flow pattern (F2) is generated by the algorithm. This is done in the same way that the initial pattern was found; i.e., by solving a set of shortest path problems. In this case however, the cost of arc j is taken to be the marginal cost of flow on the arc, $T_j'(F1_j)$. Thus, arcs with very high congestion present a great deal of impedance to the shortest path routine, and the trial-flow pattern F2 is likely to result in reduced flow on these arcs.

Setting up the shortest path problems with marginal cost is equivalent to taking a linear approximation to the objective function (1) at the current flow values F1, while the shortest path problems themselves constitute a linear program. This technique is called the Frank-Wolfe Algorithm [9]. Note that the linear program is particularly easy to solve because each origin can be treated independently of the others; i.e., we have successfully decomposed the problem by origin.

A particular advantage of using the Frank-Wolfe Algorithm is that a lower bound on the objective function value in (1) may be obtained quite easily Also, since F1 is a feasible flow pattern the current total travel time is an upper bound on the optimum value of Z. Denoting this optimum by Z*, we have

$$\sum_{j\in A}\left[T_j(F1_j) + T_j'(F1_j)(F2_j - F1_j)\right] \leq Z^* \leq \sum_{j\in A} T_j(F1_j) \quad . \tag{10}$$

It has been found that (10) may be used as a most reasonable criterion for terminating the algorithm.

The third phase of the Frank-Wolfe Algorithm is to combine the two feasible flow patterns F1 and F2 in an optimum way. Formally, we want to solve the one-dimensional optimization problem

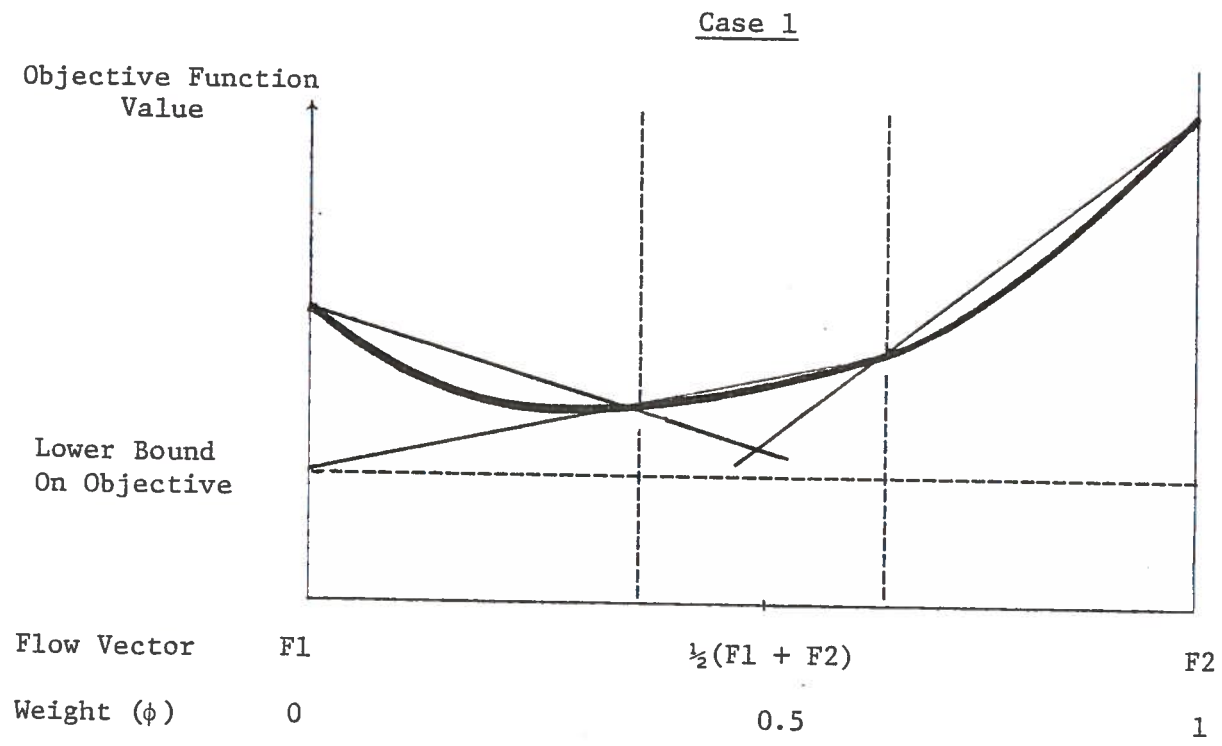
$$\text{MINIMIZE} \quad Z(\phi) = \sum_{j \in A} T_j\left[(1 - \phi)F1_j + \phi F2_j\right] \quad , 0 \leq \phi \leq 1 \quad . \quad (11)$$

For the FHWA cost function (7), the problem (11) is a polynomial in ϕ which is, furthermore, a convex function of ϕ . The most general method for solving (11) is golden section search [3], and this technique is used in CATNAP. It is an iterative method, and in [4] a lower bound on the objective value is derived based on the convexity property of the objective function. This is used in CATNAP as a termination criterion for the golden section search. The two general cases and the corresponding derived lower bounds are illustrated in Figure 2.1.

The final result of the optimization (11) is a value of ϕ^* which is used to compute a new feasible flow pattern from

$$F1'_j = (1 - \phi^*)F1_j + \phi^* F2_j \quad , \qquad (j \in A) \quad . \qquad (12)$$

The Frank-Wolfe Algorithm is then repeated using $F1'$ in place of F1 . A summary of the method as applied here is given by the following steps:



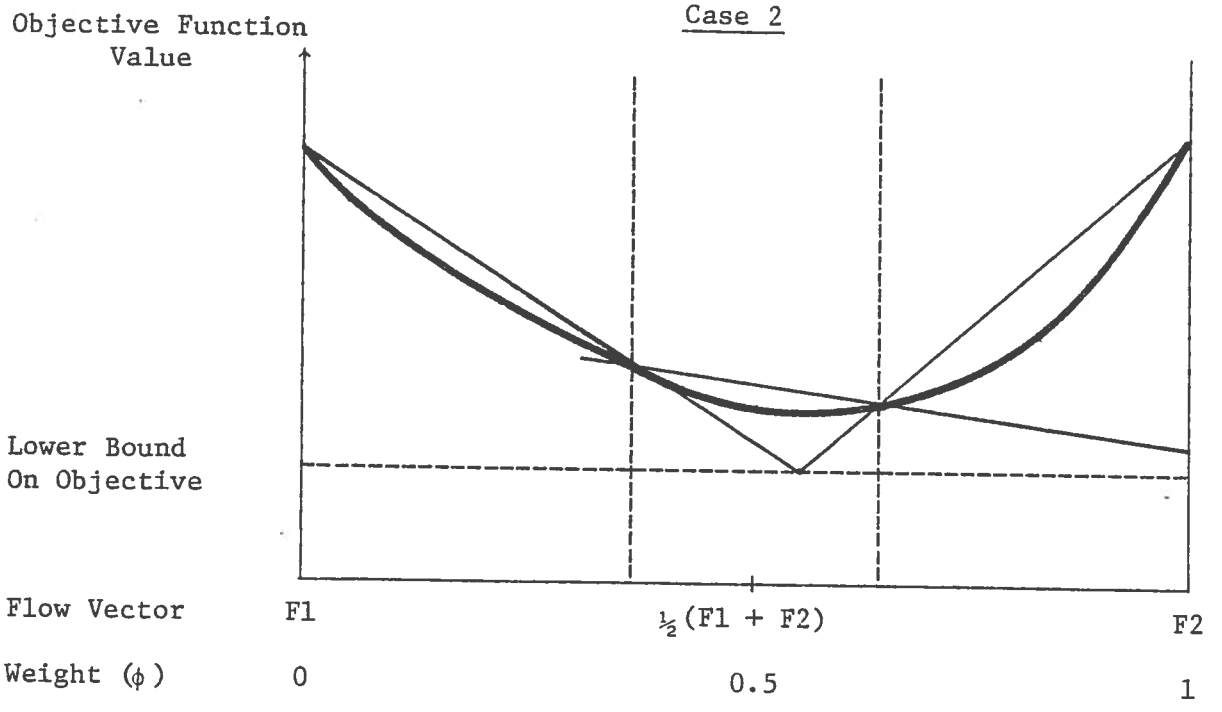


Figure 2.1 - Lower Bound on Objective in Golden Section Search

Step 1. Generate a feasible solution F1. Set the lower bound to 0.

Step 2. Evaluate the marginal costs $T_j'(F1_j)$, and use these with the shortest path routine to develop a new feasible flow pattern F2.

Step 3. Compute a new lower bound from (10). If this is greater than the current lower bound, replace the current bound with the new one.

Step 4. Solve the optimization problem (11), and replace F1 with F1′ computed from (12).

Step 5. If $\phi^* = 0$ or if the lower bound (10) is close enough to the current objective value, stop. Otherwise, go back to Step 2.

The main difficulty in applying this algorithm for the $T_j(\cdot)$ functions described in Section 2.2.1 occurs for the piecewise linear formulation (1′). The Frank-Wolfe method requires that the objective function be differentiable, and this is not the case for (1′) since $T_j(\cdot)$ is not differentiable at the breakpoints of the piecewise linear curve. Accordingly, a small positive parameter ϵ is used to "round the corners" as illustrated in Figure 2.2. The derivative of $T_j(\cdot)$ is thus assumed to vary linearly over each of the intervals $\left(\sum_{i=1}^{m} K_j^i - \epsilon \;,\; \sum_{i=1}^{m} K_j^i + \epsilon\right)$

As far as the derivative is concerned, the total time function is then

$$T_j(f_j) = \sum_{i=1}^{m} C_j^i K_j^i + C_j^m \left(f_j - \sum_{i=1}^{m} K_j^i\right) + \left[\left(C_j^{m+1} - C_j^m\right)/4\epsilon\right] \cdot \left[f_j - \left(\sum_{i=1}^{m} K_j^i - \epsilon\right)\right]^2 ,$$

for

$$\sum_{i=1}^{m} K_j^i - \epsilon \le f_j \le \sum_{i=1}^{m} K_j^i + \epsilon \quad ; \tag{13}$$

It should be noted, however, that $T_j(\cdot)$ is still computed from (1′).

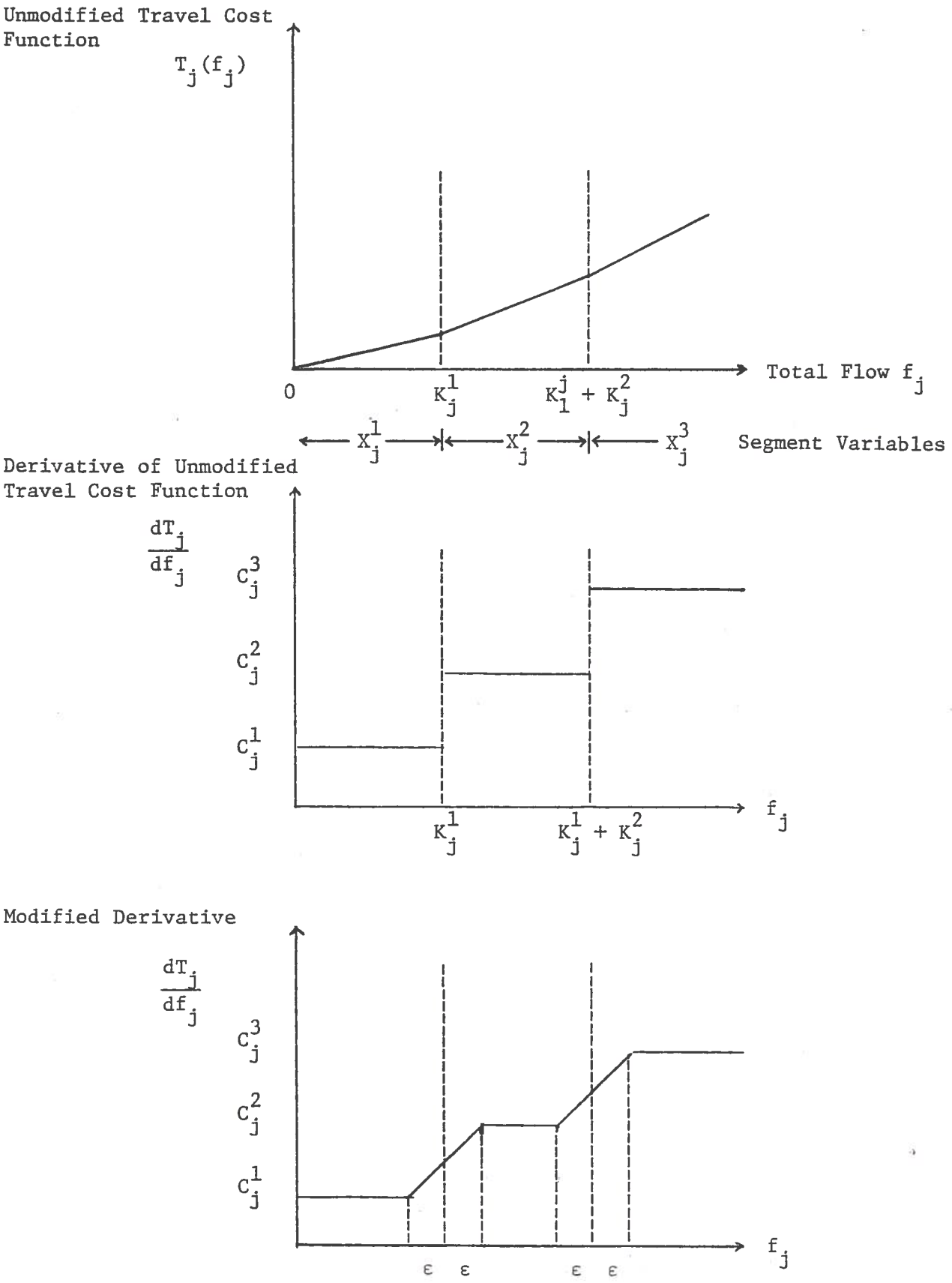


Figure 2.2 - Modicication of Piecewise Linear Travel Time Function

2.3 NETWORK DESIGN PROBLEM

The network design problem is concerned with adding or modifying links in a transportation network through the investment of money. The objective may be either to minimize some weighted combination of total travel time and invested funds or simply to minimize total travel time subject to a budget constraint. CATNAP has the capability to solve either of these formulations of the problem.

The basic idea of the network design algorithm is to redefine the link cost function to include both the amount of money spent on improving the link and also the effect of the improvement on the total travel time. When this is done the optimum investment for each link may be found as a function of only the flow on the link independent of any other investments. It turns out that the cost function is still a convex function of flow, so that the Frank-Wolfe traffic assignment algorithm described in Section 2.2.2 may still be used to find the optimum flow pattern. A budget constraint may be imposed by adjusting the relative contribution of investment to the link cost function until the correct expenditure ensues.

The decomposition of the network design problem by links in this manner greatly facilitates its solution. It is necessary, however, to use a system optimal traffic assignment when this is done and also to require that investment decisions be continuous rather than discrete. See Sections 2.3.5 and 2.3.6 for further discussion of these restrictions and for an account of methods for dealing with them.

2.3.1 Problem Formulation -- Without Budget Constraint

An important step in obtaining a mathematical formulation of the network design problem is to treat the interaction among travel time, flow and investment. Let $D_j(f_j, z_j)$ be the total travel time on link j as a function of the flow f_j and the investment decision z_j for that link, and let $G_j(z_j)$ be the cost of the decision z_j. The multiplier λ is assumed to represent the conversion factor between investment dollars and travel time. The total social transportation cost (user travel time plus investment) for link j is given by

$$D_j(f_j, z_j) + \lambda G_j(z_j) \quad . \tag{14}$$

The network design problem is then formulated as follows:

$$\text{MINIMIZE} \quad Z = \sum_{j \in A} \left[D_j(f_j, z_j) + \lambda G_j(z_j) \right] \quad , \tag{15}$$

with respect to f_j, f_j^r, z_j $(j \in A;\ r = 1, \ldots, R)$ subject to

$$\sum_{j \in W_i} f_j^r - \sum_{j \in V_i} f_j^r = h_i^r \quad , \qquad (i \in N;\ r = 1, \ldots, R) \quad , \tag{16}$$

$$f_j = \sum_{r=1}^{R} f_j^r \quad , \qquad (j \in A) \quad , \tag{17}$$

$$f_j^r \geq 0 \ , \qquad (j \in A;\ r = 1, \ldots, R) \ , \qquad (18)$$

$$L_j \leq z_j \leq P_j \ , \qquad (j \in A) \ . \qquad (19)$$

The notation in (15) through (19) is identical to that used in (1) through (4) with the addition of the following:

$D_j(f_j, z_j)$ = total travel time on link j as a function of flow f_j and investment decision z_j .

$G_j(z_j)$ = cost of decision z_j on link j .

z_j = investment decision for link j measured in units of capacity.

λ = non-negative multiplier expressing the conversion between travel time and investment dollars.

L_j = minimum value of the investment decision for link j .

P_j = maximum value of the investment decision for link j .

Of the many possible choices for $D_j(\cdot,\cdot)$ and $G_j(\cdot)$, we are restricted by the solution algorithm to those which result in a convex cost function for each link. To simplify the discussion here, we will consider only the three options which are provided in the CATNAP code itself, while recognizing that numerous other possibilities exist. See reference [1] for further discussion of this point.

When the FHWA travel time curve (7) is used, it is assumed that the effect of investment is to increase the capacity parameter CA_j, and that the investment

function $G_j(\cdot)$ is linear. We choose z_j to be measured in units of capacity increase. This then results in

$$D_j(f_j, z_j) = t_j f_j\left[1 + r\left(\frac{f_j}{CA_j + z_j}\right)^k\right] , \tag{20}$$

$$G_j(z_j) = g_j z_j \quad . \tag{21}$$

These functions result in a convex cost function for the arcs as long as $CA_j + z_j > 0$; note that this permits "negative" investments; i.e., monetary benefits resulting from a decrease in link capacity.

The second and third options in CATNAP use the piecewise linear formulation of Section 2.2.1. In both cases, $G_j(\cdot)$ is assumed to be a piecewise linear convex function, and the effect of the investment is assumed to vary the lengths of the segments in the $T_j(\cdot)$ curve. Thus we have

$$D_j(f_j, z_j) = \underset{X_j^m}{\text{MIN}} \sum_{m=1}^{M_j} C_j^m X_j^m \quad , \tag{22}$$

subject to

$$f_j = \sum_{m=1}^{M_j} X_j^m \quad , \tag{23}$$

$$0 \leq X_j^m \leq K_j^m + F_j^m z_j \quad , \quad (m = 1, 2 \ldots M_j - 1), \tag{24}$$

$$0 \leq X_j^{M_j} \quad . \tag{25}$$

Also,

$$G_j(z_j) = \sum_{n=1}^{N_j} g_j^n z_j^n \quad , \tag{26}$$

subject to

$$z_j = \sum_{n=1}^{N_j} z_j^n \quad , \tag{27}$$

$$0 \leq z_j^n \leq G_j^n \quad , \qquad (n = 1, \ldots, N_j) \quad , \tag{28}$$

where F_j^m is a factor determining the change of the length of the m-th cost curve segment with respect to the investment decision, and N_j is the number of linear segments in the investment function for link j. The effect of the investment decision z_j is thus to change the length of segment m of the travel time curve from K_j^m to $K_j^m + F_j^m z_j$.

A piecewise linear travel cost curve may arise from a variety of underlying functional forms for $T_j(\cdot)$. If it is an approximation to the FHWA curve (7), we take the abscissa values for the curve at multiples of the capacity parameter CA_j; i.e. at $\alpha_1 CA_j$, $\alpha_2 CA_j$, . . . , $\alpha_m CA_j$, and choose the ordinates to lie exactly on the FHWA curve. The slopes are then given by

$$C_j^m = t_j \frac{[\alpha_m(1 + r\alpha_m^k) - \alpha_{m-1}(1 + r\alpha_{m-1}^k)]}{\alpha_m - \alpha_{m-1}} \quad , \tag{29}$$

and the breakpoints by

$$K_j^m = (\alpha_m - \alpha_{m-1})\, CA_j \quad , \quad (m = 1, \ldots, M_j) \quad , \tag{30}$$

where α_0 is taken to be 0. For a fixed set of multipliers $\{\alpha_m\}$ then, the segment lengths for a link are functions of the capacity parameters CA_j, and the slopes are functions of the free-flow travel time t_j. An unbounded final segment ($K_j^{M_j} = \infty$) would be represented with an arbitrarily large slope $C_j^{M_j}$.

Suppose now that the effect of investment is to increase the link capacity parameter as in (20). This is the second investment option in CATNAP; since t_j does not change, the segment slopes are unaffected, but the capacity increase to $CA_j + z_j$ must be reflected according to (24) by selecting

$$F_j^m = \alpha_m - \alpha_{m-1} \quad (m = 1, \ldots, M_j). \tag{31}$$

Note that this is essentially the same problem as in (20) and (21), except for the more general form of $G_j(\cdot)$ that (26) to (28) permit in this case.

The third option in CATNAP involves investments which decrease the free-flow travel time; say, from t_j to t_j'. A simultaneous increase of capacity from CA_j to CA_j' is also allowed, but it is possible for $CA_j = CA_j'$. Here we choose the investment decision z_j to be measured in units of capacity, so that $0 \leq z_j \leq CA_j'$. Now (24) does not permit alteration of the slopes C_j^m in the piecewise linear curve. Thus, if the $D_j(f_j, z_j)$ function is to reflect properly the post-investment situation, segments with slopes corresponding to the new t_j' value must be present. These are introduced into the curve and interleaved with those for the existing t_j value; in order that this augmented curve also correspond to the zero investment case, we select segment lengths $K_j^m = 0$ for the new segments.

The effect of investment in the travel time improvement option of CATNAP is thus to cause the growth of the zero-length segments corresponding to the new time t_j' . For these segments, then, we choose the multipliers

$$F_j^m = \alpha_m - \alpha_{m-1} , \tag{32}$$

so that the post-investment curve has the correct breakpoints for the new capacity CA_j' . Of course, the segments of the $T_j(\cdot)$ curve for the old travel time t_j cannot be left in the post-investment curve; for this reason, we define the <u>negative</u> multipliers

$$F_j^m = -(\alpha_m - \alpha_{m-1})\ CA_j/CA_j' , \tag{33}$$

so that the constraint (24) causes these old segments to vanish when the improvement is complete.

Note that in this third option the definition of the improvement z_j is somewhat different than for the first two. In all cases, we have taken z_j to be measured in units of capacity (vehicles/hour, passengers/day, etc.), but for the first two options it is interpreted as units of increased capacity, while for the third we may think of it as units of the total final capacity which have been introduced. The maximum improvement in cases 1 and 2 is just $CA_j' - CA_j$, while for case 3 it is CA_j'. It should also be noted that when a partial improvement is implemented in case 3, that is when $0 < z_j < CA_j'$, the effective travel cost curve is a linear combination of the two extreme travel cost curves.

2.3.2 Solution Algorithm -- Without Budget Constraint

The solution technique developed in [1] for the network design problem (15) through (19) requires that a separate optimization be performed for each link; the goal of this optimization is to select the investment decision z_j which minimizes the total social transportation cost for the current flow f_j on the arc. The resultant cost is then

$$H_j(f_j) = \min_{z_j}[D_j(f_j, z_j) + \lambda G_j(z_j)] , \tag{34}$$

subject to

$$L_j \leq z_j \leq P_j . \tag{35}$$

The optimum investment decision z_j^* depends on the flow f_j; thus we define the function

$$I_j(f_j) = \text{value of } z_j \text{ for which } H_j(f_j) \text{ attains its minimum.} \tag{36}$$

The objective function (15) for the network design problem may now be replaced by

$$\text{MINIMIZE } Z = \sum_{j \in A} H_j(f_j) . \tag{15'}$$

It should be apparent that the problem (15'), (16) through (18) is identical to the traffic assignment problem (1) to (4) except for the cost function. However, as long as $H_j(\cdot)$ is convex, we may still use the algorithm set forth in Section 2.2.2 to determine the optimum flows for the network design problem.

The original problem has thus been decomposed into a set of subproblems, one for each link. These subproblems choose the optimum investments to be made, while the master problem, i.e., the traffic assignment problem, adjusts the network flows to achieve an overall system optimal solution. The final set of optimal investment decisions may be obtained from the final flows f_j and the functions $I_j(\cdot)$.

The success of this approach to the network design problem depends upon how easily the functions $I_j(\cdot)$ and $H_j(\cdot)$ can be computed, and if, in fact, $H_j(\cdot)$ is a convex function of total link flow. We consider these questions in turn for each of the three investment formulation options described in the previous section.

For the FHWA curve- case (20) and (21), the subproblems (34) and (35) become

$$H_j(f_j) = \min\left\{ t_j f_j \left[1 + r\left(\frac{f_j}{CA_j + z_j} \right)^k \right] + \lambda g_j z_j \right\} . \qquad (37)$$

The solution of this problem is:

$$I_j(f_j) = \begin{cases} L_j , & 0 \le f_j \le (CA_j + L_j)\ \phi_j(\lambda) , \\ \phi_j(\lambda)^{-1}\ f_j - CA_j , & (CA_j + L_j)\,\phi_j(\lambda) \le f_j \le (CA_j + P_j)\,\phi_j(\lambda) , \\ P_j , & f_j \ge (CA_j + P_j)\ \phi_j(\lambda) , \end{cases} \qquad (38)$$

$$H_j(f_j) = \begin{cases} t_j f_j \left[1 + r\left(\dfrac{f_j}{CA_j + L_j}\right)^k\right] + \lambda g_j L_j, & 0 \le f_j \le (CA_j + L_j)\,\phi_j(\lambda), \\ t_j f_j \left[1 + (k+1)\, r\, \phi_j(\lambda)^k\right] - \lambda g_j\, CA_j, & (CA_j + L_j)\,\phi_j(\lambda) \le f_j \le (CA_j + P_j)\,\phi_j(\lambda), \\ t_j f_j \left[1 + r\left(\dfrac{f_j}{CA_j + P_j}\right)^k\right] + \lambda g_j P_j, & f \ge (CA_j + P_j)\,\phi_j(\lambda), \end{cases} \tag{39}$$

where $\phi_j(\lambda)$ is constant for a fixed value of λ, and is given by

$$\phi_j(\lambda) = \left[\frac{\lambda g_j}{krt_j}\right]^{\frac{1}{k+1}} . \tag{40}$$

As discussed in [1], $H_j(\cdot)$ is a convex function in this case, and both $H_j(\cdot)$ and $I_j(\cdot)$ are quite easy to evaluate for fixed λ .

The determination of $I_j(\cdot)$ and $H_j(\cdot)$ is somewhat more involved for the two piecewise linear formulations of the previous section. Since both $D_j(\cdot,\cdot)$ and $G_j(\cdot)$ are piecewise linear in this case, both $I_j(\cdot)$ and $H_j(.)$ are piecewise linear as well; thus, the two optimal functions need be determined just once for a given value of λ and thereafter may be used in the traffic assignment procedure. We now describe briefly the approach taken to find these functions.

We note first that the optimum z_j is an increasing (although non-convex) function of flow. Suppose, then, that at some flow value f_j, an optimal investment decision z_j has been made, and that

$$f_j = \sum_{i=1}^{m} (K_j^i + F_j^i z_j) , \tag{41}$$

and

$$Z_j = \sum_{i=1}^{n} z_j^i \,, \tag{42}$$

i.e., both f_j and z_j are at the ends of linear segments m and n on the $D_j(\cdot,\cdot)$ and $G_j(\cdot)$ curves, respectively. Now if it is possible to make further investments (i.e., $n < N_j$), there are two choices available as f increases: either one may extend the current $T_j(\cdot)$ segment by investing, or one may go on to the next T_j segment. The unit cost of the next segment is, of course, just C_j^{m+1}, so the $H_j(\cdot)$ function will have a slope of C_j^{m+1} in the second case. In the first case, an investment of one unit contributes a cost of λg_j^{n+1} to $H_j(\cdot)$. In addition, there are contributions from all segments 1, 2, ..., m-1 which are now changed in length. The $H_j(\cdot)$ function slope in this case is

$$\left(\sum_{i=1}^{m} C_j^i + \lambda g_j^{n+1}\right) \Big/ \sum_{i=1}^{m} F_j^i \quad . \tag{43}$$

The decision to invest or not may be made by comparing C_j^{m+1} with the expression (43). Thus, a linear segment of the $H_j(\cdot)$ function is created. Its length depends on the length of the (n + 1)-st investment segment in case one, or the length of the (m + 1)-st travel cost segment in case two. The process is then repeated as is illustrated by Figure 2.3.

In [1], it is established that $H_j(\cdot)$ is a convex function of flow in this case. Note, however, that to apply the Frank-Wolfe Algorithm, it is necessary to use the "corner-rounding" method of Section 2.2.2 because the function $H_j(\cdot)$ is not differentiable at the breakpoints.

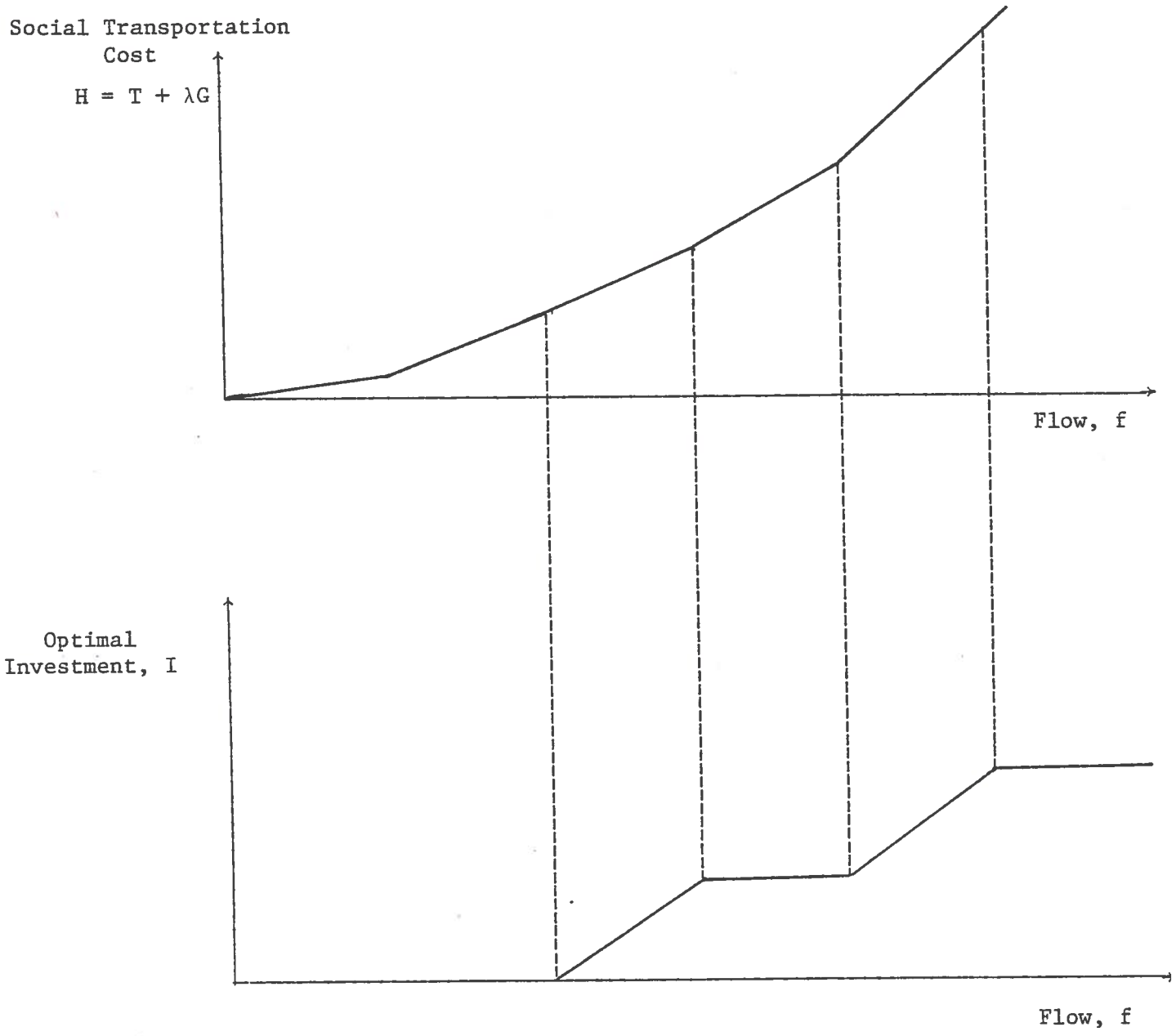


Figure 2.3 - Optimal Cost and Investment Functions for Piecewise Linear Formulations

2.3.3 Problem Formulation -- With Budget Constraint

In this form of the network design problem we wish to minimize the total travel time of all users in the system subject to an upper bound on total investment:

$$\text{MINIMIZE} \quad Z = \sum_{j \in A} D_j(f_j, z_j) \quad , \tag{44}$$

subject to

$$\sum_{j \in W_i} f_j^r - \sum_{j \in V_i} f_j^r = h_i^r \quad , \qquad (i \in N;\ r = 1, \ldots, R) \quad , \tag{45}$$

$$f_j = \sum_{r=1}^{R} f_j^r \quad , \qquad (j \in A), \tag{46}$$

$$\sum_{j \in A} G_j(z_j) \leq B \quad , \tag{47}$$

$$f_j^r \geq 0 \quad , \qquad (j \in A;\ 1, \ldots, R) \quad , \tag{48}$$

$$L_j \leq z_j \leq P_j \quad , \qquad (j \in A) \quad . \tag{49}$$

The notation here is the same as for the previous network design problem (15) to (19) with the addition of the budget constraint (47).

2.3.4 Solution Algorithm -- With Budget Constraint

The solution technique for the problem (44) to (49) uses the method described in Section 2.3.2; instead of using a fixed multiplier λ, however, the new algorithm adjusts this value (which is actually the Lagrange multiplier for the constraint (47)) until an investment close to the budget B is found.

Consider what happens in the problem (15) to (19) as λ changes. When λ is small, investment dollars have little effect on the objective function; as λ increases, however, marginal investments become less attractive, and the travel time component of Z may be allowed to increase somewhat to cause a decrease in investment. Thus, total investment is a monotone decreasing function of the Lagrange multiplier λ, although possibly a discontinuous one as illustrated in Figure 2.4.

The network design problem with a budget constraint may thus be solved by dealing with a series of problems without budget constraints, each with a different value of λ. As each problem is solved, the total investment $\sum G_j(z_j)$ is compared with the budget B; if investment is over the budget, the value of λ is increased for the succeeding problem, while the multiplier is decreased when investment is below the budget. Eventually, the situation depicted in Figure 2.4 results if the budget constraint is binding; two values of λ are found for which the optimal total investments straddle the required budget. Further adjustment of λ may be carried out by Bolzano search (interval bisection) or by a linear interpolation technique; the latter is the approach implemented in CATNAP.

The algorithm is terminated whenever an investment sufficiently close to B is found, or else whenever the difference $|\lambda_2 - \lambda_1|$ is sufficiently small.

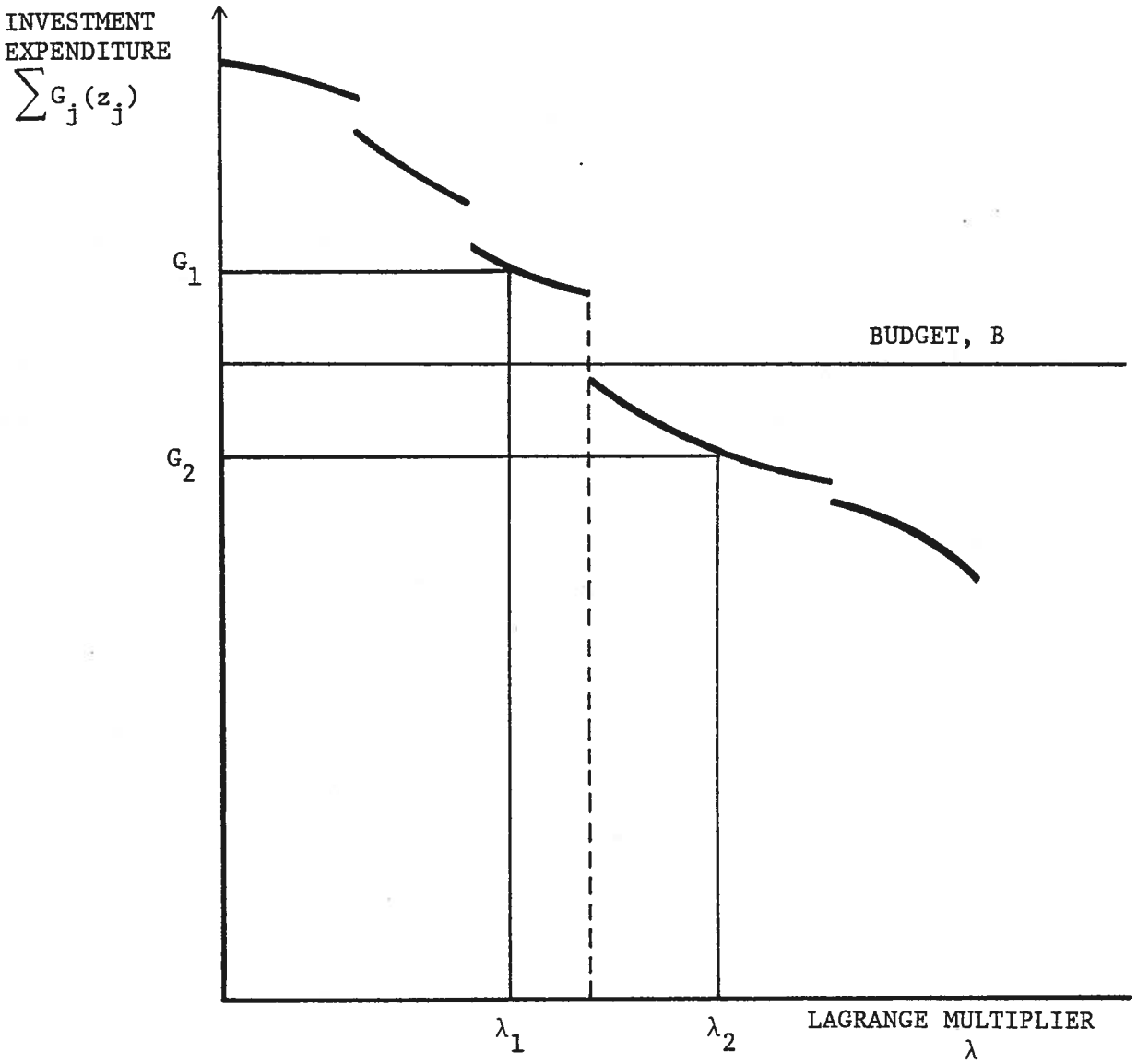


Figure 2.4 - Variation of Total Investment with Lagrange Multiplier

The final vectors of optimal investment decisions $\{Z2_j\}$ and $\{Z1_j\}$ may then be linearly combined, so that the budget constraint (47) is exactly satisfied; a final traffic assignment optimization for this budget schedule completes the solution of the network design problem.

2.3.5 User Equilibrium Assignment

As mentioned previously, the network design problem solution algorithm requires that traffic be assigned to the network in accordance with a system optimal objective function. This is necessary so that the problem may be decomposed into the link subproblems.

The system optimal objective is satisfactory for some types of networks (e.g., rail lines), but it is inappropriate for others (e.g., highway networks), where the behavior of individual users would seem to call for a user equilibrium traffic assignment. The network design objective in this case remains the minimization of total user travel time, possibly subject to a budget constraint, but the traffic assignment objective in this case is given by Eq. (6), while the objective function is evaluated while using a different relation (5). The decomposition method cannot be applied to this problem.

A fairly simple means was devised in [1] to deal with this situation. If we denote by Z_u^* the unknown actual optimum value of the objective function (total travel time) in the network design problem with user equilibrium assignment, then it is possible to obtain some bounds on Z_u^* based on the solution obtained by a system optimal traffic assignment. It we let Z_s^* be the total travel time for the final system configuration in the system optimal case, and if we perform a user equilibrium assignment of traffic in this configuration letting the resulting time be Z_u^s, then we have

$$Z_s^* \leq Z_u^* \leq Z_u^s \quad . \tag{50}$$

If the values Z_s^* and Z_u^s are fairly close, we may then assert that the optimal investment pattern for the user equilibrium problem (whatever it may be) results in a total travel time not much better than the investment pattern obtained from the system optimal problem. CATNAP has the capability of finding both Z_s^* and Z_u^s in the same run in which the final configuration is determined.

2.3.6 Network Design with Discrete Investments

A limitation of the network design solution methods of Sections 2.3.2 and 2.3.4 is that they require the investment decision variable z_j to be continuous even though in some cases they should be considered as discrete. For example, it makes little sense to add one-half or two-thirds of a lane to a highway link; note, however, that it may be perfectly reasonable to repave the first two miles of a five mile link and thereby obtain a partial improvement in the free-flow travel time.

The discrete investment network design problem is a very large integer-nonlinear program which is certainly beyond the capability of present-day computing resources even for moderate size networks. CATNAP thus approaches the task heuristically using a cost-benefit criterion to find a local discrete investment solution from the optimal continuous investments obtained in solving a network design problem as described above.

Let us assume that the continuous investment optimum solution has flow f_j^* and investment decision z_j^* on link j. Assume further that the two discrete investment decision options for each link j which are closest to z_j^* are z_j^a and z_j^b with $z_j^a \leq z_j^* \leq z_j^b$; note that z_j^a may well be zero. Now if z_j^* is very close to one or the other of these options (say, $|z_j^* - z_j^b| < \epsilon$) ,

then that option is chosen as the discrete investment decision; otherwise the cost benefit ratio

$$\frac{G_j(z_j^b) - G_j(z_j^a)}{D_j(z_j^a, f_j^*) - D_j(z_j^b, f_j^*)} , \tag{51}$$

is computed.

All arcs are then ordered by cost benefit ratio, smallest first. After allowing for investments made on arcs for which ratios were not computed, the lower bound sum $\sum G_j(z_j^a)$ is found and subtracted from the budget B. The first arc on the list is then tested: if $G_j(z_j^b) - G_j(z_j^a)$ is less than the remaining budget, the higher investment decision z_j^b is adopted and the budget adjusted; otherwise, the lower decision z_j^a is made. This procedure is repeated for all other arcs on the list until a set of discrete investment decisions satisfying the budget constraint is found.

2.4 INVESTMENT STAGING PROBLEM

The investment staging problem consists of making improvements to a transportation network over a multi-period time horizon. Each period is assumed to have its own budget limit and its own set of trip demand requirements; these requirements typically change in such a way during the planning horizon that it becomes more efficient to add some improvements earlier than others. There are two versions of this problem: one in which the ultimate configuration of the network is specified in advance, and the other in which the final configuration is to be determined as part of the problem.

Approaches to the investment staging problem which attempt to deal with all stages at once result in problems which are much too large; accordingly,

our solution method involves decomposing the problem so that it can be solved one stage at a time. The stages which are considered to be most important by the user are solved first; the results of the first solved stages are used in the solution of the other stages to insure consistency in the overall solution.

2.4.1 Problem Formulation

The investment staging problem consists of a set of network design problems indexed over a set of time periods, $t = 1, \ldots, T$. The formulation for each period is the same as in Section 2.3.3 except that the additional subscript t is used to identify the period.

$$U_t = \sum_{j \in A} D_{jt}(f_{jt}, z_{jt}) \quad , \tag{52}$$

$$\sum_{j \in W_i} f_{jt}^r - \sum_{j \in V_i} f_{jt}^r = h_{it}^r \quad , \quad (i \in N;\ r = 1, \ldots, R) \quad , \tag{53}$$

$$f_{jt} = \sum_{r=1}^{R} f_{jt}^r \quad (j \in A) \quad , \tag{54}$$

$$\sum_{j \in A} G_j(z_{jt}) \leq B_t \quad , \tag{55}$$

$$f_{jt}^r \geq 0 \quad , \quad (j \in A;\ r = 1, \ldots, R) \quad , \tag{56}$$

$$z_{j,t-1} \leq z_{jt} \leq z_{j,t+1}, \qquad (t = 2, 3, \ldots, T-1) \qquad (57)$$

$$z_{j1} \geq L_j, \qquad (j \in A), \qquad (58)$$

$$z_{jT} \leq P_j, \qquad (j \in A). \qquad (59)$$

The notation here is basically the same as that employed earlier except for the subscripts t as mentioned above.

The investment decision variable z_{jt} is the total capacity added to link j during the first t stages, so that constraint (57) expresses the linkage between time periods, namely, that improvements cannot be removed once they are installed; further, no improvement may be undertaken in the current period unless it is also undertaken in the succeeding one.

Note that in (53) the supply-and-demand quantities h^r_{it} for each node may change over the time horizon, and that the budget B_t in (55) also varies in time (it is assumed that $B_t \geq B_{t-1}$, i.e., the budgets are cumulative).

The objective function (52) for this subproblem gives the total travel time in a single period; the investment staging master problem then seeks to minimize all the U_t values, $t = 1, \ldots, T$. As mentioned previously, attempting to deal with all U_t's simultaneously results in problems which are generally intractably large; consequently, we choose an ordering of stages, from most important to least, so that we may deal with one stage at a time. A useful ordering is T, 1, 2, ..., T-1; for this choice, the master problem objective function is

$$\text{LEXICO - MINIMIZE} \quad [U_T, U_1, \ldots, U_{T-1}]. \qquad (60)$$

This notation means that we first solve the problem (52) to (59) for stage T, minimizing U_T by an appropriate choice of investment decisions Z_{jT}. We then solve the stage 1 problem, using the investment values found for stage T as upper bounds in (57).

We may specify any order for solving the subproblems. Let the first stage to be solved be $\alpha(1)$, the second $\alpha(2)$, and so forth; the objective function is then

$$\text{LEXICO - MINIMIZE } [U_{\alpha(1)}, U_{\alpha(2)}, \ldots, U_{\alpha(T)}]. \tag{61}$$

The bounds (54) in each subproblem are selected not necessarily from the immediately succeeding and following stages (which may not have been solved), but rather according to

$$\text{lower bound: } \underset{\tau}{\text{MAX}}[z_{j,\alpha(\tau)}: \ \tau < t, \ \alpha(\tau) < \alpha(t)] \ , \tag{62}$$

$$\text{upper bound: } \underset{\tau}{\text{MIN}}[z_{j,\alpha(\tau)}: \ \tau < t, \ \alpha(\tau) > \alpha(t)] \ . \tag{63}$$

The above discussion has centered on the investment staging problem for which the final configuration must be determined, i.e., $z_{jT} \leq P_j$. When the final configuration is fixed we exchange this constraint in constraint set (59) for

$$Z_{jT} = P_j, \qquad (j \varepsilon A), \tag{64}$$

where P_j is the specified final investment for link j. The stage T subproblem is solved as a traffic assignment problem if the flow pattern is required.

2.4.2 Solution Algorithm

Given the discussion in the previous section, the proposed solution method for the investment staging problem becomes quite clear: a series of T network design problems are solved by the Lagrange multiplier technique of Section 2.3.4. (Only T - 1 problems need be solved when the stage T configuration is fixed in advance.) The set-up phase for each problem requires the addition of the investment decision bounds according to (62) and (63) if we call these bounds L_{jt} and P_{jt}, it is clear that the resulting constraint is identical in form to (49), so that nothing new has been added to the problem itself.

All the features described in Section 2.3 for the network design problem with budget constraint are applicable to the single period subproblems of the investment staging problem. In particular, the discretization heuristic of Section 2.3.6 may be applied to any or all of the stages; in this case, the bounds (62) and (63) will use the discrete rather than continuous investments. Also, any of the three improvement options of Section 2.3.1 may be used.

3. COMPUTER PROGRAM

3.1 INTRODUCTION AND OVERVIEW

Control Analysis Transportation Network Analysis Program (CATNAP) is a high speed, in-core network optimization code designed to solve all the problems described in Section 2 of this report. It consists of about twenty program modules which are divided into three main groups: data input, traffic assignment, and network design (see Table 3.1). In this section, we present a summary of the main features and interrelationships of these groups; more complete descriptions are provided in Sections 3.2 to 3.4. Detailed instructions for the program user (data formats, job control language, etc.) are contained in the appendices. See Figure 3.1 for a general flow diagram.

Because CATNAP is intended to solve large problems, the input modules have been designed to accept data from a series of different files on tape or disk rather than sequentially from cards; this allows the user maximum flexibility. Raw input data are read from these files and are processed by the input modules into a compact internal representation; the internal form may then be saved in a new file and used to save processing time when starting a subsequent run.

Separate input files are used for the network topography data, for the trip table, for investment data, and for investment bounds and staging data. If piecewise linear approximations are generated, these may also be saved and used as input on a later run. The current version of CATNAP uses the same input format as the Federal Highway Administration (FHWA) UROAD code, thus allowing the new program to be used without modifying existing data sets.

TABLE 3.1: SUMMARY OF MODULES IN CATNAP

MODULE NAME	FUNCTION
INPUT GROUP	
INPUT	Exercises overall control
ECHO	Prints summary of basic parameters
RDLD	Reads network topography
RDTMX	Reads trip matrix
RDINV	Reads investment data
PWLA	Sets up piecewise linear approximations
RDBND	Reads investment bounds
LPGEN	Generates LP formulation
RESTRT	Restarts problem from get-off dump
TRAFFIC ASSIGNMENT GROUP	
TASSGN	Exercises overall control
SOLVE	Finds total travel time and slopes
PWEVAL/PWVALS	Evaluates piecewise linear approximations
FLOWS	Finds feasible flow patterns
PATHS	Solves shortest path problem
SRCH	Finds optimum flow by golden section search
TCOST	Finds travel times without slopes
GETOFF	Produces get-off dump for later restart
SOTOUE/UETOSO	Converts between system optimal and user equilibrium objectives
NETWORK DESIGN GROUP	
LKSB	Sets up solutions to link subproblems
LSRCH	Finds new Lagrange multiplier by regula falsi search
ADJUST	Finds and evaluates final solution
DATA MODULES (COMMON BLOCKS)	
PARMS	Contains basic parameters
PROB	Contains investment and flow values
LINK	Contains network topography data
TRIPS	Contains trip demand data
INVST	Contains investment data
APPX	Contains piecewise linear approximation data
WORK	Contains work arrays

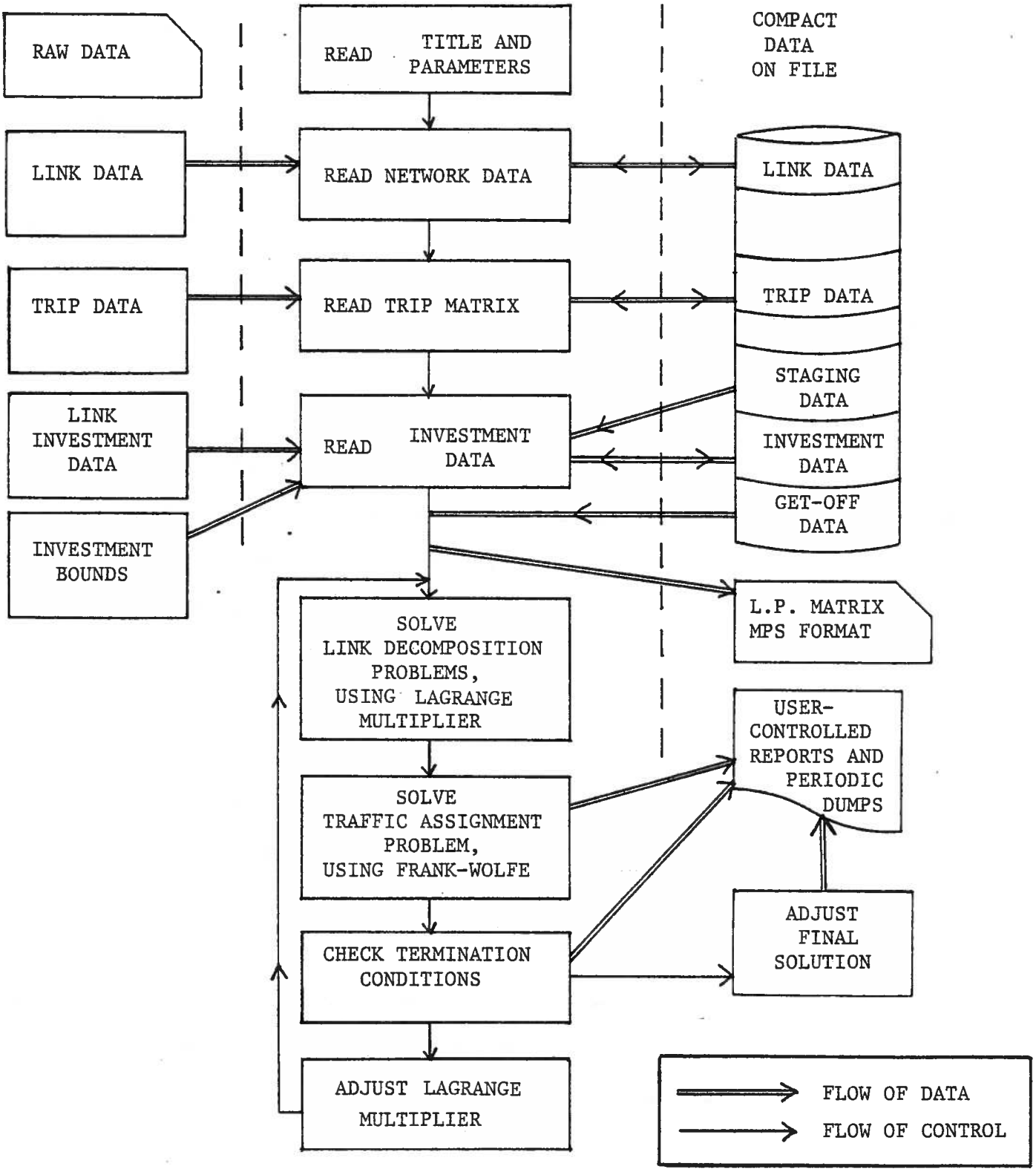


Figure 3.1 - Flow Diagram of CATNAP

The input routines also find an initial feasible flow pattern; this may be one determined in a previous run, or it may be computed by assigning all trip demands to the shortest time path without regard to congestion. This determination of a feasible flow pattern is at the heart of the CATNAP code; it is done by a subroutine called FLOWS. This routine repeatedly invokes the shortest path routine PATHS which is an implementation of the Dijkstra algorithm using a heap structure [5]. FLOWS then accumulate all trip demands over the arcs on the shortest paths.

Although they may be initially invoked by the input modules, FLOWS and PATHS are actually part of the traffic assignment module group. This group implements the Frank-Wolfe algorithm by generating a sequence of trial feasible flow patterns; each of these in turn results from calling FLOWS with the marginal cost of flow on each arc as the distance measure for PATHS. Separate routines are provided to determine these marginal costs, and also to find the optimum combination of the old and new flow patterns by golden section search.

A variety of user options are available for directing the operation of these modules; the various iteration counters, output flags and tolerance values are summarized in Table 3.2. One notable feature of the traffic assignment group is the built-in get-off feature which allows the problem state to be saved on a disk file for a later restart. This feature remains in effect even when the traffic assignment code is embedded in a network design run.

The network design modules will typically be invoked from a main program; they give the use the capabilities of setting up all the link subproblems (i.e., finding the optimum investment on each arc), of directing the Lagrange

TABLE 3.2: BASIC SYSTEM PARAMETERS IN CATNAP WITH THEIR MEANINGS

NAME	TYPE[1]	USE
MAJOR PARAMETERS		
NS	I	Stage number for staging problem (= 0 for no staging)
BUD	R	Budget for network design problem (= 0 for traffic assignment only)
ALAM	R	Initial value for Lagrange Multiplier
UE	L	If TRUE, objective is user equilibrium (traffic assignment only)
ITERATION COUNTERS		
K1MAX	I	Maximum Lagrange multiplier iterations
K2MAX	I	Maximum traffic assignment iterations (per Lagrange multiplier)
PROBLEM DATA FLAGS		
IOLD	I	Controls processing of problem Data (Link Data, Trip Matrix, INVestment data, investment BouNDs, PieceWise Linear curves) as follows: 0 Input raw data 1 Input previously saved internal data 2 Raw input, save internal form
IOTMX	I	
IOINV	I	
IOBND	I	
IOPWL	I	
INITIALIZATION FLAGS		
ISTART	I	If nonzero, start problem from old get-off file (unit 51)
IFLOW	I	If zero, start initial flows from scratch, other wise read them from unit IFLOW

[1]The "Type" column uses the following code: I, for integer; L, for logical; and R, for real.

TABLE 3.2 (continued...)

NAME	TYPE	USE
OUTPUT COUNTERS (no output if value if zero)		
KOUT1	I	Print detailed flow summary every KOUT Lagrange multiplier iterations
KOUT2	I	Print detailed flow summary every KOUT Frank-Wolfe iterations
IDUMP	I	Save a get-off dump every IDUMP Frank-Wolfe iterations
PRINT FLAGS (no output if value is FALSE)		
PRINTP	L	Print basic piecewise linear approximations
PRINTL	L	Print optimal investment and cost piecewise linear curves for each multiplier
PRINTS	L	Print detailed golden section search results
PRINTD	L	Print debug output
APPROXIMATION FLAGS		
PWT	L	Use piecewise linear approximations on all arcs if TRUE
PWI	L	Use piecewise linear approximations on investment arcs if TRUE; only on time improvement arcs if FALSE
LP	L	Dump an LP formulation of the problem if TRUE

TABLE 3.2: (continued...)

NAME	TYPE	USE
TOLERANCES		
FWTOLI	R	Relative tolerance on objective function in golden section search portion of the Frank-Wolfe algorithm
PHITOL	R	Absolute tolerance on the weight parameter in golden section search
FWTOLO	R	Relative tolerance on objective function in the Frank-Wolfe algorithm
BTOL	R	Absolute tolerance on the budget (network design)
TLAM	R	Absolute tolerance on the Lagrange multiplier (network design)
DLAM	R	Initial step size for adjusting the Lagrange multiplier

multiplier search and of adjusting the final solution. This final adjustment may merely change the investments slightly, so that the budget constraint is exactly met, or it may round the continuous investment variables to integer values using the heuristic described in Section 2.6. In either case, a last traffic assignment run is carried out for the post-investment configuration of the network.

The investment staging method described in Section 2.4 consists of a series of network design runs, one for each stage. The linkage between stages is expressed by the requirement that $z_{jt} \leq z_{js}$ for $t < s$; the staging algorithm has thus been implemented by using CATNAP's capability to set bounds on investments. A data file contains a summary of the optimal investments for all stages previously solved; these values are used as upper and lower bounds for the current stage being solved. (For stages preceding the current stage in time, the investments are lower bounds; for stages later in time, upper bounds.) Once the final adjustment has been completed for the current stage, its investment decisions are added to the file to be used for subsequent runs.

More complete descriptions of the algorithms used in each CATNAP's modules are given in Sections 3.2 to 3.4; the aim is to present the main features of the techniques leaving the detailed logical specification to the comments accompanying the computer code itself. An overall system flow diagram is included as Figure 3.1. Section 3.5 describes the capabilities and requirements of CATNAP, while Section 3.6 discusses potential extensions and modifications.

3.2 DATA INPUT MODULES

The data input modules have as their main purpose the complete initialization of the problem. Secondary purposes include the generation of a linear programming formulation of the problem in MPS format and the saving of compact internal data representations for later use.

Overall control of the input function is exercised by subroutine INPUT. This module first reads a set of control cards and then invokes the other input and initialization routines as required. There are five control cards containing 29 basic parameter values; the names and meanings of these parameters are indicated in Table 3.2, while detailed formats are given in Appendix B.

Subroutine ECHO prints a summary of these input parameters once they have been checked for consistency.

As has been mentioned, CATNAP is designed for data input from a series of files, each of which is given a unique number. The conclusion of data of a particular type is thus signaled by an end-of-file condition rather than by a sentinel or dummy value; this makes it considerably easier to handle large problems. The disadvantage is that a somewhat complex organization of files is now needed; this is set forth in Table 3.3 with some sample job control language given in Appendix C.

The first data input file contains the description of the network topography; the current version of CATNAP accepts input in the format used in the UROAD code. There is one card for each link giving the initial and terminal node numbers for the link as well as information from which the zero-flow travel time and practical capacity may be found. This file is processed by subroutine RDLD (for ReaD Link Data).

TABLE 3.3: DATA SETS USED IN CATNAP[1]

NUMBER	SYMBOL	DESCRIPTION	USE
5	ICONT	Read control cards	IN
6	IOUT	Printed output	OUT
6	IERRU	Error messages	OUT
41	LINKD	Raw link data	IN
42	NEWLD	Raw data for new links	IN
43	ITRIPD	Raw trip matrix	IN
44	INVD	Raw investment data	IN
45	IBND	Raw investment bounds	IN
46	LINKS	Saved link data	IN/OUT
47	ITRIPS	Saved trip matrix	IN/OUT
48	INVS	Saved investment data	IN/OUT
49	IBNDS	Saved investment bounds	IN/OUT
50	IPWLS	Saved approximations	IN/OUT
51	IRST	Restart unit	IN
52	IGTOF1	Get-off unit 1	OUT
53	IGTOF2	Get-off unit 2	OUT
54	LPOUT,NF	LP formulation output	OUT
55	IFLOW	Saved flow pattern	IN
56	ISOL	Saved flows and parameters	OUT
60	IBDIN	Staged investments	IN
61	IBDOUT	Staged investments	OUT

[1]Data sets are referred to by symbol name in the code; the actual data set numbers are initialized in COMMON block PARMS in a BLOCK DATA subroutine.

Once data for all of the arcs have been read, a compact representation of the network is created in COMMON block LINK. The arcs are sorted by their origin node index in increasing order; the process is simplified by keeping a linked list structure during data input. Using this scheme, a specific arc has an index which depends on the network topology, and which may well change if new arcs are added. For this reason, data describing new arcs to be constructed in a network design problem must be made available to the link data subroutine. Also, all references to a particular arc in either input data or in printed output give the initial and terminal node numbers of the arc rather than relying on a specific arc index.

Sorting the arcs by origin node allows an efficient representation of the network. An array called INDEX gives the arc indices of the first arc leaving each node; thus, the arcs leaving node N are INDEX(N), INDEX(N) + 1, ..., INDEX(N + 1) - 1. The destination node of arc J is stored in the array IJ, while the arrays CA and T store the practical capacity and free-flow travel times, respectively. For a problem with N nodes and A arcs, then, the total storage requirement is just $2\frac{1}{2}$ A + $\frac{1}{2}$ N words (since both INDEX and IJ may be half-word integer arrays). This organization is used in the UROAD system and also in Dial [8].

The second input file contains the trip-demand table; this file is processed by subroutine RDTMX (for ReaD Trip MatriX). There is likely to be a great deal of data in this file since the UROAD input format specifies a single card for each origin-destination pair. CATNAP has the convention that O-D nodes have the lowest index numbers; thus, the trip table may be readily stored in a Z x Z matrix, where Z is the maximum number of O-D's (also called zones). Because of limitations in the input data as well as the need to conserve storage, the trip matrix is stored as half-word integers.

Since shortest path problems must be solved for each origin node in the network, a determination of which zones are never origins but only destinations is made by RDTMX. This information is stored in the one-byte (quarter word) logical array NOTORG; a shortest path problem is needed for node I only if NOTORG(I) has the value FALSE. This array brings the total memory requirement to $\frac{1}{2}Z^2 + \frac{1}{4}Z$ words, where Z is the number of zones.

It should be noted that most of the computer memory in really large problems (say, over 1000 nodes) is needed for the trip matrix. Since the only function of this data is to load the correct flows onto the network in the generation of a new feasible flow pattern, only the information for a single origin needs to be in core at any given time. Thus, modification of RDTMX to set up a disk file containing demand data by origin is needed to manage very large transportation networks; such modifications would not be very difficult.

The third input file contains the investment data; there will be no such data, of course, for traffic assignment problems. A single card is needed for each discrete investment possibility on each link. This card contains the arc identification (i.e., initial and terminal node numbers), the cost of the investment and the effect of the investment (i.e., the post-investment link capacity and free-flow travel time). Up to nine discrete investments may be specified for each arc; each investment must monotonically improve the arc parameters over all investments which involve a lower amount.

It is likely that the majority of arcs in practical network design problems will not be candidates for investment. To conserve storage, therefore, each investment is given an index number when its card is read and an auxiliary array (INV) is used to indicate which investment number is applicable for a specific link; $INV(J) = 0$ implies that no investment is possible on link J. This results in a total computer memory requirement of $6\frac{1}{2}C + \frac{1}{2}A$ words, where C is the total number of investments and A the number of arcs in the problem.

The cost, final free-flow travel time and final capacity are stored for each investment. In addition, investments for a given arc are linked together into a list; the INV array entry points to the most expensive possibility, while a link field for each investment indicates the next cheapest option. Also needed for each investment are storage for a constant as well as for the investment bounds.

The exact interpretation of the stored values for a particular investment depends on the type of improvement involved. If the most expensive investment has the same free-flow travel time as the unimproved link, the other discrete investment possibilities on the arc are normally ignored until the final adjustment phase of the problem and the continuous capacity improvement method (option 1) described in Section 2.3.2 is used. (Note that use of the piecewise linear formulation may be forced by setting the input parameter PWI to TRUE; in this case, the values have the same meaning as for time improvements below.) The stored values corresponding to this capacity improvement are interpreted as follows:

INV(J)	Investment index, arc J ; denoted by NC below.
TNEW(NC)	New free-flow travel time; equals T(J) in this case.
CANEW(NC)	New capacity; must be greater than CA(J).
ZLOW(NC)	Initial lower bound on improvement; set to zero.
ZHIGH(NC)	Initial upper bound on improvement; set to CANEW(NC) - CA(J).
COST(NC)	Equals input cost divided by ZHIGH(NC); i.e., unit cost for capacity increase, g_j.
CONS(NC)	Used to find $\phi_j(\lambda)$; equal to $\left[\frac{g_j}{kr\, t_j}\right]^{1/(k+1)}$.
LIST(NC)	Points to the next cheapest investment (zero if none exists).

Other discrete investment options in the capacity improvement case above are stored with the CONS value set to zero and with the COST value as input. These options are used only to round off the final continuous solution if the technique described in Section 2.3.6 is used.

When any of the investment possibilities for a link involve a decrease in the free-flow travel time, a piecewise linear formulation is required as described in Section 2.3.1 (option 3); this will also be true for every investment whenever either of the input flags PWI or PWT are set to TRUE (option 2). In this case, the stored investment values are:

INV(J)	Investment index, arc J ; denoted by NC below.
TNEW(NC)	New free-flow travel time; will be less than T(J).
CANEW(NC)	New practical capacity; may be greater than or equal to CA(J).
ZLOW(NC)	Set to zero.
ZHIGH(NC)	Set to CANEW(NC).
COST(NC)	The input total cost for the improvement.
CONS(NC)	Not used; set to zero .
LIST(NC)	Points to the next cheapest investment .

Once the investment possibilities have been set and printed out by subroutine RDINV (for ReaD INVestment data), the PieceWise Linear Approximations are generated for each arc by subroutine PWLA. These approximations are for both $T_j(\cdot)$ and $G_j(\cdot)$ curves; a multiplier array to indicate the amount by which a given improvement changes the length of the linear segments is also required.

The travel-time curves are at present generated from the basic FHWA non-linear curve (see (7), Section 2.2.1). The length of each segment is a multiple of the input practical capacity, while the slopes depend only on the free-flow travel times. Using the FHWA congestion function with r = 0.15, k = 4, and the intervals now set in CATNAP results in:

Segment	Length	Slope
1	1.0000·CA(J)	1.1500·T(J)
2	0.3582·CA(J)	2.5168·T(J)
3	1.1418·CA(J)	13.2223·T(J)
4	1.0000·CA(J)	65.1344·T(J)
Final	Unbounded	100. T(J) + 200

The user may select any number of segments from two to five; note that the last segment always has the slope listed for "Final."

Whenever investment is possible on a particular link, the multiplier value F_j^m must be found for each segment in the travel-time curve according to Eqs. (31) and (32) in Section 2.3.1. The interleaving of reduced slope segments described in Section 2.3.1 must also take place whenever link investment involves a lower zero-flow travel time. Subroutine PWLA has provisions for both these features.

Piecewise linear investment functions are also determined by PWLA. On arcs where only capacity improvements occur, investment functions are computed only when PWI is set to TRUE. In this case, the function gives the cost in monetary units of a specific increase in link capacity; the breakpoints in the function are determined by the input discrete investments for the arc. Since the investment function used for network design must be convex, however, the adjustment depicted in Figure 3.2 will take place whenever the discrete investments lie on a non-convex curve.

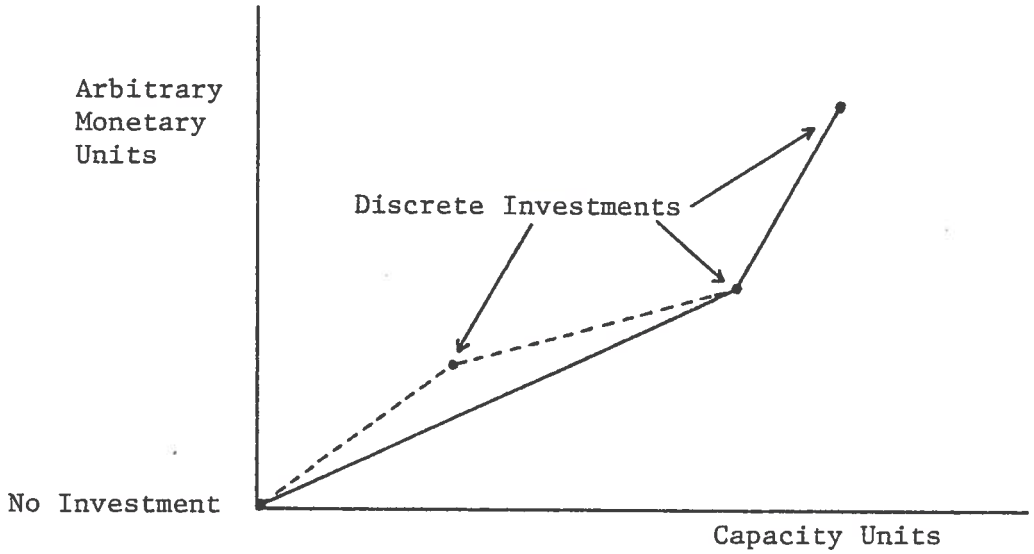
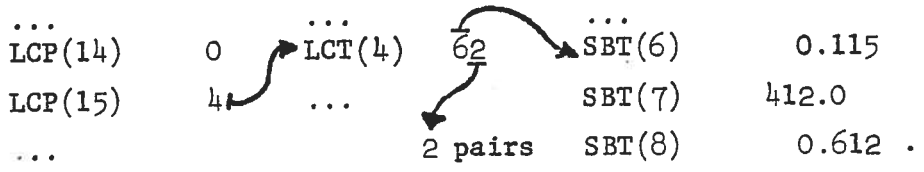


Figure 3.2: Generation of Piecewise Linear Investment Curve (Solid Line) from Input Discrete Investments

A similar adjustment is called for on arcs where improvements involve free-flow travel time. In this case, however, the abscissa of the investment curve ranges over the total final capacity of the link rather than over the change in capacity. This is because the travel-time curve segments corresponding to the new free-flow time are required to expand from a zero initial value to the final link capacity.

All of the piecewise linear approximations are stored in the common block APPX according to the following scheme. Since most links in really large problems will not have piecewise linear approximations, a single pointer array LCP is used just as the INV array is used for investments, i.e., LCP(J) indicates which approximation is to be used for arc J with LCP(J) = 0 implying that

none exists. A second pointer array is used for each piecewise linear function; the units digit of this pointer value gives the number of segments in the approximation, while the rest of the pointer indicates the location of the first function slope in a slope-breakpoint array. The following example should help to clarify this:



No piecewise linear approximations have been generated for arc 14 in this case since LCP(14) = 0; arc 15, however, uses approximation 4. The time curve for arc 15 consists of two linear segments since the units difit of LCT(4) is 2; the first slope for the time curve is stored in SBT(6) (LCT(4)/10 = 6), and the first segment has a width of 412 units (SBT(7)).

Locator arrays and slope-breakpoint arrays are provided in APPX for travel-time curves $T_j(\cdot)$ (LCT and SBT), investment curves $G_j(\cdot)$ (LCI and SBI), optimal social transportation cost curves $H_j(\cdot)$ (LCC and SBC) and optimal investment curves $I_j(\cdot)$ (LCIO and SBIO). The latter two curves are initialized by LKSB for each Lagrange multiplier value; see Section 3.4. A separate locator array is needed for the capacity expansion fractions F_j^m (LCF and F); in this case, the number of points need not be stored as it is just one less than for the corresponding travel time curve.

It is somewhat difficult to determine the total amount of storage needed for the piecewise linear approximations; it is clear though that these requirements

will seriously limit the size of the problem which CATNAP can solve unless most of the arcs use nonlinear curves. If we thus assume that piecewise linear approximations are used only for time-improvement investments and that four segments are used in the time curve, an estimate of the storage requirement is $\frac{1}{2}$ A + 44$\frac{1}{2}$ P, where P is the maximum number of investments.

The final input module is called RDBND (for ReaD investment BounDs); its function is to set upper and lower limits on the amount of money spent on a given improvement. The bounds are tested when an investment decision is made for a link during a network design problem, and the decision is forced to conform to the bounds. It is far more convenient, however, for the network design code to impose the bounds directly on the improvements rather than on monetary amounts. The input data, which are in monetary units for user convenience, are thus converted by RDBND to bounds on the improvements which result from the specified investments; recall that these improvements are in units of capacity.

RDBND also reads in the results from all previously solved stages in an investment-staging problem; these results are saved in compact form in a file on an external storage device. The first record on the file consists of a vector indicating which stages have already been solved; this vector is updated for the current stage, and then written out on a new file. Solutions from the other stages are read in one at a time and used for bounds as described in Section 3.1; those solutions required for subsequent stages are then also written out on the new file. Note that it may be unnecessary to save all of the input solutions; for example, the stage 1 results need not be saved when the problem currently under consideration is for stage 2 since the current results will provide uniformly better bounds for subsequent stages. The solution to the current stage is added to the output file (unit 61) by subroutine ADJUST; see Section 3.4.

Once all the input data have been read, subroutine INPUT modifies the capacity data when a user equilibrium traffic assignment is called for. Using the FHWA curve (8) instead of (7) requires that all system capacities be multiplied by the factor $(k + 1)^{1/k}$ (see Section 2.2.1); this operation is carried out by subroutine SOTOUE (for System Optimal TO User Equilibrium). The piecewise linear curves and investment data are also changed by SOTOUE. The final user equilibrium solution, it will be recalled, must be evaluated using the actual $T_j(\cdot)$ curve once the proper flow pattern has been found; the subroutine UETOSO (User Equilibrium TO System Optimal) is thus provided to change the capacity parameters and related data back to the original values as input.

When a problem formulation as a linear program is called for (input parameter LP; see Table 3.2), subroutine LPGEN is next invoked by INPUT. For this to take place, of course, piecewise linear functions must be used on all arcs in the problem (controlled by input parameter PWT). The output from LPGEN is in MPS input format which is readily accepted by many commercial linear programming codes.

The final operation performed by INPUT is the generation of an initial feasible solution to the traffic assignment problem. This may be done by invoking subroutine FLOWS as described in the next section, or a vector of feasible flows found in an earlier run may be read in from an external source. Alternatively, INPUT may invoke the restart routine to restore a previously saved partial problem solution.

3.3 TRAFFIC ASSIGNMENT MODULES

The modules in this group implement the Frank-Wolfe traffic assignment algorithm described in Section 2.2. Given the data structures and parameters discussed in Section 3.2, the operation of these modules is quite straightforward. This section thus describes only the salient features of the algorithms used, leaving details of the logic to the comments in the computer code.

Overall control of the traffic assignment algorithm is exercised by subroutine TASSGN. This module uses an iteration counter (K2) to determine the number of Frank-Wolfe steps to be carried out (up to K2MAX; see Table 3.2). Because TASSGN is used with CATNAP's restart capability, K2 must be set before the routine is called the first time. The K2 counter is also used for controlling the amount of information printed out at each iteration.

The current flow pattern in the network is first evaluated by subroutine SOLVE; recall that the input modules provide such a pattern when the problem is initialized. This routine computes not only the total travel time on each arc but also the slope of the travel time curve (using, where appropriate, the "corner-rounding" technique of Section 2.2.2). For network-design problems, SOLVE also determines the optimum investment amount and the resulting post-investment travel time; the slope which is found in this case is for the total social transportation cost function, i.e., $H_j'(f_j)$. The final function of SOLVE is to print an iteration summary; this is controlled by the single input parameter to the subroutine, N. If N = 1, SOLVE prints the flow on each arc, while if N = 0, only a single summary line is printed.

Once all the marginal flow costs (slopes) have been found by SOLVE, TASSGN next invokes FLOWS to find a new feasible flow pattern. This is done by solving a shortest path problem for each origin node in the network using the marginal costs as a distance measure and then assigning all the demands originating at that origin to the shortest path. The shortest path subroutine is called PATHS: it uses the Dikjstra algorithm with a binary heap which results in excellent performance for the typically sparse networks encountered in traffic assignment applications. See reference [5] for a more complete description. The assignment of traffic flows (called "arc loading") is performed by FLOWS itself; the routine must have access to the trip matrix for this reason. If it were necessary to store a very large trip matrix out of core as discussed in Section 3.2 some straightforward changes to FLOWS would be required.

The old and new flow patterns (which are stored in the arrays F1 and F2, respectively) must now be linearly combined to produce an optimum pattern; i.e., we find the value of ϕ such that the flow pattern $(1 - \phi)F1 + \phi F2$ has the minimum total cost. Since this cost is a convex function of ϕ, golden section search is an appropriate method here and has been implemented in subroutine SRCH. See reference [4] for details of the algorithm and the derivation of a lower bound on the travel-time function. Two termination tests are used in SRCH: an absolute test on the change in value of ϕ (input parameter PHITOL) and a relative test on the lower bound on the total travel time (input parameter FWTOLI). SRCH will also terminate if the cost as a function of ϕ is found to be non-convex; this is normally caused by computer round-off

error when it occurs. Because the computation of marginal costs by SOLVE involves considerable overhead, a routine called TCOST which computes only total costs is used by SRCH to evaluate total time for the various values of ϕ.

It has been found that CATNAP spends most of its time in shortest path and arc-loading calculations and that golden section search is relatively inexpensive. It seems best, therefore, to set the termination tolerances for SRCH tight (e.g., PHITOL = 0.0001, FWTOLI = 0.00001); this makes the best use possible of the expensive trial solution from FLOWS. However, the slow convergence of the Frank-Wolfe Algorithm may possibly be speeded up if SRCH chooses a suboptimal ϕ; this may be a fruitful area for future work.

Once a final value for ϕ has been returned to TASSGN, the flow vector F1 is updated and a new lower bound on the objective function is found according to Eq. (10) in Section 2.2.2. The traffic assignment algorithm is then terminated if this lower bound is within a specified relative tolerance of the objective function value (see parameter FWTOLO in Table 3.2). Termination will also occur if a zero value of ϕ is returned by SRCH, or if the iteration counter K2 has reached K2MAX.

The problem state (all flows, investments and linear approximations) is automatically saved by a call to subroutine GETOFF every IDUMP Frank-Wolfe iterations (IDUMP is an input parameter; see Table 3.2). Note that in a network design problem the get-off dump counter is independent of the Frank-Wolfe counter K2. These get-off dumps are taken alternately on units 52 and 53; this is to prevent information being lost if program execution is halted during the get-off process itself.

The final result of a traffic assignment run may be printed out by simply invoking subroutine SOLVE. A complete main program for a traffic assignment problem is thus given by the following cards:

```
COMMON cards for PARMS (contains  K2 )
CALL INPUT
K2 = 0
CALL TASSGN
CALL SOLVE(1)
STOP
END
```

3.4 NETWORK DESIGN MODULES

The network design problem is the primary focus of the CATNAP code and it is embedded in both the input (RDINV, RDBND) and traffic assignment (SOLVE, TCOST) modules to a considerable extent. Nevertheless, there are several subroutines which are directly related only to the network design problem and these are described in this section.

The main application of decomposition to the network design problem is the determination of the optimum investment for each link as a function of the total traffic flow on that link alone. This so-called link subproblem is solved by subroutine SOLVE (or by TCOST during golden section search); some preliminary calculations are necessary, however, to set up the subproblems. These calculations are performed by the module LKSB (for LinK SuBproblem).

LKSB first computes $\lambda^{1/(k+1)}$ for use with CONS in finding $\phi_j(\lambda)$ from (39). Next the functions $H_j(\cdot)$ and $I_j(\cdot)$ are found for all links with piecewise linear travel and investment costs; the method described in Section 2.3.2 is used. The resulting piecewise linear curves are stored in common block APPX as discussed in Section 3.2.

Once a network design problem with a budget constraint has been solved for a given value of the Lagrange multiplier, the multiplier must be adjusted according to the total expenditure. This is the function of subroutine LSRCH. For the initial problem (in which the multiplier is set to the input value ALAM, see Table 3.2), LSRCH increases the multiplier if the total investment is greater than the budget and decreases it otherwise. The multiplier is further increased (or decreased) in later steps until a pair of investment vectors whose total values straddle the budget is obtained; the step size for changing the multiplier (input parameter DLAM; see Table 3.2) is doubled in each successive step until the budget is straddled.

The situation after the budget is straddled is depicted in Figure 2.4; corresponding to the two multipliers λ_1 and λ_2 are respective total investments G_1 and G_2 with $G_1 > B > G_2$. If either G_1 or G_2 is within BTOL units of B, LSRCH returns a logical value of true to the calling routine to indicate that a satisfactory solution has been obtained; BTOL is an input parameter (see Table 3.2). Otherwise, a new multiplier value between λ_1 and λ_2 is found; this may correspond to a Bolzano (bisection) search in which case we select

$$\lambda = \frac{1}{2}(\lambda_1 + \lambda_2), \tag{65}$$

or a linear interpolation (regula falsi) search may be used with

$$\lambda = \lambda_1 + \frac{G_1 - B}{G_1 - G_2} (\lambda_2 - \lambda_1) \quad . \tag{66}$$

This second approach is currently used in CATNAP, but a change to (65) will be very easy to implement. The linear interpolation method seems to perform better when most improvements have nonlinear $H_j(\cdot)$ functions, while the bisection technique is to be preferred whenever there are many piecewise linear components in the problem.

As mentioned above, LSRCH returns a value of true for its single argument when a satisfactory investment is found; this signals the calling routine to terminate the network design algorithm. A true value will also result when the number of multiplier iterations reaches a preset limit (input parameter KIMAX; see Table 3.2) or when the difference between λ_1 and λ_2 is less than an input tolerance (parameter TLAM; see Table 3.2). If an argument value of false is returned by LSRCH, another traffic assignment problem will be solved for the new multiplier after LKSB has been invoked to set up the sub-problem.

When all the multiplier interations have been completed, CATNAP provides the capability to adjust and evaluate the final solution obtained; this is done by the module ADJUST. For convenience we define here the terms used to describe the various network configurations which may appear in this subroutine:

Final Continuous Solution - The result of the final Lagrange multiplier iteration. Note that this need not be the best solution so far obtained, i.e., the one with investment closest to the budget.

Lower Solution - The solution with total investment closest to the budget from below. If no such solution is found during the multiplier iterations, a solution corresponding to the lower limits L_j on the investment decisions is used.

Upper Solution - The solution with total investment closest to the budget from above; note that the multiplier value for this solution is less than for the lower solution. The configuration corresponding to the upper limits P_j is used if a better one has not been found.

Adjusted Continuous Solution - A solution arrived at by taking a linear combination of the investment values of the upper and lower solutions; this is done so that the budget constraint is always exactly satisfied. A module Z is provided in CATNAP to find the investment decisions Z_j corresponding to the monetary values which arise in the finding the adjusted continuous solution.

Discrete Investment Solution - This solution is the result of applying the heuristic of Section 2.3.6 to the adjusted continuous solution; the required flows f_j^* are determined in a separate traffic assignment run for the adjusted continuous solution.

Besides producing the adjusted continuous and discrete investment solutions, ADJUST also solves one or more traffic assignment problems for the above network configurations. These problems typically do not require many Frank-Wolfe iterations because the existing feasible flow pattern for the final continuous solution is used as the initial feasible solution, and the optimal flow pattern is usually not too different from this one.

A network design problem is first solved for the adjusted continuous solution; this is actually a traffic assignment problem since both L_j and P_j (investment decision bounds) are set to $\bar{Z}_j$, which is the required decision from the interpolation process. The result of this assignment is a set of flows f_j^* that may be used in the discretization procedure if the user specifies that it is to take place.

The final configuration of the network is then determined from the discrete investment solution (if one is found) or else from the adjusted continuous solution. This is done by modifying the capacity and free-flow travel-time parameters for each link depending on the investment decision for that link; for links with time improvements, the resulting model is <u>not</u> identical to the adjusted continuous or discrete investment case with the investments fixed.

The final configuration is then evaluated by means of two more traffic assignment runs: one for a system optimal assignment and the other for user equilibrium. The resulting total-travel times (Z_s^* and Z_u^s) may be used according to (50) to get bounds on Z_u^*. This enables the user to assess the applicability of a system optimal network design solution to a user equilibrium problem.

The user may specify the actions to be taken by ADJUST with a single control card (see Appendix B for formats). The four logical variables set by this card are interpreted as follows:

<u>DISC</u> If true, generate the discrete investment solution; if false, the final configuration will be the adjusted continuous solution.

DUMP If true, the final configuration and final flows are saved on an external data set.

SOTA If true, perform the final configuration system optimal traffic assignment.

UETA If true, perform the final configuration user equilibrium traffic assignment.

A second control card gives the unit number for saving the final flows and configuration; this card need not be present if DUMP is false.

The final function of ADJUST is to save the final vector of investment decisions when the network design run is one stage of an investment staging run. This is added to the end of the output file created by subroutine RDBND: see Section 3.2 for more details.

There is no overall control program for the network design problem to correspond to the TASSGN module for the traffic assignment problem. A main program for network design is not very long, however; a sample is given in Appendix D.

3.5 CAPABILITIES AND LIMITATIONS

As has been mentioned, CATNAP has the capability of solving traffic assignment, network design, and investment staging problems in large transportation networks. In this section we indicate the limitations imposed on CATNAP by the size of the problem to be solved.

The first major difficulty encountered in setting up a really large network problem is the preparation of the input data sets in the correct format. The input modules of CATNAP contain a large number of edit checks to detect clearly erroneous input cards, but the amount of checking that can be done

in this way is limited. An erroneous node number on a link data card or an incorrect trip demand figure, for example, will not be detected.

There are two main limitations which will be encountered as the problem size increases: CATNAP will require more computer main memory, and execution times will increase. The main memory is needed to store data describing the problem; the data structures used are described in Section 3.2. All of the problem-size dependent arrays are stored in named COMMON blocks, which are as follows:

PARMS	67 words; contains basic parameters.
LINK	$2\frac{1}{2}A + \frac{1}{2}N + 3$ words; contains the network topography.
PROB	Maximum [3A + N, 3A + 3C + 22] words; contains the current flows and investments.
TRIPS	$\frac{1}{2}Z^2 + \frac{1}{4}Z + 1$ words; contains the trip table.
INVST	$\frac{1}{2}A + 6\frac{1}{2}C + 2$ words; contains the investment possibilities.
APPX	$\frac{1}{2}A + 44\frac{1}{2}P$ words; contains the piecewise linear approximations.
WORK	Maximum [2C, 3N] words; stores temporary vectors.

The parameters indicated above have the following meanings:

A	number of arcs in the network.
N	highest node index used; this would be the number of nodes if all possible indices are used.

C number of possible distinct investments in the network.

Z highest index for any zone (i.e., origin or destination) in the network.

P number of arcs for which piecewise linear investments are needed.

Note that the core requirements are based on the use of the half word and quarter word storage abilities of IBM System 360/370 computers.

Assuming that the number of investments C lies in the range $N/3 < C < 2N/3$, the total memory needed for the CATNAP arrays is given by

$$6\tfrac{1}{2}A + 3\tfrac{1}{2}N + 9\tfrac{1}{2}C + 44\tfrac{1}{2}P + \tfrac{1}{2}Z^2 + \frac{1}{4}Z + 95 \text{ words} \quad . \tag{67}$$

Since (67) is somewhat unwieldy, we make the following assumptions about the other parameters in terms of the maximum node index, N :

$$A = 3N ,$$

$$Z = N/5,$$

$$C = N/2,$$

$$P = C/3 = N/6.$$

Using these values, (67) becomes

$$\text{Total storage for } N \text{ node problem} = .02N^2 + 35.2N + 95 \text{ words} \quad . \tag{68}$$

To this value must be added the core required by the computer code itself as well as for input-output buffers and system utility routines; when this is done, the results in Table 3.4 are obtained. (Note that 1 Kbyte = 256 words.)

Even though the coefficient of N^2 in (66) and (67) is quite small, this term accounts for most of the storage when $N > 2000$; for this reason, it is suggested that the trip table be organized into separate segments by origin and kept on an external storage device with only the segment for the current origin in FLOWS core-resident at a given time. This would require some straightforward modifications to the modules RDTMX and FLOWS. The resultant memory requirements (see Table 3.4) are within the capability of many large computer installations, particularly those with virtual storage capability.

TABLE 3.4 COMPUTER MEMORY FOR CATNAP

Size of Problem (Nodes)	Traffic Assignment Code (Bytes)	Network Design Code (Bytes)
100	110K	115K
200	120K	130K
500	165K	190K
1000	270K	315K
2000	360K*	450K.
5000	745K*	980K*
10000	1.39M*	1.87M*

*These estimates assume that the trip table is stored externally and is brought into memory 100 origins at a time.

In contrast to the exact bounds which can be found for the memory limitations on CATNAP, the determination of computer time requirements is somewhat more difficult. The code has been tested on problems with up to 1450 nodes, 3800 arcs, and 160 zones, so that time estimates for larger problems are extrapolations. It is nevertheless believed that the figures in Table 3.5 give reasonable values for requirements planning in dealing with large networks:

TABLE 3.5 COMPUTER TIME FOR CATNAP

Size of Problem (Nodes)	Traffic Assignment Code (Minutes)*	Network Design Code (Minutes)*
80	.2**	.5
100	.25	.75
200	.5	2
394	1.3**	3.5**
500	2	7
1000	4.5**	15
1450	6.1**	20
2000	10	30
5000	40	120
10000	90	270

*Times are for the IBM 370 Model 168 Computer, and are based on 20 Frank-Wolfe iterations for the traffic assignment problem and 20 for the network design problem.

**These times were observed in solving actual problems.

3.6 POTENTIAL EXTENSIONS AND MODIFICATIONS

To solve practical transportation network problems, a computer code must be both efficient and flexible. CATNAP, although originally envisioned as an exploratory tool, is both efficient and flexible enough to be used as a production code. The code is organized in a highly modular fashion and improvements are relatively easy to implement.

In this section we outline some extensions and modifications which are potential enhancements for the code. They fall into one or more of the following four general categoties and are given approximately in that order:

a) Broaden the capabilities of the code.

b) Increase the size of problem which can be solved.

c) Improve the efficiency of the code.

d) Enhance user convenience.

CATNAP has the capability of accommodating a variety of travel-time curve formulations besides the FHWA curve (7). Introducing a new nonlinear curve would involve changes in several of the program modules.

As described in Section 3.2, piecewise linear travel-time curves are derived from the basic FHWA curve. If alternative derivations are desired, only module PWLA is affected.

Several enhancements to the network design problem are possible. One would involve the introduction of a piecewise linear convex $G_j(\cdot)$ function with the first investment option (capacity improvement , nonlinear $H_j(\cdot)$ function). Another possibility is to allow for selective discretization, i.e., some links must have discrete investments while others have continuous ones. It would be possible to allow "negative" improvements; e.g., to

obtain an increase in the budget by decreasing the capacity of a given link. This might be useful in a deferred maintenance model.

As mentioned in Section 4.4, the Schimpeler-Corradino heuristic developed in [7] has not been added to CATNAP.

Each of these features could be implemented without much difficulty.

Another extension to the network design problem which might be useful to the planner is the ability to impose several independent budget constraints. The constraints might be non-overlapping, as would be the case for regional budgets relating to disjoint regions; or they could be overlapping in the sense that the same investment opportunity might appear in more than one budget constraint. There are several ways of implementing this feature.

The final improvement in the category of increasing planner options is that of incorporating a rigorous branch-and-bound procedure for finding an optimal network design solution with discrete investments in place of the current heuristic procedure. This would be a major extension.

An extension (which has already been mentioned) that would increase the problem handling capacity of CATNAP is the use of external storage for the trip table in very large networks. It would be easy to modify the modules RDTMX and FLOWS to incorporate this change. Further, the system could be made to accept rectangular, as well as square, trip tables.

Another improvement relating to problem size and efficiency is to incorporate a dynamic core allocation feature so that dimensioning of arrays is "custom-tailored" for the problem in hand.

Currently node numbers are used as indices to locate data relevant to the node, which implies that if there are significant gaps in the node numbering sequence an inefficient use of storage results. By building a referrence array and treating node numbers as node names this inefficiency could be avoided.

There are several algorithm variations which could be explored to improve the efficiency of the code. In the Lagrange multiplier procedure the Frank-Wolfe iterations could be terminated prematurely as soon as it became evident that the current multiplier value was not yielding expenditures close to the budget limit.

The slow convergence of the Frank-Wolfe Algorithm toward the final solution represents an area for further investigation. One possibility is to modify the arc-loading operation in FLOWS so that only a subset of the origins is investigated at a time while holding the flows from the other origins fixed; this modification can require the use of external storage for the fixed-flow vectors. Another possibility, already mentioned, is purposely to choose a sub-optimal ϕ in the golden section search with the hope of finding a better direction of improvement on the succeeding trial solution.

Another improvement in the golden section search procedure would be to treat the non investment arcs in a different and more efficient manner than the investment arcs in the cost calculation phase.

A better initial lower bound could be found in the Frank-Wolfe procedure by solving the traffic assignment problem assuming no congestion.

The shortest path algorithm could be improved in CATNAP by changing the list processing rules to conform to some recently published results by Pape.

When piecewise linear travel cost and investment functions are in use there is an alternative way to solve the subproblems in network design, which could prove to be more efficient than the current method. This procedure is to preprocess and solve the H functions as functions of the Lagrange multiplier for each investment arc. The critical multplier values, where the form of the H function changes, would be recorded.

Lastly there are some improvements which would enhance the user convenience aspects of CATNAP. These features are mainly concerned with input and output organization. A set of default parameter values could be invoked for those parameters not set by the user and more flexibility could be provided for input and output.

4. NUMERICAL RESULTS

In this section, we summarize the results obtained from applying CATNAP to some actual transportation network problems. The intention is to compare the performance of the new code with the results of previous workers as well as to indicate the capability of CATNAP to solve problems which are large enough to be of practical interest to transportation planners.

The first Section (4.1) summarizes the details of the actual test problems which CATNAP has solved. Section 4.2 describes the performance of the Frank-Wolfe traffic assignment algorithm, and Section 4.3 gives results for a comparable network design problem; the latter section also contrasts CATNAP's performance with that of another approach. A comparison of results is also undertaken in Section 4.4 for the investment-staging problem. Finally, Section 4.5 sets forth some recommendations for further testing.

4.1 TEST PROBLEMS SOLVED

Two basic networks have been used in testing CATNAP: A 24-node "toy" network, which is described in Section 4.1.1, and a 394-node network, which is described in Section 4.1.2. The generation of data for the investment staging problem is discussed in Section 4.1.3. A version of a network representing Washington, D.C. has also been run, but only in the traffic assignment mode; this problem contains 1451 nodes and 3738 arcs with 163 zones.

It has been the aim in developing CATNAP to produce a code capable of solving practical transportation problems. The only problem described here in detail which is in any sense "practical" is the 394-node model of Sioux Falls, S.D., discussed in Section 4.1.2; the 24-node example is included here only because the related network design problem is as large

as can be reasonably managed by commercial LP codes. It is also planned to test CATNAP further on some larger problems; see Section 4.5.

4.1.1 Twenty-four Node Sample Problem

The 24-node problem originally presented in the network design study conducted by Northwestern University is reported in [6]. The network has 24 nodes and 76 arcs, and has been developed by the Northwestern researchers from the model of Sioux Falls, S.D., described in Section 4.1.2. An illustration of the network topography is reproduced from [6], and is included as Figure 4.1. The problem has an almost symmetrical 24-zone trip table and a set of 10 possible investments, some of which improve travel time and some capacity.

The original formulation of this problem in [6] included capacity parameters and trip demands in units of vehicles per day; these have been converted to the vehicles per hour figures used in the UROAD code, so that the existing CATNAP input modules could be used. A set of five possible investments was also specified in [6]; these were constrained by the formulation to be "two-way" improvements; i.e., equal investments are required in each direction on the two arcs between a pair of nodes. These investments have been converted into a set of 10 single-arc improvements with the proper capacity and time units for use with CATNAP. The numerical results reported here thus will not be the same as those indicated in [6]; for purposes of comparing solution methods, however, the problems are essentially similar.

4.1.2 Three-Hundred-Ninety-Four Node Sample Problem

The 394-node problem represents the city of Sioux Falls, S.D. It is of moderate size as a highway transportation network; 394 nodes and 1042 arcs with 84 zones. (Note that node indices are set aside for the additional

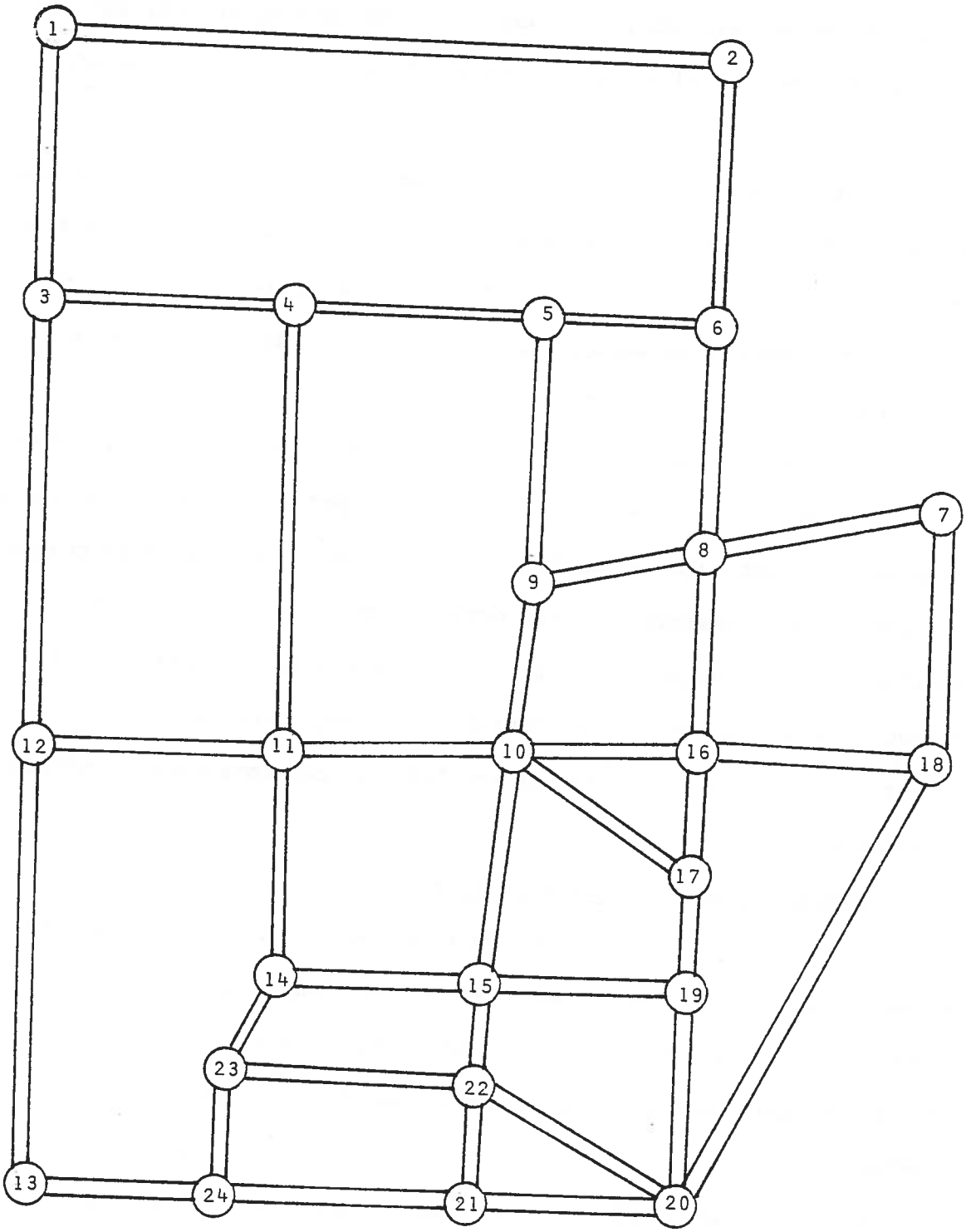


Figure 4.1 -- Twenty-four Node Sample Problem

zones 85 to 99; the actual network topography, then, has only 379 distinct nodes, but the storage and processing-time requirements are the same as if there were 394 actual nodes, and so the latter is used to indicate the size of the problem.)

The trip-demand data for this problem result in fairly severe congestion with vehicle flows on some arcs approaching two or three times practical capacity. The trip matrix is not completely symmetrical, but it is sufficiently so to provide roughly equivalent flows in both directions between pairs of nodes joined by a two-way linkage.

No investment data was provided for this network, and so a set of 103 different possible improvements has been developed; links which the traffic assignment problem solution described in Section 4.2 showed to be congested were chosen for investments. Only capacity improvements have been selected, so that the nonlinear $H_j(\cdot)$ curve formulation (option 1) of Section 2.3.2 can be used. The investment amounts were chosen to be roughly proportional to the actual arc lengths, but considerable variation was allowed.

4.1.3 Investment Staging Sample Problem

The investment-staging results reported in Section 4.4 used the basic 394-node problem discussed in the previous section. The same network topography and investment possibilities were used for each of the four stages; different budgets were adopted, however, and the trip table was also modified.

The basic trip table was used without modification for stage 1. All trip demands were inflated by 20 percent for the stage 4 problem, while a random inflation factor was determined by choosing a pair of randomly distribu

numbers between 0 and 20 for each O-D pair; the minimum of these was the percentage increase for stage 2, while the maximum was used for stage 3.

4.2 TRAFFIC ASSIGNMENT RESULTS

The CATNAP code has been tested on several small problems including the 24-node example of Section 4.1.1. These runs verified that CATNAP operates correctly since the expected results were obtained in each case.

A system optimal traffic assignment problem was also solved for the 394-node example of Section 4.1.2; this was done both to determine the time required by CATNAP to solve such large problems and also to investigate the convergence of the Frank-Wolfe Algorithm. An added benefit was the generation of a vector of flows which was used as a good initial feasible solution for the network design problem.

The results of the traffic assignment are plotted in Figure 4.2 and summarized in Table 4.1. Note that the Frank-Wolfe lower bound gives a fairly good convergence criterion, and that the final approach to the optimal solution is quite slow. Results which are correct to within 3 or 4 percent are obtained after 15 to 20 iterations, however, and this seems reasonable for practical applications.

The problem was solved on the IBM 370 Model 168 computer; as noted in Table 4.2, the total time was less than 4 minutes for 60 Frank-Wolfe iterations. The total computing cost was under 50 dollars.

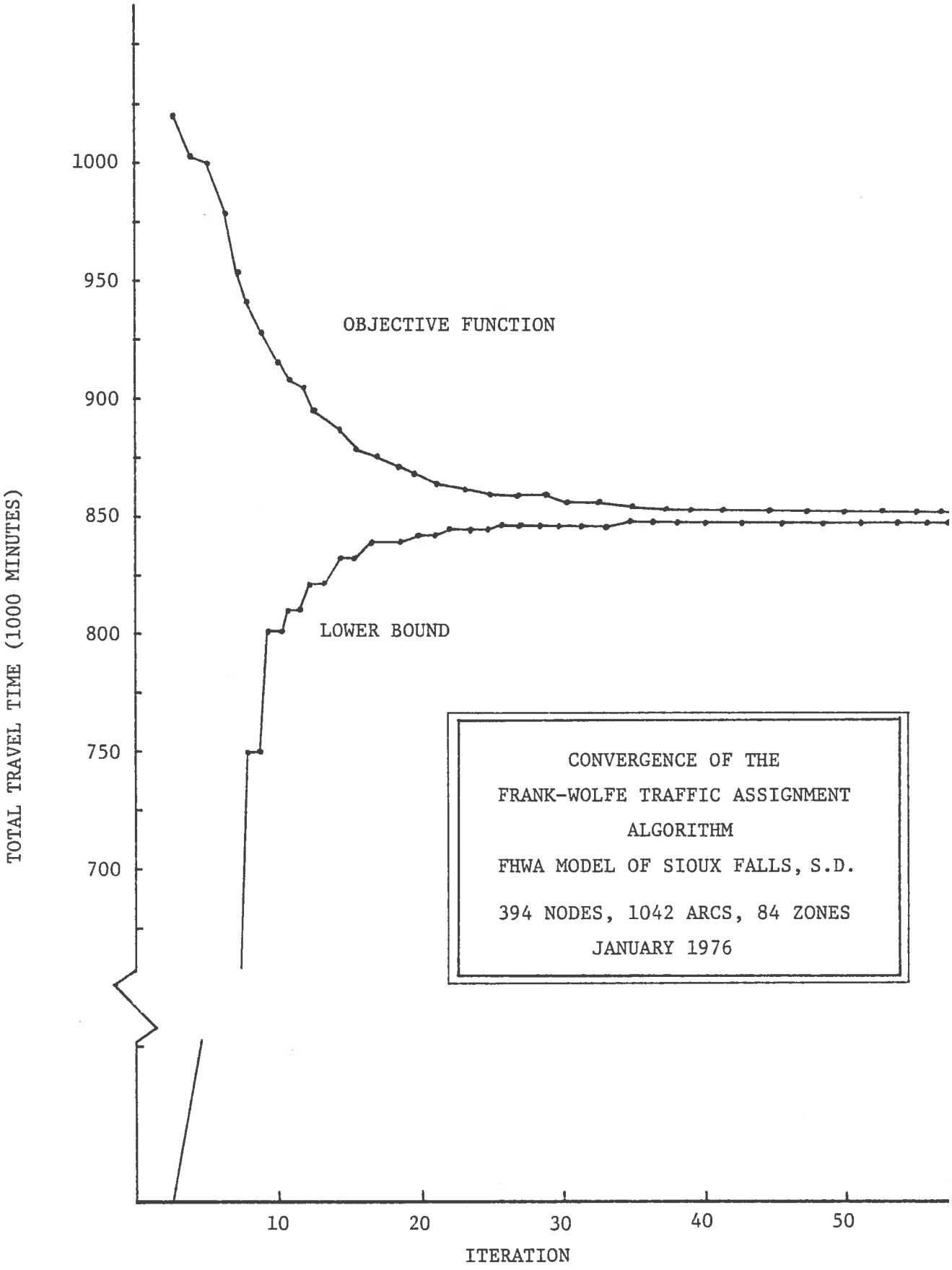


Figure 4.2 -- Traffic Assignment Algorithm Convergence

TABLE 4.1: TRAFFIC ASSIGNMENT ALGORITHM CONVERGENCE

Iteration	Computer Time (minutes)	Total Travel Time (minutes)	Difference of Lower Bound (percent)	Difference of Final Bound (percent)
1	0.26	3,372,100	100.00	74.70
5	0.49	1,010,200	42.45	15.54
10	0.77	922,100	12.60	7.47
15	1.06	891,400	6.24	4.28
20	1.34	876,800	3.80	2.69
25	1.62	869,000	2.38	1.82
30	1.91	865,000	1.78	1.36
40	2.48	860,700	0.99	0.86
50	3.05	858,700	0.68	0.63
60	3.78	857,600	0.51	0.51

4.3 NETWORK DESIGN RESULTS

The first network design problem solved by CATNAP was the 24-node example problem described in Section 4.1.1. Piecewise linear travel-time functions with four segments (the last was unbounded) were used on each of the 76 arcs; the problem as thus formulated is a linear program and may be solved by any general-purpose linear-programming code. This technique was suggested for the network design problem by Morlok, et al. [6].

To compare the two approaches (CATNAP and the LP), a commercial LP code (IBM's MPS/360) was used to solve the 24-node network design problem. The problem has 702 rows and 2868 columns; this is a fairly large linear program (although a very small network), and it required 7929 simplex iterations and over 40 minutes of IBM 370/168 computer time to solve. The total computing cost was over 530 dollars; the optimal objective (total travel time) was 16,285 minutes.

By comparison, CATNAP needed 5 Lagrange multiplier and 93 Frank-Wolfe iterations to solve the same problem; this required just over 10 seconds of computer time for a cost of less than 3 dollars. The final solution resulted in a total travel time of 16,698 minutes which is within 2.5 percent of the optimum found by MPS.

It is probably not completely fair to the Northwestern formulation to use a general-purpose linear-programming code for solution (although that is what is suggested in [6]). A code which uses a shortest path routine to obtain an initial feasible solution together with a column generation scheme to implement the simplex method would certainly improve on the poor performance reported above; the difficulty which remains is the very large size of the linear program even in a small network. This difficulty prevents the Northwestern results from being applied to any practical transportation problems.

The second network design problem was chosen to demonstrate CATNAP's ability to solve such problems in large networks. The 394-node network of Section 4.1.2 was solved in 10 Lagrange multiplier iterations, starting with the feasible flow pattern which was found after the 60 Frank-Wolfe iterations used in the traffic assignment run described in Section 4.2. The network design problem required a total of 110 additional Frank-Wolfe iterations. The solution is summarized in Table 4.2.

A total of 5.63 minutes was needed to solve this problem; the total cost was about 68 dollars. It should be pointed out that the tolerances for this problem were purposely set very tight, and that more Frank-Wolfe iterations were allowed than are strictly necessary. A satisfactory solution can probably be obtained from scratch in about 3.5 minutes for a cost of about 40 dollars. The total cost for both the network design and traffic assignment solutions was about 115 dollars.

The 394-node network design problem was also subjected to final adjustment by subroutine ADJUST; see Section 3.4 for more details of this process. The following results were obtained when this is done (the pre-investment

TABLE 4.2: NETWORK DESIGN ALGORITHM CONVERGENCE

Iteration	Lagrange Multiplier	Total Investment (dollars)	Total Travel time (minutes)	Objective Function Maximum Error (percent)	Frank-Wolfe Iterations
1	0.5000	29,810	778,900	0.31	40
2	0.6000	27,810	779,700	0.28	9
3	0.8000	23,940	782,200	0.32	10
4	1.2000	17,800	787,910	0.29	18
5	2.0000	11,880	796,620	0.30	21
6	1.5783	13,410	793,860	0.29	4
7	1.4414	14,170	792,670	0.32	3
8	1.3861	14,390	792,360	0.36	1
9	1.3530	14,800	791,740	0.31	3
10	1.3425	14,840	791,670	0.31	1

solution is given for comparison):

Solution	System Optimal Total Travel Time (minutes)	User Equilibrium Total Travel Time (minutes)
Pre-investment	857,600	936,540
Adjusted Continuous	791,470	859,480
Discrete Investment	825,060	897,150

It can be seen that the bounds suggested by (50) are not especially tight in this case (the difference is about 8 percent), so that we cannot claim, based on the bounds, that the adjusted continuous system optimal solution is particularly close to the best user equilibrium solution. Note, however, that the degradation in total travel time is comparable for all three cases above, so that the solution may not in fact be far off. Note also the higher travel time which results from the discretization process.

4.4 INVESTMENT STAGING RESULTS

The most practical approach to the investment-staging problem developed prior to the current study seems to be the heuristic method described by Schimpeler-Corradino Associates in reference [7]. This is largely because no other known solution technique can deal with large networks in a reasonable way.

The basic Schimpeler-Corradino method is to compute a cost-benefit ratio for each investment possibility and then to select the "best" investments which can be accommodated within the budget. The cost-benefit ratios are found by comparing a traffic assignment with all of the network improvements present to a traffic assignment on the unimproved network; this is

done initially for the final stage (for which the configuration must be specified) with the budget figure for stage T - 1. The resulting set of investments is then fixed for stage T - 1, a new traffic assignment is performed, and the budget for stage T - 2 is used with the new cost-benefit ratios to determine the stage T - 2 configuration. The procedure is repeated until all stages have been specified.

The original method described in [7] sets forth some very complex means for finding cost-benefit ratios; since these techniques make explicit use of the individual commodity flows f_j^r, they could not be implemented with CATNAP which uses only the aggregated link flows f_j. One heuristic described in [7] could be used, however; this defines the "benefit" on an investment link to be "total vehicle-miles saved." If we let f_{jt} be the flow on link j for the stage t traffic assignment, CA_j be the unimproved capacity, and ℓ_j be the length of the link in miles, then the cost-benefit ratio is given by

$$CBR_{jt} = \frac{G_j(P_j) - G_j(L_j)}{\ell_j(f_{jt} - CA_j - L_j)} , \tag{69}$$

Where P_j and L_j are the bounds on the possible improvement. The ratios found from (69) are then used with the budget B_{t-1} to select the best improvements to be made at stage t - 1.

The main advantage of the Schimpeler-Corradino approach is that it requires only a single traffic assignment (say, 20 Frank-Wolfe iterations) at each stage instead of the more expensive network design problem (about 60

Frank-Wolfe iterations). The disadvantages include the complexity of the method (especially when heuristics more sophisticated than (69) are employed) and the fact that a simple cost-benefit ratio cannot take into account the complex interactions between sets of improvements in the network. Also, the sequence of stages (T, T - 1, ..., 2, 1) in this case is fixed; it is pointed out in [1] that other sequences (for example; T, 1, 2, ..., T - 1) may be more appropriate. Finally, the Schimpeler-Corradino method requires that the final network configuration be specified in advance, and this may require a separate network design run.

The four-period investment-staging problem described in Section 4.1.3 was solved using both the Schimpeler-Corradino method and the lexico-graphic objective function of Section 3.4.1. The cost-benefit ratios (69) and the corresponding investments were determined manually for the Schimpeler Corradino method; this method has not been added to CATNAP. The sequence of stages for the lexico-graphic method was (4, 1, 2, 3). Both solutions used system optimal traffic assignment and continuous investments.

The results of the investment staging runs may be summarized as follow

Stage Stage	Traffic Increase (percent)	Budget (dollars)	Total Travel Time	
			CATNAP (minutes)	Schimpeler-Corradino (minutes)
0	0	0	857,595	857,595
1	0	6000	812,453	819,751
2	Random; about 7	11000	852,122	870,532
3	Random; about 13	15000	917,750	924,328
4	20	17980	991,313	991,313

The proposed CAC method clearly results in lower total travel times than the Schimpeler-Corradino method; the differences are not great however, and the lexico-graphic approach needed about three times as much computer time (cost per stage of about 95 instead of 35 dollars).

4.5 RECOMMENDATIONS FOR FURTHER STUDY

The development of a computer code useful to transportation planners requires that a variety of real problems be solved during the code's development. This not only permits testing and debugging of the code, but also allows the introduction of enhancements which increase its utility to the eventual user.

CATNAP has thus far been used only for the moderate-size 394-node network discussed in Section 4.1.2 plus a preliminary traffic assignment case for the larger Washington, D.C. model. It is planned to solve several larger problems in the near future; models of Minneapolis - St. Paul, Minnesota and Washington, D.C. with up to 1500 nodes are expected to be used for this purpose.

The authors feel that the testing of CATNAP's network design and investment staging capabilities should be continued. In particular there is a real need for network design problem with actual investment data to extend the results reported here to real problems.

REFERENCES

[1] Dantzig, G.B., Maier, S.F., and Lansdowne, Z.F., "Application of Decomposition to Transportation Network Analysis," Control Analysis Corporation Technical Report No. DOT-TSC-OST-76-26, October 1976.

[2] Dijkstra, E.W., "A Note on Two Problems in Connexion with Graphs," Numer. Math., vol. 1, p. 269-281, 1959.

[3] Luenberger, D.G., Introduction to Linear and Nonlinear Programming, Addison-Wesley, Reading, Mass., 1973, p. 134-137.

[4] McKnight, R.W., "Greatest Lower Bound on a Convex Function of Unknown Nature," Control Analysis Corporation Technical Report, No. R-60-FG, November 1976.

[5] Robinson D.W., "Analysis of a Shortest Path Algorithm for Transportatic Applications," Control Analysis Corporation Technical Report No. R-53-I March 1976.

[6] Morlok, E.K., Schofer, J.L., et. al., Development and Application of a Highway Network Design Model, (Final Report Prepared for Federal Highwa Administration, Environmental Planning Branch), Department of Civil Engineering, Northwestern University, Evanston, Illinois, 1973.

[7] Schimpeler-Corradino Associates, Optimum Staging of Projects in a Highv Plan, Report Prepared for Federal Highway Administration, Office of Highway Planning, Louisville, Kentucky, March 29, 1974.

[8] Dial, R.B., Algorithm 360, "Shortest-Path Forest with Topological Ordering," Communications of the ACM, vol. 12, no. 11, p. 632-633, November 1969.

[9] Frank M., and Wolfe, P., "An Algorithm for Quadratic Programming," Naval Research Logistics Quarterly, vol. 13, p. 95-110, 1956.

APPENDIX A

CODE LISTING

A complete listing of the CATNAP code modules together with a sample main program is distributed under separate cover and is available from Control Analysis Corporation.

APPENDIX B

INPUT FORMATS

SUBROUTINE INPUT

Control Cards:

1. Title Card. Problem title in columns 1-72 (Format 18A4).

2. Stage Card. Stage number right-justified in colums 1-4 (Format I4). Field should be blank or zero if this is not an investment-staging run.

3. Integer Parameters. Each parameter is right-justified in its four-column field (Format I4); see Table 3.2 for meanings.

Columns	Parameter	Columns	Parameter
1-4	K1MAX	25-28	IOINV
5-8	K2MAX	29-32	IOBND
9-12	KOUT1	33-36	IOPWL
13-16	KOUT2	37-40	ISTART
17-20	IOLD	41-44	IDUMP
21-24	IOTMX	45-48	IFLOW

4. Logical Parameters. A T (for true) or F (for false) is punched within the four-column field specified below (Format L4); see Table 3.2.

Columns	Parameter	Columns	Parameter
1-4	PWT	17-20	PRINTP
5-8	PWI	21-24	PRINTL
9-12	LP	25-28	PRINTD
13-16	UE	29-32	PRINTS

5. Real Parameters. Each value is punched with a decimal point within the nine-column field given below (Format F9.0); see Table 3.2.

Columns	Parameter	Columns	Parameter
1-9	FWTOLI	37-45	BTOL
10-18	FWTOLO	46-54	ALAM
19-27	PHITOL	55-63	DLAM
28-36	BUD	64-72	TLAM

SUBROUTINE RDLD

Link Data Card:

Columns	Format	Description
2-6	I5	Origin node for link.
8-12	I5	Destination node for link.
14-17	I4	Length of link (tenths of miles).
18	A1	Type indicator. If S, the following number is the speed in miles per hour; if T, it is the time in tenths of minutes.
19-21	I3	Time/speed field for traffic on the origin to destination link ("forward link").
32-36	I5	Vehicle flow in vehicles per hour at which the time/speed value is measured.
37	I1	Number of lanes for the forward link.
41	A1	Type indicator as in column 18 (Note 1).
42-44	I3	Time/speed field for the link from destination to origin ("reverse link").
55-59	I5	Vehicle flow as in columns 32-36.
60	I1	Number of lanes for the reverse link.
66	I1	Facility type index (Note 2).
68	I1	Area type index (Note 2).

NOTES:

1. The type indicator field for the reverse link (Column 41) may have the following additional values:

(Blank) No reverse link (values may be given on another card).

U Reverse link parameters are identical to forward link.

X Reverse link parameters are identical to forward link, except reverse capacity is one-half forwarded.

2. There are five different facilities and five different areas specified by FHWA. Each combination has its own practical capacity per traffic lane according to the following table (values are in vehicles per hour):

AREA TYPE	FACILITY TYPE				
	1 (Freeway)	2 (Expressway)	3 (2-way Art)	4 (1-way Art)	5 (Cent. Con.)
1 (Central Business District)	1312	600	450	525	7500
2 (Fringe)	1312	750	412	412	7500
3 (Residential)	1312	825	412	675	7500
4 (Outer CBD)	1312	750	412	688	7500
5 (Rural)	1312	825	412	1425	7500

New Link Card:

Columns	Format	Description
2-6	I5	Origin node for link.
8-12	I5	Destination node for link.
16	A1	Reverse link indicator. If this is U a reverse link may be built; if not, only the forward link may be built.

New links are initialized with a capacity of 0.001 and a time of 1000 minu

SUBROUTINE RDTMX

Trip Demand Card:

Columns	Format	Description
1-4	I4	Origin node.
5-8	I4	Destination node.
12-14	I3	First trip table index on this card (Note 1).
15-16	I2	Number of trip table entries on this card (Note 1).
17-24 25-32 ... 65-72	I8	Number of trips required from origin to destination in each trip table.

NOTE:

1. A single input parameter to RDTMX specifies which trip table is desired by CATNAP. Up to seven different demands may be specified on each card, each demand corresponding to a different trip table index.

SUBROUTINE RDINV

Columns	Format	Description
1-4	I4	Origin node for link.
5-8	I4	Destination node for link.
9-20	F12.2	Total cost of improvement.
21-29	F9.4	Post-investment free-flow link travel time.
30-38	F9.4	Post-investment link capacity.

SUBROUTINE RDBND

Staging bounds are read from an unformatted file created by a previous network design run.

Bound Card:

Columns	Format	Description
1-4	I4	Origin node for link.
5-8	I4	Destination node for link.
9-20	F12.5	Lower bound on investment (monetary units).
21-32	F12.5	Upper bound on investment (monetary units).

SUBROUTINE ADJUST

Control Cards:

Read from data set 5 immediately following the last control card from INPUT.

1. Logical parameters. A T (for true) or F (for false) is punched within the columns given below (format L4).

Columns	Parameter	Description
1-4	DISC	If true, adjust all investments to discrete values; if false, leave them continuous.
5-8	DUMP	If true, save the final network configuration and the final flows on disk.
9-12	SOTA	If true, do a systems optimal traffic assignment on the final network configuration.
13-16	UETA	If true, do a user equilibrium traffic assignment on the final network configuration.

2. Data Set Number. Gives the data set number for saving the final network configuration right-justified in columns 1 to 4 (Format I4). This card need not be present if DUMP is false.

APPENDIX C

JOB CONTROL LANGUAGE

The CATNAP code has been writtin in IBM's version of FORTRAN IV, and it is intended to be run on System 360/370 computers. In particular, extensive use is made of special data types (half-word integers, quarter-word logical variables), subroutine linkage conventions (e.g., ENTRY statements), and input-output feastures (END = option on the READ statement), all of which are "peculiar" to the IBM implementation of the language. Modification of the code for another computer system will be somewhat involved but would not change the basic structure of CATNAP; there could be a considerable cost in storage however, if all the half-word integer arrays were increased to full-word storage.

Because of the dependence of the current version of CATNAP on 360/370 computers, this section provides a summary of the job control language needed to solve network problems using the code as an aid to the prospective user. It is assumed that the reader is familiar with basic OS/360 job control language. Note that some of the examples given here may be incorrect at some installations; therefore, they must be checked with local documentation.

It is suggested that the source modules for CATNAP be separately complied, and the object decks be kept in a subroutine library. Such a library requires a total space of a least 14 6444-byte blocks (14 tracks on a model 2314 disk, 7 tracks on a model 3330), with sufficient directory blocks for 21 different modules. Members may be added to the library using a standard FORTRAN compile/link-edit procedure with the parameter NCAL set for the linkage editor. See Example 1 for details. (Members in the library may be revised by specifying DISP = ØLD on the last card.)

With all the of the CATNAP modeuls in a library, the running of a given network problem requires only a small source language input. The sample traffic assignment main program at the end of Section 3.3 or the network design program of Appendix D are combined with the BLØCK DATA subprogram given with the code listing in Appendix A; see Example 2.

The BLØCK DATA subroutine sets various constants used in CATNAP. These include all data set reference numbers (see Table 3.3 and below) as well as the following parameters:

NAMAX	Maximum number of arcs in the problem.
NCMAX	Maximum number of distinct investments in the problem.
NNMAX	Maximum number of nodes in the problem.
NTMAX	Maximum number of piecewise linear curves in the problem
NZMAX	Maximum number of zones in the problem.
R	FHWA curve parameter r (currently set to 0.15).
KEXP	FHWA curve parameter k (currently set to 4).
NT	Number of piecewise linear segments desired in the $T_j(\cdot)$ curve approximation (between 2 and 5).
RK	Constant equal to $r(k + 1)$.

The firs t five parameters above give the dimensionality of the arrays set up for CATNAP in the various CØMMØN blocks, and thus represent upper limits; specific values for the number of nodes, etc., in a given problem are determ during data input.

The size of the various data sets required by the CATNAP code depends largely upon the array dimension parameters. There are two basic types of d

```
// EXEC FØRTCL,PARM.LKED='NCAL,MAP,LIST'
//FØRT.SYSIN DD *
      { Cards for module ADJUST
      {
/*
//LKED.SYSLMØD DD DSN=CATNAP.LIB(ADJUST),VØL=SER=PUBOO1,UNIT=DISK,
//   DISP=MØD
```

Example 1. Adding a module to the subroutine library; specific details of the catalogued procedure may differ between installations.

```
// EXEC FØRTCLG
//FØRT.SYSIN DD *
{     Cards for main program
{         and BLØCK DATA subroutine
/*
//LKED.SYSLIB DD DSN=CATNAP.LIB,VØL=SER=PUBOO1,UNIT=DISK,DISP=SHR
```

Example 2. Source language input for a CATNAP run. Many OS implementations require a slightly different specification for the subroutine library.

set: raw input files which contain formatted card images, and compact files which contain unformatted internal arrays which are saved externally.

The raw input data sets in CATNAP are as follows:

Number	Symbol	Use	Specifications
5*	ICØNT	Control cards	Card reader input (SYSIN).
6*	IØUT	Normal output	Line printer output (SYSØUT=A)
6*	IERRU	Error messages	Line printer output (SYSØUT=A)
41	LINKD	Link data	80-byte card images
42	NEWLD	New links	80-byte card images
43	ITRIPD	Trip matrix	80-byte card images
44	INVD	Investments	80-byte card images
45	IBND	Investment bounds	80-byte card images
54	LPØUT	LP formulation	80-byte card images (output)

*These data set reference numbers should conform to the installation-specific assignments for the card reader (5 is used here) and line printer (6).

These data sets may consist of actual cards or may be files of card images on tape or disk. Note that not all of these need exist for a particular CATNAP run; those not needed may be DUMMY'ed out, if desired. See Example 3; in this case, the link data are read from disk, there are no new link data, the trip matrix is read from tape, and the investment data from cards. Also, the LP output is directed to the punch. See Appendix B for the required formats for the input cards.

```
{Source language input cards
    (see Example 2)
//GØ.FT41F001 DD DSN=CATNAP.INPUT.LINK,VØL=SER=PUB002,UNIT=DISK,
//  DISP=ØLD
//GØ.FT42F001 DD DUMMY

//GØ.FT43F001 DD VØL=SER=CAC45,UNIT=TAPE9,LABEL=(,NL),DISP=ØLD
//GØ.FT44F001 DD *
    {Cards for investment
          data
/*
//GØ.FT45F001 DD DUMMY
//GØ.FT54F001 DD SYSØUT=B
//GØ.SYSIN    DD *
    {Control Cards
/*
```

Example 3. Raw data file specifications for a CATNAP run. It is assumed that the catalogued procedure describes the line printer output data set.

All the remaining data sets used by CATNAP are initially created by dumping internal program arrays onto an external storage device; the arrays are not written under format control both to save time and to prevent any roundoff errors when the data are subsequently read back in. Data sets for FORTRAN unformatted write statements must be created using the VS or VBS options for RECFM; to conserve disk storage space, it is recommended that the VBS option (Variable-length, Blocked, Spanned records) be used. When using this option, it is necessary to specify the block size (which may be selected for maximum efficiency with the external storage device) and the length in bytes of the largest logical record to be written.

The internal arrays saved in the compact files generally coincide with an entire COMMON block; thus the record length for the data set depends on the problem size. Using the array dimension parameters above, we have the following requirements for compact data sets in CATNAP:

Number	Symbol	COMMON Blocks and Arrays Saved	LRECL
46	LINKS	LINK	$10 \cdot NMAX + 2 \cdot NNMAX + 16$
47	ITRIPS	TRIPS	$2 \cdot NZMAX^2 + NZMAX + 8$
48	INVS	INVST, array T in LINK	$6 \cdot NAMAX + 26 \cdot NCMAX + 12$
49	IBNDS	Arrays in PROB and INVST	$16 \cdot NCMAX + 16$
50	IPWLS	APPX, array IJ in LINK	Depends on the size of APPX; roughly $(40 \cdot NT-18)\ NTMAX + 4 \cdot NAMAX + 24 \cdot NCMAX + 4$
51 52 53	IRST IGTOF1 IGTOF2	PROB, APPX	Depends on the size of APPX; roughly $(40 \cdot NT-18)\ NTMAX + 12 \cdot NAMAX + 36 \cdot NCMAX + 92$
55	IFLOW	Array F1 in PROB	$4 \cdot NMAX + 4$
56	ISOL	LINK	$10 \cdot NMAX + 2 \cdot NNMAX + 16$
60 61	IBDIN IBDOUT	Array of investments	$4 \cdot NCMAX + 8$

Note that the maximum LRECL which can be specified is 32,768 bytes; if the logical record length is larger than this, the parameter may simply be omitted from the job control language.

The DCB parameters (RECFM, LRECL and BLKSIZE) described above and the SPACE parameters for these data sets need be specified only when the file is first created. To determine the space parameter, note that in most cases only a single record is written on the data set; exceptions are ISOL (two records) and IBDOUT (one record for each stage already solved plus one additional). The required number of tracks, of course, depends on the particular disk drive available.

An example of the compact data set specifications is included as Example 4; note the omission of the LRECL parameter from data sets 52 and 53, and the use of previously assigned data sets (DISP=OLD).

```
{ Source language input and raw data cards
        (see Examples 2 and 3)

//GØ.FT46F001 DD DSN=CATNAP.SAVE.LINK,VØL=SER=PUB004,UNIT=DISK,
//    DISP=(NEW,KEEP),DCB=(RECFM=VBS,LRECL=16016,BLKSIZE=6444),
//    SPACE=(TRK,(2,1))
//GØ.FT47F001 DD DSN=CATNAP.SAVE.TRIP,VØL=SER=PUB001,UNIT=DISK,
//    DISP=ØLD
//GØ.FT48F001 DD DUMMY
//GØ.FT49F001 DD DUMMY
//GØ.FT50F001 DD DUMMY
//GØ.FT51F001 DD DSN=CATNAP.GETØFF1,VØL=SER=PUB002,UNIT=DISK,
//    DISP=ØLD
//GØ.FT52F001 DD DSN=CATNAP.GETØFF2,VØL=SER=PUB003,UNIT=DISK,
//    DISP=(NEW,KEEP),DCB=(RECFM=VBS,BLKSIZE=6444),SPACE=(TRK,(3,1))
//GØ.FT53F001 DD DSN=CATNAP.GETØFF3,VØL=SER=PUB004,UNIT=DISK,
//    DISP=(NEW,KEEP),DCB=(RECFM=VBS,BLKSIZE=6444),SPACE=(TRK,(3,1))
//GØ.FT55F001 DD DUMMY
//GØ.FT56F001 DD DUMMY
//GØ.FT60F001 DD DSN=CATNAP.STAGE2,VØL=SER=PUB001,UNIT=DISK,
//    DISP=ØLD
//GØ.FT61F001 DD DSN=CATNAP.STAGE3,VØL=SER=PUB005,UNIT=DISK,
//    DISP=(NEW,KEEP),DCB=(RECFM=VBS,LRECL=1008,BLKSIZE=6444),
//    SPACE=(TRK,(1,1))
```

Example 4. Compact data sets needed for a CATNAP run. SPACE parameters will depend on the type of disk drives available.

APPENDIX D

SAMPLE NETWORK DESIGN MAIN PROGRAM

The following main program may be used to solve network design problems:

```
      COMMON /PARMS/ TITLE(18), BTOL,   BUD,    DLAM,   EPS,    FWTOLI,
     +               FWTOLO,   IBDIN,  IBDOUT, IBND,   IBNDS,  ICONT,
     +               IDUMP,    IERRU,  IFLOW,  IGTOF1, IGTOF2, INVD,
     +               INVS,     IOUT,   IPWLS,  IRST,   ISOL,   ISTART,
     +               ITRIPD,   ITRIPS, KEXP,   KOUT1,  KOUT2,  K1MAX,

     +               K2MAX,    LINKD,  LINKS,  LPOUT,  NAMAX   NCMAX,
     +               NERR,     NEWLD,  NNMAX,  NS,     NT,     NTMAX,
     +               NZMAX,    PHITOL, R,      RK,     TLAM,   LP,
     +               PRINTD,   PRINTL, PRINTP, PRINTS, PWI     PWT,
                     TA,       UE

      LOGICAL*1 LP,     PRINTD, PRINTL, PRINTP, PRINTS, PWI,    PWT
     +          TA,     UE
C
      COMMON /PROB/ Z1, ZI , ZT,  ZN, ZLIM,  K1, K2, ALAM,   PHI, KH,  KL,
                    AL1, AL2, ZI1,  ZI2, ZT2,  ALAMK, F1(1500),   F2(1500),

     +              CS(1500), AINV(500),  AINV1(500),  AINV2(500)
      REAL*8 Z1, ZI, ZT, ZN, ZLIM
C
      LOGICAL DONE
C
C ---  READ PROBLEM DATA
      CALL INPUT
C     CHECK RESTART/GETOFF FLAG
```

```
      IF ( ISTART .GT. 0 ) GO TO 100
C
C --- LAGRANGE MULTIPLIER LOOP BEGINS HERE
C
C     SET UP LINK SUBPROBLEM SOLUTIONS

   50 CALL LKSB
      K2 = 0
      ZLIM = 0.0
C
      SOLVE TRAFFIC ASSIGNMENT PROBLEM
  100 CALL TASSGN
      K2 = K2 + 1
C     PRINT PROBLEM SUMMARY (IF REQUIRED)
      N = 0
      IF (MOD(K1, KOUT1) .EQ. 0 ) N = 1
      CALL SOLVE ( N )
C
C     GET NEW LAMBDA
      CALL LSRCH (DONE)
      IF ( DONE ) GO TO 150
      K1 = K1 + 1
C
C     END OF MULTIPLIER LOOP
      GO TO 50
```

```
C --- PRINT FINAL CONTINUOUS SOLUTION
  150 IF ( UE ) CALL UETOSO
      IF ( UE ) ZLIM = 0.0
      CALL SOLVE ( 1 )

      WRITE (IOUT, 200)
  200 FORMAT(//'OFINAL CONTINUOUS SOLUTION')
C
C --- FINAL ADJUSTMENT (NETWORK DESIGN PROBLEM ONLY)
      IF ( .NOT. TA ) CALL ADJUST

C
C --- ALL DONE
      STOP
      END
```

APPENDIX E

DEFINITION OF SYMBOLS

In this Appendix are gathered for ready reference definitions of the symbols appearing in the mathematical formulas in this report. Each symbol is also defined in the body of the report where it is first used.

Symbol	Definitions
A	The set of arcs in the network.
B	The total budget amount for improvements in the network design problem.
B_t	The total budget for periods 1, 2, ..., T in an investment staging problem.
$C_j(\cdot)$	The average (per unit) travel cost function for travelers on link j.
C_j^m	The slope of the m^{th} segment in a piecewise linear travel cost function for link j.
CA_j	The practical capacity of link j (measured in vehicles/hour, trips/day, etc.)
CBR_{jt}	The Schimpeler-Corradino cost-benefit ratio for the improvement on arc j to be undertaken in time period t.
$D_j(\cdot,\cdot)$	The travel cost function for arc j for a specified flow (first argument) and investment (second argument).
$D_{jt}(\cdot,\cdot)$	The travel cost function for arc j in stage t of an investment staging problem.

Symbol	Definitions
f_j	The total flow on arc j.
f_j^r	The flow on arc j which originated at origin r.
f_j^*	The optimum total flow on arc j.
f_{jt}	The total flow on arc j at stage t of an investment staging problem.
f_{jt}^r	The flow on arc j at stage t which originated at origin r.
F_j^m	The multiplier which specifies the change in the width of the m^{th} segment of the piecewise linear travel cost curve for arc j for each unit of improvement on the arc.
$F1_j$	The total flow on arc j at the beginning of a Frank-Wolfe iteration.
$F2_j$	The trial flow on arc j generated by the Frank-Wolfe procedure.
g_j	The unit cost of improvement on arc j.
g_j^n	The slope of the n^{th} segment of the piecewise linear improvement cost curve for arc j.
G_j^n	The width of the n^{th} segment of the piecewise linear improvement cost curve for arc j.
$G_j(\cdot)$	A function which gives the cost (in monetary units) of a given improvement (in capacity units) for arc j. This is normally a linear (slope g_j) or piecewise linear (slopes g_j^n, segment widths G_j^n) curve.

Symbol	Definition
h_i^r	The net supply (positive) or demand (negative) at node i for trips originating at origin r.
h_{it}^r	The net trip supply/demand at node i for origin r during stage t of an investment staging problem.
$H_j(\cdot)$	A function giving the minimum social transportation cost (weighted combination of travel time and investment dollars) for arc j as a function of flow.
i	A subscript used to index the nodes in the network; $i \varepsilon N$.
$I_j(\cdot)$	A function giving the optimum improvement decision (i.e., the one minimizing social transportation cost) for arc j for a fixed value of flow on the arc.
j	A subscript which is used to index the arcs in the network; $j \varepsilon A$.
k	The exponent used in the FHWA congestion function (7).
K_j^m	The width of the m^{th} segment in a piecewise linear travel cost function for arc j.
ℓ_j	The length of arc j in miles.
L_j	A lower bound on the improvement to be undertaken on arc j (capacity units).

Symbol	Definitions
m	A subscript used to index segments on the piecewise linear cost (travel time) curve on each arc j; m = 1, ..., M_j.
M_j	The number of segments in the piecewise linear travel cost curve for arc j.
n	A subscript used to index segments on the piecewise linear improvement cost curve for each arc j; n = 1, ..., N_j.
N	The set of nodes in the network.
N_j	The number of segments in the piecewise linear improvement cost curve for arc j.
O_{ij}	The number of trips which originate at origin i and terminate at destination j.
P_j	An upper bound on the improvement to be undertaken on arc j.
r	(1) A subscript used to index nodes from which trips may originate ("origins"); r = 1,...,R. (2) The multiplier used in the FHWA congestion function Eq (7).
R	The number of origin nodes in the network; these are taken to have indices 1,...,R.
t	A subscript used to index periods in the planning horizon for an investment staging problem; t = 1, ..., T.
t_j	The free-flow (uncongested) travel time for arc j.

Symbol	Definitions
T	The number of time periods in an investment staging problem.
$T_j(\cdot)$	The total travel cost function for arc j.
U_t	The total travel cost (summed over all arcs) for stage t in an investment staging problem.
V_i	The set of arcs which terminate at node i.
W_i	The set of arcs which originate at node i.
X_j^m	A decision variable giving the amount of the m^{th} segment of the piecewise linear travel cost curve for arc j which is needed to match the total flow on the arc.
z_j	The improvement decision for arc j; measured in capacity units.
z_j^*	The optimum improvement for arc j.
z_{jt}	The improvement decision for arc j at stage t of an investment staging problem.
z_j^a	A discrete improvement decision for arc j which is less than the optimum decision z_j^*.
z_j^b	A discrete improvement decision for arc j which is greater than the optimum decision z_j^*.
Z	The objective function value for a traffic assignment (total travel cost) or network design (total social transportation cost) problem.