DOT-TSC-RSPA-79-20

THREE GEOGRAPHIC DECOMPOSITION APPROACHES IN TRANSPORTATION NETWORK ANALYSIS

G.B. Dantzig, K.A. Epstein, R.P. Harvey, Z.F. Lansdowne
S.F. Maier, R.D. McKnight, and D.W. Robinson

Control Analysis Corporation
285 Hamilton Avenue
Palo Alto CA 94301

Final Report
March 1980

DOT-TSC-RSPA-79-20

Prepared for

U.S. Department of Transportation
Research and Special Programs Administration
Office of Transportation Programs Bureau
Office of Systems Engineering
Washington DC 20590

NOTICE

This document is disseminated under the sponsorship of the Department of Transportation in the interest of information exchange. 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.

4. Title and Subtitle
THREE GEOGRAPHIC DECOMPOSITION APPROACHES IN TRANSPORTATION NETWORK ANALYSIS

5. Report Date
March 1980

6. Performing Organization Code

7. Author(s)
G.B. Dantzig, K.A. Epstein, R.P. Harvey, Z.F. Lansdowne, S.F. Maier, R.D. McKnight, and D.W. Robinson

8. Performing Organization Report No.
DOT-TSC-RSPA-79-20

9. Performing Organization Name and Address
Control Analysis Corporation*
800 Welch Road
Palo Alto CA 94304

10. Work Unit No. (TRAIS)
RS905/R9516

11. Contract or Grant No.
DOT-TSC-1059

12. Sponsoring Agency Name and Address
U.S. Department of Transportation
Research and Special Programs Administration
Office of Transportation Programs Bureau
Washington DC 20590

13. Type of Report and Period Covered
Final Report
June 1976 - March 1979

14. Sponsoring Agency Code

15. Supplementary Notes
*Under contract to: U.S. Department of Transportation
Research and Special Programs Administration
Transportation Systems Center
Cambridge MA 02142

16. Abstract

This document describes the results of research into the application of geographic decomposition techniques to practical transportation network problems. Three approaches are described for the solution of the traffic assignment problem. One approach uses geographic decomposition for the solution of shortest path problems. The second and third approaches use geographic decomposition to solve directly the traffic assignment problem with fixed demands. The second approach uses the Generalized Benders Method and the third approach uses the Dantzig-Wolfe decomposition method.

The first two approaches have been implemented via computer codes. In this report the algorithms and the basic structure of the computer codes are described, and preliminary numerical results are given. The work is regarded primarily as providing basic tools for further research into geographic decomposition methods.

17. Key Words	18. Distribution Statement
Budget Constraint Network Design and Analysis Traffic Assignment Transportation Network	Document is available to the public through the National Technical Information Service, Springfield, Virginia 22151

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

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

* U. S. GOVERNMENT PRINTING OFFICE 1979 725-504/329

PREFACE

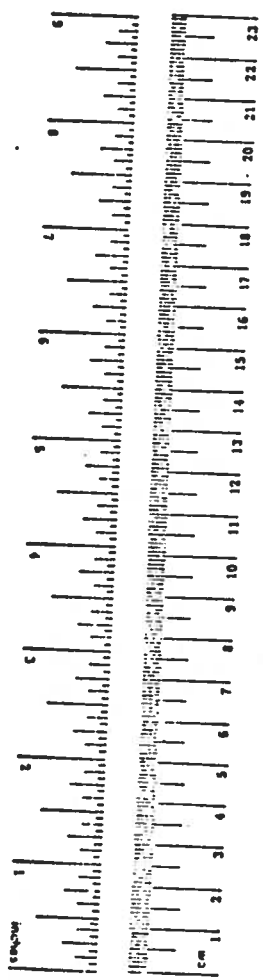
The research herein reported was funded by the Systems Management and Control Program under the auspices of the Research and Special Programs Administration, U.S. Department of Transportation. Technical review is the responsibility of the Systems Evaluation Branch, Research Division, Transportation Systems Center. The objective of this program is to stimulate the basic scientific research areas that are of major importance to the U.S. Department of Transportation. This particular project is intended to develop computerized algorithms that will permit network analysis techniques to be applied to very large networks.

This report describes three approaches to geographic decomposition in transportation network analysis including the implementation, preliminary testing and performance of two geographic decomposition algorithms, one of which was presented in an earlier Control Analysis Corporation report.

METRIC CONVERSION FACTORS

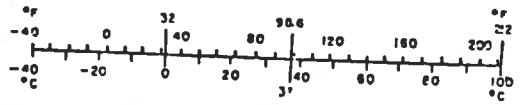
Approximate Conversions to Metric Measures

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
	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

ILLUSTRATIONS AND TABLES

1. INTRODUCTION

1.1 BACKGROUND

The research reported in this study is concerned with the solution of large network models of transportation systems and is part of a program of research into the use of decomposition techniques in this area. A survey of applicable transportation network problems together with proposed decomposition algorithms for their solution was presented in the report "The Application of Decomposition to Transportation Network Analysis" by G. B. Dantzig, S.F. Maier and Z. F. Lansdowne [1].

A computer code called CATNAP (Control Analysis Transportation Network Analysis Program) has also been developed. It implements some of these decomposition ideas for solving the following problems:

a) Highway traffic assignment with fixed demands

b) Highway network design with or without a budget constraint

c) Optimal staging of network investments over time.

The CATNAP implementation is described, with numerical results and computer run times, in the report "Computer Code for Transportation Network Design and Analysis" by R. P. Harvey and D. W. Robinson [2].

In this report we discuss preliminary research on the use of geographic decomposition techniques in the solution of large transportation network problems. Three approaches are described: in one, geographic decomposition is applied to the shortest path module within a traffic assignment algorithm and in the second and third, geographic decomposition is applied directly to the traffic assignment problem. Computer codes have been written and tested which implement the first two approaches. Both codes are designed to solve only the highway traffic assignment problem with fixed demands, in contrast to the more general capability of CATNAP.

The objective of the highway 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 consumption for example, or accident hazard level.

The decomposition techniques in earlier reports have involved breaking up a large problem mathematically into smaller (and more tractable) parts. In this report the decomposition we are using has a physical interpretation as well as a mathematical one: the network being analyzed is broken into separate parts each of which covers a particular geographic region.

There are at least three reasons for wishing to study geographic decomposition of transportation networks. In the first place, some networks are simply too large to be handled as a single entity in computer processing operations and geographic decomposition may then allow the problem to be solved piecemeal within the size limits of the computer without loss of accuracy. Secondly, it may be possible to use geographic decomposition to reduce the overall running time, as it may be more efficient to solve a series of smaller problems, rather than solving the original problem. Thirdly, it is often desired to change the network within a small area and to investigate the effect of the change without having to obtain a detailed solution for the whole network. We do not pursue this third motivation for studying geographic decomposition methods except to indicate how one of the three methods might be used in this connection.

This report will study the usefulness of applying the first two geographic decomposition approaches for solving the highway traffic assignment problem. We will show that the method based upon decomposing the shortest path problem (which we refer to throughout as the "SP Method") can save substantial amounts of computer memory and running time when compared with the non-decomposed method; while the method based upon directly decomposing the traffic assignment problem (which is called the "BD Method" since it is an application of generalized Benders' decomposition) did not perform as well as the non-decomposed algorithm on our test problems.

In this section we briefly describe these three approaches and discuss potential applications and extensions. In the remainder of the report a more detailed approach is adopted: Section 2 contains a mathematical formulation of the traffic assignment problem, a discussion of network partitioning techniques and a description of the sample network test problems used in this study; Sections 3, 4 and 5 contain descriptions of the three algorithms and Section 6 contains a summary for this phase of research.

1.2 SHORTEST PATH GEOGRAPHIC DECOMPOSITION

A network consists of a set of nodes connected by directed links, where each link has a certain length. The "shortest path problem" is to find the shortest directed path between two given nodes of the network and this problem occurs in many applications. In particular, it is the basic subproblem in the solution of the traffic assignment problem with fixed demands when using the Frank-Wolfe method (see reference [2]). The SP (shortest path) method solves the traffic assignment problem by applying geographic decomposition to the shortest path subproblem of the Frank-Wolfe algorithm. The basic idea of this method is first, to find all pertinent shortest paths within each region of the network and then, to combine the regional paths so as to obtain the

shortest path over the entire network. A "master" network consisting only of "boundary" nodes is used for the combination step.

As discussed in Section 3, when the SP method is applied to a network model of Washington DC (consisting of 1287 nodes, 3752 links and 151 zones), there is substantial running time or storage savings when compared with the non-decomposed approach.

1.3 GENERALIZED BENDERS GEOGRAPHIC DECOMPOSITION

Generalized Benders' decomposition is a mathematical technique (described in [12] and [13]) for partitioning complex optimization problems to make them easier to solve. It assumes that the decision variables in the original problem may be divided into two sets, X and Y, such that if the values in the Y set are held constant, then the resulting problem in the X variables is relatively easy to solve. The solution procedure consists of solving an X variable problem (called a "subproblem") and then using this solution to adjust the Y variables (a process which we call the "master problem"); the subproblem-master problem iteration is repeated until a satisfactory answer is obtained.

The BD (Benders' decomposition) method solves the traffic assignment problem by applying generalized Benders' decomposition. In this application we take the X variables to be the link flows within the geographic regions and the Y variables to be the flows across the region boundaries.

The results obtained so far for the BD method are not as encouraging as those for the SP method, and it does not seem to be likely that the BD method would be very useful in solving practical traffic assignment problems.

1.4 DANTZIG-WOLFE GEOGRAPHIC DECOMPOSITION

The Dantzig-Wolfe decomposition technique [18] is a method for solving large linear programs which are comprised of a number of subproblems

linked by a relatively small number of interaction constraints. The interaction constraints form the basis of a "master problem". The solution procedure consists of solving the master problem and the subproblems alternately with a guaranteed convergence to the optimal solution. The master problem provides "prices" which are used in the objective functions of the subproblems to produce desirable (improving) subproblem solutions which become proposals (candidates) for the master problem on the next iteration. In the Dantzig-Wolfe decomposition procedure as applied to the traffic assignment problem, the subproblems correspond to the flows in the geographic subnetworks; and the interaction relationships (which form the basis of the master problem) are the flows across regional boundaries, i.e., flows from one subnetwork to another.

The formulation for this approach and an outline of the algorithm are presented in Section 5. The method has not been implemented. However, it is felt that it is similar in some respects to the BD method and would probably provide comparable solution times.

1.5 POTENTIAL APPLICATIONS AND EXTENSIONS

This report has been limited to the study of the application of geographic decomposition to the solution of highway traffic assignment problems with fixed demand. However, geographic decomposition has a wider application because the traffic assignment problem with fixed demands is a basic subproblem in the solution of several other transportation network problems.

The network design problem involves determining which links in a network to improve in order to minimize total travel time subject to a budget constraint. For the case in which there are convex investment costs, continuous investment decision variables and system optimal traffic assignment, it has been shown

in [1] that the network design problem can be solved as a series of traffic assignment problems. Thus, a method that will geographically decompose the traffic assignment problem will also geographically decompose the network design problem.

Another important problem is to determine how best to carry out a series of major improvements for a transportation network. Since it will probably be desirable to make some improvements earlier than others, the so-called investment staging problem seeks the optimum sequence in which to carry out the individual link improvements at given stages over a fixed time horizon. For the case in which there are convex investment costs, continuous investment decision variables and system optimal traffic assignment, it has also been shown in [1] that the investment staging problem can be solved as a series of network design problems, each of which is solved as a series of traffic assignment problems. Thus, the investment staging problem can also be geographically decomposed using the methods developed in this report.

In the traffic assignment problem with elastic demands, the number of trips between a particular origin and destination depends upon the cost of travel between that pair of zones (or nodes). This relationship is specified by a demand function. The problem is to determine the user equilibrium traffic assignment subject to those elastic demand functions. In [1] it is shown how this problem may be converted into a traffic assignment problem with fixed demands. Thus, the traffic assignment problem with elastic demands can also be geographically decomposed using the methods developed in this report.

Geographic decomposition may be used in a number of new applications as well. By treating each transportation mode in an urban highway network as a single geographic region, for example, it may be possible to solve the

so-called modal split problem in a new and possibly more efficient way. In a rail application a single geographic region could consist of the tracks owned by a single rail line; while in a freight routing problem a single region could consist of the routes serviced by one of several trucking firms. All of these areas would involve additional work in problem formulation, of course, but the same solution methods described here might be applicable.

2. DESCRIPTION OF PROBLEM

2.1 INTRODUCTION

In Section 2.2 we present a mathematical formulation of the traffic assignment problem with fixed demands. The two computer codes described in this report are both designed to solve this problem with a system optimal objective. Both codes use the BPR (Bureau of Public Roads) travel cost function form $T_j(\cdot)$, commonly used by the Federal Highway Administration (FHWA)

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

where

f_j = total flow on link j,

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).

Two approaches for partitioning a network are described in subsection 2.3; one approach is used for the SP (shortest path) method and the other for the BD (Benders decomposition) method. In Section 2.4 we describe the sample test problems used in this study.

2.2 TRAFFIC ASSIGNMENT PROBLEM FORMULATION

The system optimal traffic assignment problem can be written using the node-arc* notation of Nguyen [3] as:

$$\text{Minimize } Z = \sum_{j \in A} T_j(f_j) = \sum_{j \in A} f_j C_j(f_j) \tag{2}$$

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

subject to:

$$\sum_{j \in W_i} f_j^r - \sum_{j \in V_i} f_j^r = h_i^r \quad (i \in N;\ r = 1, \ldots, R) \tag{3}$$

$$f_j = \sum_{r=1}^{r=R} f_j^r \quad (j \in A) \tag{4}$$

$$f_j^r \geq 0 \quad (j \in A;\ r=1, \ldots, R) \tag{5}$$

where

A = set of links in the network

$T_j(f_j)$ = increasing link travel cost function, where f_j is the total flow on link j (See equation (1) for the form of $T_j(f_j)$ used in this study.)

$C_j(f_j)$ = unit travel cost on link j as a function of total flow on link

f_j^r = flow on link j for origin r

$$h_i^r = \begin{cases} -O_{ri} & \text{if } i \text{ is a destination node} \\ \sum_j O_{rj} & \text{if } i = r \\ 0 & \text{otherwise} \end{cases}$$

N = set of nodes in the network

O_{ij} = number of trips between all origin-destination pairs ij

R = number of origin nodes in the network

V_i = set of links terminating at node i

W_i = set of links originating at node i.

This problem is a minimum convex cost multi-commodity network flow problem without coupling constraints. The travel cost functions are convex and differentiable and therefore amenable to a natural decomposition by commodities, in this case origin nodes. The resulting solution scheme due to Murchland [4] and Nguyen [3] is given in [1].

2.3 GEOGRAPHIC DECOMPOSITION

2.3.1 Partition by Nodes

Geographic decomposition is based on the observation that very large networks are often only loosely connected; in other words, if a small set of links are deleted from such a network, it will decompose into a series of disjoint subnetworks. For example, if four links are deleted from the network in Figure 2.1,

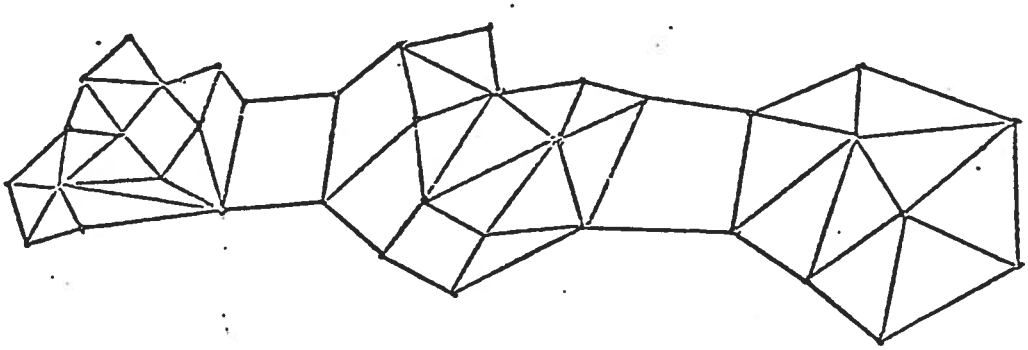


FIGURE 2.1 -- Non-Decomposed Network

we could decompose the network into a series of unconnected subnetworks, as in Figure 2.2.

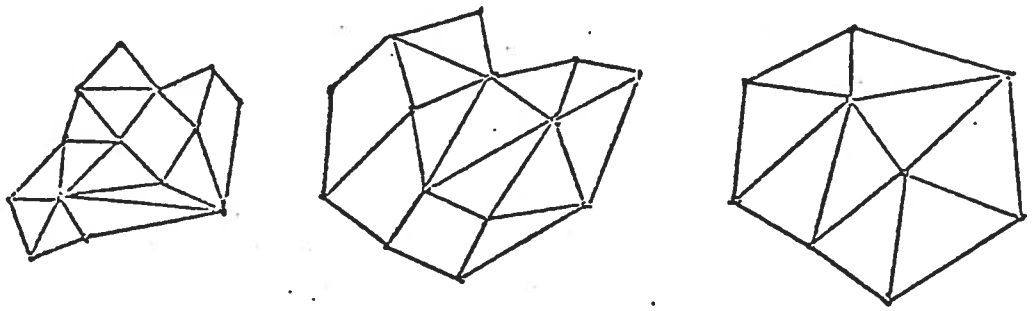


FIGURE 2.2 -- Decomposition of Network After Partition by Nodes

We assume that a network has arc set A and node set N. Let S be the set of deleted links as in the above example. With the removal of the links in S, the network (N,A) can be decomposed into a series of disjoint subnetworks (N_ℓ, A_ℓ) $\ell = 1, \ldots, L$, where

$$N = N_1 \cup N_2 \cup \ldots \cup N_L$$

$$A = A_1 \cup A_2 \cup \ldots \cup A_L \cup S.$$

This may be thought of as geographic decomposition by link deletion and is equivalent to partitioning by nodes. Each link of S has the property that it connects two nodes of different regions. This type of decomposition is used for the SP algorithm discussed in Section 3.

Our algorithm also requires a general "connected" property that a directed path exists from origin to destination for each origin/destination pair which has a non-zero demand.

2.3.2 Partition by Arcs

Another approach to geographic decomposition is to partition by arcs, so that the arcs in A are assigned to different geographic areas. We assume

that there are L such areas and that A_ℓ is the set of arcs in area ℓ, where $A = A_1 \cup A_2 \cup \ldots A_L$ and $A_\ell \cap A_k = \phi$ for $\ell \neq k$. We adopt the following conventions:

The nodes in the set N_ℓ are classified into two types. If all of the arcs incident* to node i belong to the same region ℓ, then node i is placed in the set N^*_ℓ. On the other hand, if arcs from more than one region are incident to node i, then node i is replicated as follows.

First we consider the case when all arcs incident* to some node i belong to one or other of two regions ℓ and ℓ' as shown in Figure 2.3(a). The original node i is split into nodes i and node i' as illustrated in figure 2.3(b). All arcs in A_ℓ that were incident to the original node are still incident to node i. All arcs in $A_{\ell'}$ that were incident to node i are now incident to node i'. The node i is placed in the set N^{**}_ℓ and the node i' in the set $N^{**}_{\ell'}$. If the original node was an origin or destination node, then node i is still an origin/destination node with the same demand as the original node. Node i' has no external demand. With the creation of i', we also create two artificial arcs p and p'. The two artificial arcs are placed in the arc set S.

Next we consider the extension to the case when the arcs incident to some node i belong to more than two regions. An example for three regions is illustrated in Figure 2.4.

a) arc j_1 ($j_1 \in A_\ell$) — i — arc j_2 ($j_2 \in A_{\ell'}$)

b) arc j_1 ($j_1 \in A_\ell$) — i ⇄ (arc p / arc p') ⇄ i' — arc j_2 ($j_2 \in A_{\ell'}$)

Figure 2.3 - Boundary Node of Two Regions

*Arcs which originate or terminate at node i.

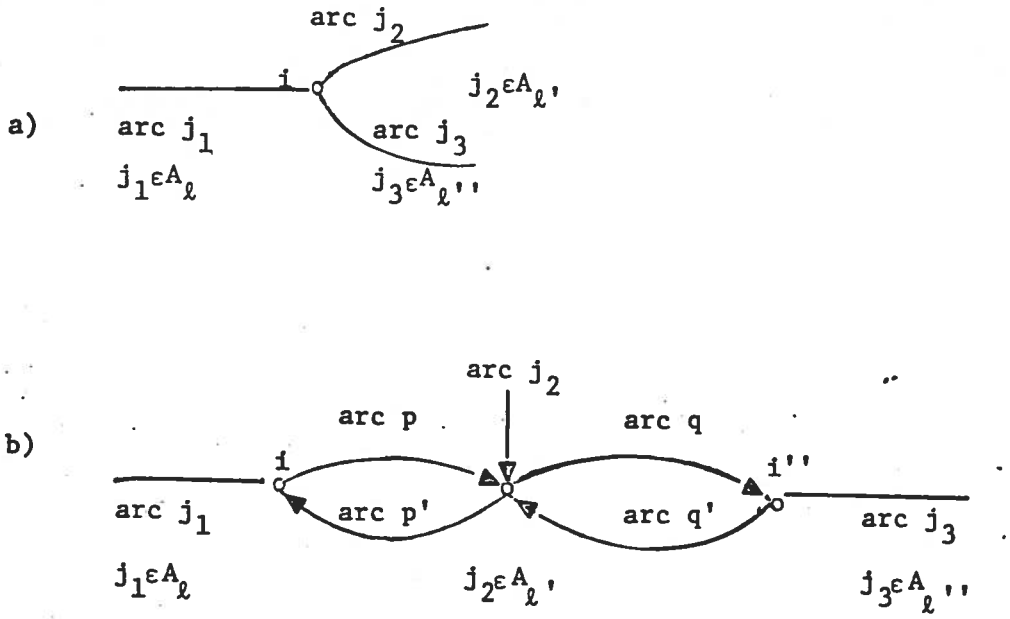


Figure 2.4 - Boundary Node of Three Regions

In this case we create two dummy nodes i' and i''. The node i is placed in N_{ℓ}^{**}, the node i' is placed in N_{ℓ}^{**}, and the node i'' is placed in N_{ℓ}^{**}. If the original node i was an origin/destination node, then node i is still an origin/destination node with the same demand as the original node. Nodes i' and i'' have no external demand. The artificial arcs p, p', q, and q' are placed in the set S. The extension to more than three nodes is straightforward. Let $N_{\ell} = N_{\ell}^{*} \cup N_{\ell}^{**}$. With these conventions we see that geographic decomposition by partitioning of arcs can be made equivalent to decomposition by partitioning of nodes as discussed in Section 2.3.1.

To complete our definitions we now define:

$h_{i'}^{r}$ = 0 for all newly created dummy nodes i', $(r = 1, \ldots, R)$.

W_{i}^{*} = set of all arcs originating at node i that also belong to A. (This excludes all artificial arcs in S.)

W_{i}^{**} = set of all arcs originating at node i that also belong to S.

V_i^* = set of all arcs terminating at node i that also belong to A. (This excludes all artificial arcs in S)

V_i^{**} = set of all arcs terminating at node i that also belong to S.

This approach to partitioning is used in the BD method described in Section 4.

We also require that the network and associated partition have a "strongly connected" property in the following sense.

Define node subsets N_ℓ^r as follows. A node i is a member of set N_ℓ^r if it is a member of N_ℓ and is either:

(i) the origin or a destination node for commodity, r i.e., $h_i^r \neq 0$

or (ii) a node at which an arc in the set S either originates or terminates.

If for each pair of nodes i and j in the set N_ℓ^r there exists a directed chain from i to j and a directed chain from j to i, and such chains exist for all node sets N_ℓ^r, r = 1, ..., R, and ℓ= 1, 2, ..., L, the network (N,A) and partition will be called strongly connected. (A directed chain is a set of links that connect i and j which are all oriented in the same direction.)

In general, one would expect a transportation problem to be strongly connected, since the nodes in the node sets N_ℓ^r are usually zones (centers for demand) or connector nodes on major arteries.

2.4 SAMPLE PROBLEMS

This report includes numerical results from applying our decomposition techniques to two different sample problems: a hypothetical network model with 81 nodes; and an actual model of Washington DC with 1287 nodes. The

basic characteristics of these problems are summarized in Table 2.1. The region sizes given are for the node partition method of Section 2.3.1. The 81-node hypothetical model has a structure which decomposes very readily into three regions. The Washington DC model was developed in 1968 by the Metropolitan Washington Council of Governments, and for the purposes of our test runs we decomposed this model into four regions.

TABLE 2.1 - SAMPLE PROBLEM CHARACTERISTICS

PROBLEM	HYPOTHETICAL			WASHINGTON DC			
Total Nodes	81			1287			
Total Links	204			3752			
Total Zones	31			151			
Regions	3			4			
Boundary Nodes	10			63			
Region	1	2	3	1	2	3	4
Nodes	22	21	38	427	397	221	242
Links	50	50	94	1240	1098	628	718
Zones	8	7	6	52	48	32	19

Because of the relatively poor performance of the BD code on the smaller hypothetical problem, we decided not to test this code on the much larger Washington DC model; thus, only results for the hypothetical model are given in Section 4. However, the SP code was tested on both the hypothetical and Washington DC models, and these results are given in Section 3.

3. SHORTEST PATH DECOMPOSITION

3.1 BACKGROUND

The traffic assignment problem described in Section 2.2 is customarily solved by an application of the Frank-Wolfe algorithm. This is discussed in some detail in references [1] and [2]; briefly, the technique involves the generation of a series of feasible flow patterns each of which satisfies all of the conservation of flow constraints at each node. The patterns are obtained by routing all of the trip demands for each origin-destination pair via the links of the network which lie on the shortest path between the pair of nodes. The costs for the arcs are adjusted as the algorithm proceeds and so the shortest paths must be redetermined at each iteration; 70-80 percent of the computer time is typically required for finding these paths.

Because of the many applications for the shortest path problem (besides the traffic assignment problem), considerable attention has been devoted to its solution. Much of this attention has been focused on ways of adapting a few basic algorithms (e.g., Dijkstra [5], Floyd [6], Pape [21]) to the special structure of specific classes of problems. Some examples of special structures which some workers have chosen to exploit are:

(1) Link sparsity in the network in which the shortest path is to be found. This occurs when each node is connected to only a few (for example, four or five) other nodes.

(2) Geographic locality. This is somewhat stronger than (1) in that nodes are connected only to other nodes which are "nearby" in some X-Y grid.

(3) Link lenths which are non-negative. (This property is required for the Dijkstra algorithm).

(4) Link lengths which are restricted to a small range of positive values or to a range of integer values. (This property is required for the Dial algorithm [17]).

(5) A network which may be divided into two or more regions whose only connection is via a relatively small set of nodes.

(6) A network with property (5) whose regions are connected in a linear or corridor fashion, i.e., each region communicates only with its neighbors on either side.

The types of networks with which we are dealing in this report will certainly have properties (1) - (3) above; furthermore, property (4) may probably be assumed in most cases (as it is in the UTPS code UROAD). For the present section we will be considering transportation networks which have property (5) but not necessarily property (6).

Previous efforts at structural decomposition of the shortest path problem have concentrated on networks with properties (5) and (6) but have generally assumed the network to be dense (i.e., each node is connected to most of the others) within a given region. (See for example, Hu [8], Hu and Torres [9], or Glover, Klingman and Napier [10].) Since transportation networks will almost always have properties (1) - (3) within each region it may be advantageous to use the resulting special problem structure in the solution algorithm. The shortest path decomposition algorithm described next is a new approach that can take advantage of the structure encountered in highway networks (properties (1) - (3)) and it does not require the corridor assumption (property (6)) that is assumed by the previous efforts.

A "boundary node" is a node located in one region that is connected directly to a node located in another region. The shortest path decomposition algorithm described next does assume that the network has property (5), namely that it can be partitioned into two or more regions with only a relatively

small set of boundary nodes. The basic idea of our method is to exploit the fact that a shortest path between an origin node in one region and a destination node in another region must pass through one or more boundary nodes. The algorithm given in the next section has been adapted to solve the shortest path subproblem within a Frank-Wolfe traffic assignment code. This means that explicit shortest paths are not produced; rather, trip demands for each origin-destination pair are directly assigned to the links lying on the shortest path between the pair. However, the paths could be produced explicitly, if needed.

3.2 ALGORITHM

In the following, we assume that the network has been geographically decomposed by assigning each node to one of L regions; this partition by deleting links is described in Section 2.3.1. We further assume that the network is connected in the sense defined in Section 2.3.1. In region ℓ the set of nodes N_ℓ is divided into three subsets: D_ℓ, the origin-destination nodes; B_ℓ, the boundary nodes (i.e., those connected directly by one of the deleted links to a node in some other region); and R_ℓ, the remaining nodes. The computer code requires that these three subsets be disjoint; if they are not disjoint, dummy nodes and links may be introduced as needed to satisfy this condition. A single node is denoted by a lower case letter; e.g., $d_{\ell i}$ is the ith origin/destination node in region ℓ.

The algorithm proceeds by solving several shortest path problems, each of which is restricted to nodes and links in a single region. The resulting distances are called conditional shortest paths, the condition being that the paths are restricted to the particular region. The conditional distance from

node d_i to node b_j is denoted by $d_c(d_i, b_j)$. As the solution proceeds, paths traversing more than a single region are found; the shortest of all such paths we call the unconditional shortest path. This unconditional distance is given by $d_u(d_i, b_j)$; note that this may be identical to the conditional distance for nodes in the same region.

The algorithm consists of four main steps. The first step is to find the conditional shortest distances from each boundary node to the other nodes in the same region. Using the results from this first step, a network consisting only of boundary nodes is constructed. The second step is to find the unconditional shortest distances between each pair of boundary nodes. The third step is to find the sequence of boundary nodes that defines the shortest path between each pair of origin and destination nodes. This is done by first computing the conditional shortest distances between the origin node and each boundary node in the origin region, computing the conditional shortest distances between each boundary node in the destination region and the destination node; and combining this information with the unconditional shortest distances between all pairs of boundary nodes. The fourth and final step determines the unconditional shortest path between each pair of boundary nodes identified in the third step or between the final boundary node and destination. Next, we will describe this algorithm in detail:

> Step 1. Boundary Node Conditional Distances. Find the conditional shortest path from each boundary node to the other nodes in the same region. Any convenient method may be used for this step; the computer code (see Section 3.3) uses Pape's algorithm [21] . The resulting distances for region ℓ are displayed in the tableau in Figure 3.1.

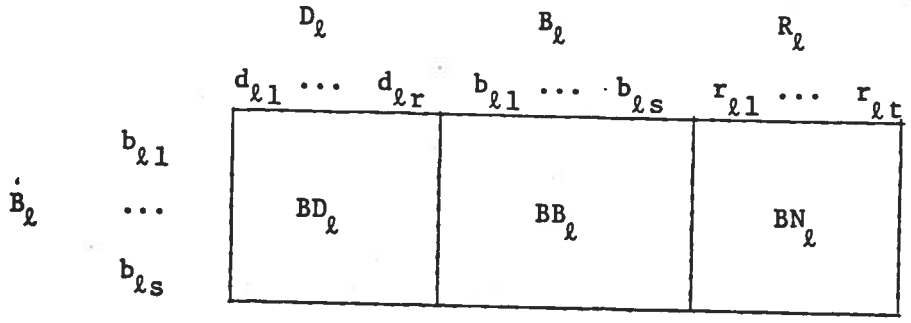


Figure 3.1 - Shortest Path Tableau

The entry in row $b_{\ell i}$ and column j is $d_c(b_{\ell i}, j)$. The submatrices BD_ℓ, BE_ℓ must be saved; BN_ℓ will be used in Step 4 but it need not be retained if there is not sufficient room (there generally is not).

Step 2. Boundary Node Unconditional Distances. A network whose nodes are just the boundary nodes of the original network is used for this step. Links are introduced joining each pair of boundary nodes in the same region; the length of each link is taken from the BB_ℓ matrix found in Step 1. The remaining links in this network are the links originally deleted to obtain the partition; we refer to them as boundary links. The matrix of Figure 3.2 results.

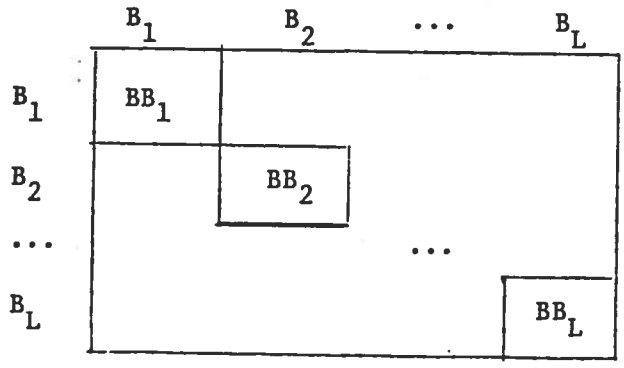


Figure 3.2 - Boundary Node Distance Matrix

The boundary links appear in the off-diagonal part. A shortest path problem is now solved for every node in the new network. Once again any method may be used but because of the density of arcs the basic Floyd algorithm [6] is probably best. The shortest distance between each pair of boundary nodes and also the shortest path itself (i.e., the nodes on the path) are both saved for later steps.

Step 3. Fan Out. In this step the trips originating at each origin are assigned to the proper arcs in the origin's home region. Trips which remain entirely within the home region are routed directly to their destination nodes while those which leave the region are routed to the proper boundary node. The steps below (3a-3e) are repeated for each origin region ℓ and for each origin node $d_{\ell i} \in D_\ell$:

a. Solve a shortest path problem to find $d_c(d_{\ell i}, n_{\ell j})$ for each node $n_{\ell i} \in N_\ell$. Then repeat the following steps (3b-3d) for each destination region k, k = 1, ..., L:

b. Find the unconditional shortest path from the origin node $d_{\ell i}$ to each boundary node $b_{km} \in B_k$ in the destination region using

$$d_u(d_{\ell i}, b_{km}) = \min_n [d_c(d_{\ell i}, b_{\ell n}) + d_u(b_{\ell n}, b_{km})] ,$$

where the first term was found in Step 3a and the second in Step 2. Both the unconditional distance and the value of n are saved; trips which are destined for boundary node b_{km} will be routed via the boundary node $b_{\ell n}$ in origin $d_{\ell i}$'s home region.

c. Find the unconditional shortest path from the origin node $d_{\ell i}$ to each destination node $d_{kj} \in D_k$ using

$$d_u(d_{\ell i}, d_{kj}) = \min_m [d_u(d_{\ell i}, b_{km}) + d_c(b_{km}, d_{kj})] ,$$

where the first term was found in Step 3b and the second is from the BD_k matrix found in Step 1. When $\ell = k$ the direct path found in Step 3a may be shorter; if so, it is used. If not, then the shortest path from $d_{\ell i}$ to d_{kj} is the path from $d_{\ell i}$ to $b_{\ell n}$ (n was recorded in Step 3b and the path was found in Step 3a), then from $b_{\ell n}$ to b_{km} (the path was found in Step 2) and finally from b_{km} to d_{kj} (the path was found in Step 1, but it may have been discarded).

d. Because the requirement for shortest paths in the Frank-Wolfe traffic assignment algorithm is to find the proper routing of trip demands, we combine this feature directly with the geographically decomposed shortest path determination in order to minimize computer system overhead. To do this we augment the trip demand table by adding all of the boundary nodes as potential origins and/or destinations; all boundary trip demands are set initially to zero.

If $k = \ell$ and the direct path from Step 3a is the unconditional shortest path, the basic trip table will not be modified. Otherwise, the following four steps are carried out [O(i,j) refers to the number of trips between origin i and destination j]:

(1) Increase $O(d_{\ell i}, b_{\ell n})$ by $O(d_{\ell i}, d_{kj})$; this routes the trips on the first leg of the shortest path, i.e., to the home region boundary node.

(2) Increase $O(b_{\ell n}, b_{km})$ by $O(d_{\ell i}, d_{kj})$; this routes the trips to the proper boundary node in the final region. Note that there may be further re-routing needed along the path found in Step 2; this is deferred until Step 4a.

(3) Increase $O(b_{km}, d_{kj})$ by $O(d_{\ell i}, d_{kj})$; this routes the trips to the destination node.

e. Using the shortest paths found in Step 3a and the trip demands between $d_{\ell i}$ and $n_{\ell j} \varepsilon (D_\ell \cup B_\ell)$ found in Step 3d, increase the flow on each link in region ℓ by adding the appropriate demands.

Step 4. Fan In. In this step the demands temporarily assigned to boundary nodes are routed to the proper destination.

a. First re-route all trip demands according to the shortest paths found in Step 2. This may be done by manipulating the trip table; for example, if the shortest path in Step 2 were $b_i \rightarrow b_j \rightarrow b_k$ the following steps would be performed:

(1) Increase $O(b_i, b_j)$ and $O(b_j, b_k)$ by $O(b_i, b_k)$.

(2) Set $O(b_i, b_k)$ to zero.

b. For each region ℓ and for each $b_{\ell i} \varepsilon B_\ell$ find the conditional shortest path from $b_{\ell i}$ to each $n_{\ell j} \varepsilon N_\ell$; this was first done in Step 1 but must be repeated if the data was not saved. Route the trips in the trip table from $b_{\ell i}$ to each $d_{\ell j} \varepsilon D_\ell$ and to each $b_{\ell n} \varepsilon B_\ell$ according to these paths. Finally, route the trips from $b_{\ell i}$ to each $b_{kj} \varepsilon B_k$, $k \neq \ell$ via the proper boundary links.

3.3 IMPLEMENTATION

The shortest path algorithm of Section 3.2 has been implemented in a FORTRAN program which is briefly described in this Section; for convenience, the program is referred to as the "SP Program". This program uses the shortest path decomposition approach to solve the shortest path subproblem within a Frank-Wolfe traffic assignment algorithm. The material here is not intended to serve as documentation for the prospective user of the code nor does it provide sufficient detail to enable one to reconstruct the program; the aim, rather, is to give an overview of the way in which the algorithm has been set up.

The SP code is quite similar to the CATNAP code described in [2]; it uses the same input formats, many of the same data structures, and also the basic Frank-Wolfe algorithm. Most of the special features of CATNAP have been omitted from SP, however; SP solves only the traffic assignment problem with a system optimal objective function and has no network design or investment staging capability. These features could be added to SP without much difficulty though, because of its great algorithmic similarity to CATNAP.

The SP program consists of 27 modules in three different groups: input, traffic assignment and data management. The first two groups function much as do their counterparts in CATNAP; the third group, however, is unique to SP. The program incorporates a dynamic memory allocation method which automatically sets aside just enough main memory to store the problem data within a user-supplied array; if there is insufficient memory, the data management routines set up external storage for the data on tape or disk. This is described more fully below.

The basic network topography data is stored much as it is in CATNAP: the links are sorted by origin node within each region and a pointer array gives for each node the index of the first link originating there. The practical capacity, free flow travel time and destination node index are stored for each link as well. The major difference between SP and CATNAP is that the node indices within a given region in SP are assigned independently of the other regions; thus, a reference array is used to give for each node its global index (the index used in the problem input data). Using half word integers for storing indices, the total memory requirement for region ℓ is then $N_\ell + 2\frac{1}{2} \cdot A_\ell$.

Core storage must also be allocated for trip tables and for the current and trial flow values on each link. Since the topography, trip table and flow values for only a single region need be available at any given time, the values for the other regions may be stored outside of main memory and brought in only when required. A set of three buffers is thus allocated as follows:

(1) Flow values (existing flows, trial flows and marginal costs); the size is $\max_\ell \ 3 \cdot A_\ell$ words.

(2) Network topography, as described above; $\max_\ell \ N_\ell + 2\frac{1}{2} \cdot A_\ell$ words.

(3) Trip table for a single origin; $\frac{1}{2} \cdot Z$ words. (Z is the total number of origins and destinations.)

If additional memory is available it is assigned for extra buffers in the order given above; otherwise, the remaining values are "paged out" to an external data set, one buffer at a time. If sufficient memory for one buffer of each type is not supplied, program execution is halted.

In addition to buffered values, memory must be allocated for arrays which are core-resident at all times. These arrays include:

(1) The topography and flow values for the links and nodes in the boundary area, i.e., between regions.

(2) The BB and BD matrices described in Section 3.2.

(3) The augmented trip table needed to re-route trip demands via boundary nodes.

(4) Various work vectors (e.g., to find the conditional shortest path solutions) and status vectors (e.g., to give the size of the network in each region).

Once memory allocation has been performed and problem data read in the SP code solves a series of shortest path problems using the algorithm of Section 3.2. If external storage must be used the code will read the trip table once and the topography and flow data three times per Frank-Wolfe iteration; this minimizes both processor time and input-output charges.

The amount of main memory needed by the SP program depends critically on the number of boundary nodes and the number of regions as well as the size of the problem in each region. The total amount of non-buffer array storage is given approximately by

$$5 \cdot NB^2 + 7 \cdot NB + 12 \cdot NR + 1\tfrac{1}{2} \cdot NB \cdot NZR + 3 \cdot NNR \text{ words,}$$

where NB is the number of boundary nodes, NR the number of regions, NZR the maximum number of zones in a single region and NNR the maximum number of nodes in a region. In addition to this, core must be added for at least one of each type of buffer.

The minimum amount of array storage required for the 81-node hypothetical problem discussed in Section 2.4 is 1191 words (compared to 2087 words for CATNAP). Table 3.1 gives approximate minimum array sizes for a series of traffic assignment problems of increasing size; it is assumed that the number of links is about three times the number of nodes in each region and that the regions are approximately of equal size.

TABLE 3.1 - ESTIMATED MINIMUM ARRAY STORAGE
FOR VARIOUS SIZED NETWORKS

Nodes	Regions	Boundary Nodes	Minimum Array Storage SP	CATNAP
2000	2	40	22,600	39,600
3000	3	60	28,000	59,400
5000	5	100	43,000	98,900
10,000	10	200	107,500	197,700

3.4 COMPUTATIONAL EXPERIENCE AND RESULTS

The SP code has been tested on a variety of traffic assignment network problems. The answers given by SP at each Frank-Wolfe iteration should be identical to those from CATNAP on the same problem, because the two codes use the same basic Frank-Wolfe traffic assignment algorithm. In practice, however, the results from these two codes are slightly different. Because shortest paths are not always unique in real networks, the two programs may choose different initial solutions, so that these codes may then approach the optimum solution by different paths. When a small random value is added to each of the free-flow travel times (thus making all shortest paths unique), the two codes produce identical results.

As discussed in Section 1, there are two reasons for studying geographic decomposition: Firstly, some networks may be too large to be handled as a single entity in computer processing operations, and geographic decomposition may then allow the problem to be solved piecemeal within the size limits of the computer without loss of accuracy. This reason will be examined in Section 3.4.1 using the 81-node hypothetical problem, where computational results will be given for several SP runs which use varying amounts of core storage but

with the same regional decomposition (3 regions). Secondly, it may be possible to use geographic decomposition to reduce the overall running time, as it may be more efficient to solve a series of smaller problems rather than solving the original size problem. This second reason will be examined in Section 3.4.2 using the 1287-node Washington DC network, where computational results will be given for several SP runs which use varying regional decompositions (2, 3 or 4 regions) but without any external storage.

3.4.1 Hypothetical Network

First, we will discuss the computational results from running the SP code on the 81-node sample problem of Section 2.4. As indicated previously, this problem requires a minimum of 1191 words of array storage (vs. 2037 words for CATNAP), although this problem could be managed entirely in-core with 2768 words of storage.

Table 3.2 gives the performance of the SP and CATNAP codes after 26 Frank-Wolfe iterations. Note first that the in-core version of SP (run number 1 in Table 3.2) requires about 35 percent more array storage than CATNAP, but costs about the same to solve. The slight time difference may be accounted for by the fact that SP code must read the input data five different times in order to set up all the buffers. Thus from an algorithm point of view, the SP program is about as efficient as CATNAP for this problem.

Because the topography, trip tables and flow values for only a single region need be available at a time for the SP code, the values for the other regions may be stored outside of core memory and brought in only when required. Several different storage allocation schemes can be devised, depending upon which arrays are retained in-core and which arrays are chosen to be stored externally (see run numbers 2, 3 and 4 in Table 3.2). Note that when external

TABLE 3.2 PERFORMANCE OF TRAFFIC ASSIGNMENT CODES ON THE 81 NODE HYPOTHETICAL PROBLEM: BASED UPON 26 FRANK-WOLFE ITERATIONS

Run Number	Program	Number of Regions	Actual Array Storage (words)	CPU Time[a] (IBM 370) (sec.)	Disk I/O	Processing Cost[b] (in dollars)	Minimum Array Storage[c] (words)
1	SP	3	2768	4.74	40	.92	1191
2	SP	3	2075	13.23	401	2.78	1191
3	SP	3	1439	13.90	499	2.98	1191
4	SP	3	1191	29.00[d]	738[d]	10.00	1191
5	CATNAP	1	2087	4.41	13	.84	2087

a Includes execution time only, not compilation time; both CATNAP and SP used the Bellman [7] shortest path algorithm.

b Based upon the "evening" computer rates at the Stanford Computation Center, which are: $11.34 CPU cost per minute; $.70 per 1000 operations for Disk I/O.

c Because the topography, trip table and flow values for only a single region need be available at a time for the SP code, the values for the other regions may be stored outside of core memory and brought in only when required.

d Run did not finish.

storage is being used, then the additional CPU and disk changers cause the solution cost to rapidly increase. This effect is not due to algorithmic differences, because all four of the SP runs carried out the same sequence of operations and gave identical results. Rather, this increased cost results from computer system overhead for input and output.

Our conclusions for this problem are the following: The in-core version of SP costs about the same as CATNAP to solve; while SP does allow the traffic assignment problem to be solved with reduced amounts of core array storage, the solution costs rapidly increase as the available amount of core storage decreases.

3.4.2 Washington DC Network

The Washington DC network model includes 1287 nodes and 3752 arcs and it is specified by three data sets:

a) Link data which includes the to and from nodes, the link's length, the free flow travel time on the link, the number of lanes, whether or not a reverse link can be built, and information needed to determine link capacity.*

b) Grid data which consists of the horizontal and vertical coordinates of each node in the network.

c) Trip demand data which is the number of trips required between each origin and destination pair.

By plotting the grid and link data, it was possible to decompose the network into four regions (see Figure 3.3).

a) Virginia, by cutting along the Potomac River.

---

* The data specifying the Washington DC network was obtained from the Federal Highway Administration (FHWA), and it was "cleaned up" by deleting three obviously erroneous links.

b) Northern Maryland suburbs, consisting of Montgomery County and small portions of Howard, Prince George's and Anne Arundel counties; the eastern cut was between Highways I-95 and Maryland 564.

c) Southeastern Mayland, consisting of the remainders of Prince George's and Anne Arundel counties that are south of Maryland Highway 564. This region also includes the portion of Washington DC that is east of the Anacostia River; and

d) Central Washington, consisting of all of Washington DC except that part east of the Anacostia River which is included in region c.

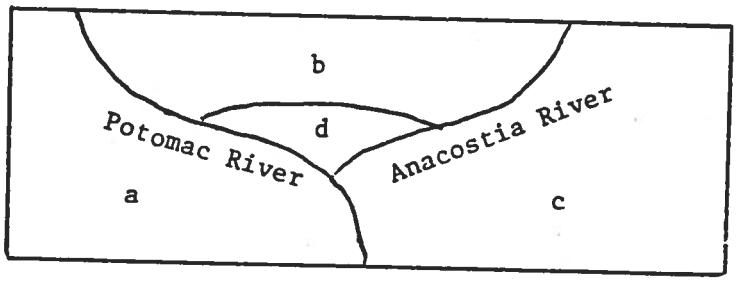


Figure 3.3 Geographic Decomposition of Metropolitan Washington DC

These regions can be recombined, yielding numerous regionalizations which may be studied. We looked at five of these possibilities as described in Table 3.3. The complete regionalization is described by alternative E in Table 3.3.

The results from running CATNAP and the SP codes on the Washington DC network are summarized in Tables 3.4 and 3.5. Consider first Table 3.4 which is based upon 4 Frank-Wolfe iterations. CATNAP required 42432 words of array storage, processing cost of $14.44, and 1.26 minutes of CPU time.

TABLE 3.3 WASHINGTON DC GEOGRAPHIC COMPOSITION ALTERNATIVES

Alternative	Number of Regions	Descriptions of Regions		Boundary	Region 1	Region 2	Region 3	Region 4
A	1	1. Washington DC Area	Nodes		1287			
			Arcs		3752			
			Zones		151			
B	2	1. Virginia 2. Central DC and Northern & S.E. MD	Nodes	18	427	860		
			Arcs	20	1240	2492		
			Zones		52	99		
C	3	1. Virginia 2. Northern MD 3. Central DC & S.E. MD	Nodes	56	427	397	463	
			Arcs	60	1240	1098	1354	
			Zones		52	48	51	
D	3	1. Virginia 2. Northern MD & Central DC 3. S.E. MD	Nodes	43	427	639	221	
			Arcs	46	1240	1838	628	
			Zones		52	67	32	
E	4	1. Virginia 2. Northern MD 3. S.E. MD 4. Central DC	Nodes	63	427	397	221	242
			Arcs	68	1240	1098	628	718
			Zones		52	48	32	19

TABLE 3.4 PERFORMANCE OF TRAFFIC ASSIGNMENT CODES ON THE WASHINGTON DC NETWORK: BASED UPON 4 FRANK-WOLFE ITERATIONS

Run Number	Program	Geographic Decomposition Alternative[a]	Number of Regions	Actual Array Storage[b] (words)	CPU Time[c] (IBM 370) (min.)	Disk I/O	Processing Cost[d] (in dollars)	Minimum Array Storage[e] (words)
1	CATNAP	A	1	42432	1.26	215	14.44	42432
2	SP	B	2	47301	1.18	502	13.73	20754
3	SP	C	3	48542	1.12	529	13.07	20740
4	SP	D	3	54176	1.09	526	12.73	20698
5	SP	E	4	54597	1.09	554	12.75	20873

[a] Refer to the definitions given in Table 3.3

[b] This is the actual core array storage used in this run, which resulted in the indicated CPU time, Disk I/O, and processing costs, and it corresponds to having no external storage.

[c] Includes execution time only, not the compilation time; both CATNAP and SP used the Pape [21] shortest path algorithm.

[d] Based upon the "evening" computer rates at the Stanford Computer Center which are: $11.34 CPU cost per minute; $.70 per 1000 operations for Disk I/O.

[e] Because the topography, trip table and flow values for only a single region need be available at a time for the SP code, the values for the other regions may be stored outside of core memory and brought in only when required. The minimum core storage requirements are indicated in this column. If this minimum storage option were used, then the CPU time, Disk I/O and processing costs would be higher than those listed, although the sequence of calculations would be the same.

TABLE 3.5 PERFORMANCE OF TRAFFIC ASSIGNMENT CODES ON THE WASHINGTON DC NETWORK: BASED UPON 25 FRANK-WOLFE ITERATIONS

Run Number	Program	Geographic Decomposition Alternative[a]	Number of Regions	Actual Array Storage[b] (words)	CPU Time[c] (IBM 370) (min.)	Disk I/O	Processing Cost[d] (in dollars)	Minimum Array Storage[e] (words)
1	CATNAP	A	1	42432	4.76	216	54.13	42432
2	SP	E	4	54597	2.86	554	32.82	20873

Refer to Table 3.4 for footnotes

SP runs were made for the four different geographic decomposition alternatives that were described in Table 3.3. Each of these SP runs used the in-core version of the code (i.e., there was no external storage) and each of these runs resulted in smaller CPU times and processing costs than did CATNAP. Note that the running times for the SP code are influenced by both the number of boundary nodes and the number of regions. For example, both alternatives D and E (in Table 3.3) required 1.09 minutes of CPU time; thus the added efficiency gained from decomposing the network into an additional region (from 3 to 4) was exactly balanced by the loss of efficiency due to the increase in boundary nodes (from 43 to 63). Note that alternative D resulted in the lowest processing cost, which was $12.73 and is about 12 percent less than the cost for CATNAP. Table 3.4 also gives the minimum array storage requirements for each of the decomposition alternatives. Note that alternative D also had the smallest array storage requirements, which was 20698 words and is about 50 percent less than the storage used for CATNAP. However, as we have seen in Section 3.4.1, the solution costs rapidly increase as the available amount of core storage decreases; thus, if this minimum storage were used, the CPU time, Disk I/O, and processing cost would be much larger than the in-core values listed in Table 3.4.

When comparing the SP and CATNAP codes, it is important to realize that SP must do some initial work setting up buffers for each region prior to initiating the Frank-Wolfe algorithm. Thus, the total CPU time for an SP run includes an initialization component that is not needed by CATNAP. This implies that the comparison of running times between SP and CATNAP will become more favorable for SP as the number of Frank-Wolfe iterations increase. This is illustrated in Table 3.5, in which SP was compared with

CATNAP after 25 Frank-Wolfe interations and SP was exercised in decomposition alternative E (which has 4 regions), resulting in a running time that is about 40 percent less than the running time for CATNAP. Note that in the corresponding comparison given in Table 3.4 for 4 Frank-Wolfe iterations, the running time for SP was only about 13 percent less than the time for CATNAP.

Thus, for the Washington DC problem, our conclusion is that the SP code results in a substantial decrease in running time and processing costs when compared with CATNAP.

4. GEOGRAPHIC DECOMPOSITION USING THE GENERALIZED BENDERS ALGORITHM

4.1 INTRODUCTION

This section presents a method, based on generalized Benders Decomposition [12], that provides a geographic separation of the basic traffic assignment problem into smaller, more manageable subproblems. We will describe the method in connection with the system optimal traffic assignment problem with fixed demands, and use the partitioning by arcs version of geographic decomposition described in Section 2.3.2.

The Benders Decomposition algorithm [13] was proposed for the solution of certain mixed-variable mathematical programming problems. The variables are partitioned into two sets, x and y, for example, where the y set variables are to be regarded as complicating variables in the sense that if values are assigned to them, the resulting mathematical program in x variables is considerably more tractable than the original problem. The solution procedure consists of alternately solving a problem in only the x variables (the subproblem) and solving a "relaxed" master problem in the y variables. The master problem has a very large number of constraints and it is relaxed in that all but a few of these constraints are ignored. Each time the subproblem is solved a new constraint is generated and appended to the set of master problem constraints. The algorithm is enhanced if the y variables can be restricted to "feasible" solutions, that is, solutions for which the subproblem is feasible. The procedure is guaranteed to converge to an optimal solution. The method generates upper and lower bounds on the optimal objective value which in the limit equal this value. The problem considered in the Benders paper is a mixed integer programming problem with the y variables as the set of discrete variables. The subproblem is a linear program in the x variables and the master problem is an all-integer program.

Geoffrion [12] generalized the decomposition procedure of Benders to allow a broader class of problems to be considered, making use of non-linear convex duality theory. The subproblems are not restricted to be linear programs. This generalization is summarized in Appendix 1. (See also Florian and Nguyen [14]).

Consider the following formulation of the traffic assignment problem. We will refer to this as problem P1.

$$\text{Min} \quad \sum_{\ell} \sum_{j \in A_\ell} T_j(f_j) \tag{6}$$

subject to

$$\sum_{j \in W_i^*} f_j^r - \sum_{j \in V_i^*} f_j^r = h_i^r \quad (i \in N_\ell^*;\ \ell = 1, 2, \ldots L;\ r = 1, 2, \ldots, R) \tag{7}$$

$$\sum_{j \in W_i^*} f_j^r - \sum_{\substack{j \in V_i^* \\ i}} f_j^r = h_i^r - \sum_{j \in W_i^{**}} g_j^r + \sum_{\substack{j \in V_i^{**} \\ i}} g_j^r \tag{8}$$

$$(i \in N_\ell^{**};\ \ell = 1, 2, \ldots, L;\ r = 1, 2, \ldots, R)$$

$$f_j = \sum_{r=1}^{R} f_j^r \qquad (j \in A) \tag{9}$$

$$f_j^r \geq 0 \qquad (j \in A) \tag{10}$$

$$g_j^r \geq 0 \qquad (j \in S) \tag{11}$$

where g_j^r is the flow in directed arc j of commodity r and where $T_j(f_j)$,

f_j^r, h_i^r, R, A, A_ℓ, W_i^*, W_i^{**}, V_i^*, V_i^{**}, N_ℓ^*, N_ℓ^{**} and S are as defined in Sections 2.2 and 2.3.2.

We will assume that the network and associated partition are strongly connected in the sense described in Section 2.3.2.

The problem P1 is a convex cost multi-commodity flow problem where the commodities are distinguished as flows which originate at the same zone. Practical problems could be very large, for example 5000 arcs, 500 origins and 2,500,000 decision variables.

In Dantzig, Maier, and Lansdowne [1] two approaches are described for the solution of this problem (P1) using the Benders algorithm with geographic decomposition. These two approaches are summarized very briefly and superfically as follows:

Approach 1: Partition the variables of problem P1 into the two sets (f_j^r, f_j) and (g_j^r) and apply the generalized Benders algorithm as outlined in Appendix 1. The subproblem with variables (f_j^r, f_j) decomposes into a series of disjoint subnetwork problems related to the networks (N_ℓ, A_ℓ) $\ell = 1, 2, \ldots, L$, each of which is a traffic assignment problem. If we define a commodity as being all flows from an original origin (or to an original destination) each subnetwork problem is a minimum convex cost multicommodity network flow problem with commodity interactions only through the convex differentiable objective functions. The natural decomposition by commodity used by Murchland [4] may therefore be used in its solution.

The master problem is a linear program in the (g_j^r) variables. It is possible to ensure feasibility in the (g_j^r) variables in the sense defined earlier by attaching the following additional constraints to the master problem,

and making the assumption that the "strongly connected" property holds for the network.

$$\sum_{i \in N_\ell} \left[h_i^r - \left(\sum_{j \in W_i^{**}} g_j^r - \sum_{j \in V_i^{**}} g_j^r \right) \right] = 0 \qquad (12)$$

$$(r = 1, 2, \ldots, R; \quad \ell = 1, 2, \ldots, L)$$

To initialize the algorithm we need a feasible set of values for the (g_j^r) variables. Then we alternate between the subproblem and the master problem generating a new master constraint (a cut) each time the subproblem is solved. Upper and lower bounds on the optimal objective value are maintained at each step and the procedure terminates when these bounds are sufficiently close.

Approach 2: The second version of the geographic decomposition procedure is obtained by first applying the natural decomposition by commodity. This is justified because the only interaction between commodities is in the objective function and the objective function is convex and differentiable. We then apply the Benders algorithm to each such commodity partition successively and iterate over commodities until the solutions are unchanging.

The Benders technique as applied to the problem resulting when values are fixed for all commodities but one is similar to, but simpler than, the previous approach. Again, the Benders subproblem decomposes into L independent problems corresponding to the geographic decomposition, each of which is a minimum convex cost network flow problem. Approach 2 is the algorithm implemented in the BD computer code which is described in the succeeding sections.

Remark 1: We note that a commodity associated with a unique origin in the original network may have multiple origins in a subnetwork.

Remark 2: As a result of the "partitioning by arcs" geographic decomposition approach the objective function of problem P1 has no component from the g_j^r variables. It is a convex differentiable function of the f_j variables only.

Remark 3: One is tempted to solve the master problem in approach 1 by applying the decomposition by commodity procedure to it to obtain a sequence of linear programs each of which relates to one commodity only. However, the objective function in the master problem is convex but not necessarily differentiable and the procedure is not guaranteed to converge to a global optimal solution. The objective of the master is of the form

$$\min_{g} \left[\max_{k} \; L_k(g, u^k) \right]$$

where L_k is the linear constraint generated from the kth solution of the subproblem, u^k is the dual multiplier vector from the kth solution of the subproblem and g is the vector of master variables. The function $\max_k \; L_k(g, u^k)$ is convex and piecewise linear.

4.2 DEVELOPMENT OF THE ALGORITHM

First we define a single commodity problem which we call $P2_r$; this problem results from decomposing problem P1 by commodity. We assume that feasible flows f_j^p are known ($j \in A$, $p \neq r$).

Problem $P2_r$

$$\min \sum_{\ell} \sum_{j \in A_\ell} T_j(K_j + f_j^r) \tag{13}$$

$$\text{subject to} \quad \sum_{j \in W_i^*} f_j^r - \sum_{j \in V_i^*} f_j^r = h_i^r \quad (i \in N_\ell^* ; \ell = 1, 2, \ldots L) \tag{14}$$

$$\sum_{j \in w_i^*} f_j^r - \sum_{j \in V_i^*} f_j^r = h_i^r - \sum_{j \in W_i^{**}} g_j^r + \sum_{j \in V_i^{**}} g_j^r \tag{15}$$

$$(i \in N_\ell^{**} ; \ell = 1, 2, \ldots, L)$$

$$f_j^r \geq 0 \qquad (j \in A)$$

$$\tag{16}$$

$$g_j^r > 0 \qquad (j \in S)$$

$$K_j = \sum_{p \neq r} \hat{f}_j^p \quad \text{where } \hat{f}_j^p \text{ is the given flow on link j for commodity p.}$$

Solution Algorithm for Problem P1

Following approach 2, the steps of the overall algorithm are as follows:

Step 0 Set $r \leftarrow 1$

$K_j \leftarrow 0 \qquad (j \in A)$

Step 1 Find a feasible solution $\bar{f}_j^r$ and $\bar{g}_j^r$ to problem $P2_r$; such a solution always exists because of the "strongly connected" assumption.

Set $\hat{f}_j^r \leftarrow \bar{f}_j^r$, $\hat{g}_j^r \leftarrow \bar{g}_j^r$

Set $K_j \leftarrow K_j + \bar{f}_j^r \qquad (j \in A)$

Set $r \leftarrow r + 1$

If $r = R$, set $\hat{f}_j^R \leftarrow 0$, $\hat{g}_j^R \leftarrow 0$ and go to Step 2;

Otherwise repeat Step 1.

Step 2 — Solve problem $P2_r$ and obtain the solution $\bar{f}_j^r$ and $\bar{g}_j^r$.

Set $K_j \leftarrow K_j + \bar{f}_j^r$ $\quad (j \in A)$

If the current solution $(\bar{f}_j^r, \bar{g}_j^r)$ is equal to the old solution $(\hat{f}_j^r, \hat{g}_j^r)$ within some tolerance, to Step 3;

Otherwise set $\hat{f}_j^r \leftarrow \bar{f}_j^r$, $\hat{g}_j^r \leftarrow \bar{g}_j^r$, $q \leftarrow 0$ and go to Step 4.

Step 3 — Set $q \leftarrow q + 1$.

If $q \neq R$, go to Step 4;

Otherwise terminate the algorithm.

Step 4 — Set $r \leftarrow r$ modulo $(R) + 1$.

Set $K_j \leftarrow K_j - \hat{f}_j^r$.

Go to Step 2.

Comments on the Algorithm

Step 0 is used just once for initialization.

Step 1 is also an initialization procedure to obtain a set of feasible flow values which are denoted by $\hat{f}_j^n$. When such flow values have been obtained for all commodities except one, the main iterative loop is entered at Step 2. This iterative loop involves alternating between Step 2 and Step 4 during which new flow patterns are obtained by commodity. Step 3 is used to determine when the termination condition has been reached, namely, when the solutions have settled down to within a tolerance for each commodity.

We now turn our attention to the solution of $P2_r$, which is solved by an adaptation of the generalized Benders Decomposition procedure as follows. The variables of $P2_r$ are partitioned into the two groups f_j^r and g_j^r, which we refer to as vectors f^r and g^r. The variables g^r are the so-called "complicating variables" which become the variables of the master problem. The set f^r are the variables of the subproblem. Feasibility constraints similar to (12) are added to the master problem.

Solution Algorithm for Problem $P2_r$

Step 0 Select a feasible g^r, say $\bar{g}^{ro}$.

Set $\bar{x}_r^o \leftarrow 0$.

UBD $\leftarrow \infty$.

Select a suitable convergence parameter ε.

Set $K \leftarrow 1$.

Step 1 Set $\bar{g}^r \leftarrow \bar{g}^{rK-1}$.

Solve the subproblem $P3_r$:

$$\min Z(f^r) = \sum_{\ell} \sum_{j \in A_\ell} T_j(K_j + f_j^r) \tag{17}$$

$$\text{subject to} \sum_{j \in W_i^*} f_j^r - \sum_{j \in V_i^*} f_j^r = h_i^r \qquad (i \in N_\ell^*;\ \ell = 1, 2, \ldots, L) \tag{18}$$

$$\sum_{j \in W_i^*} f_j^r - \sum_{j \in V_i^*} f_j^r = h_i^r - \sum_{j \in W_i^{**}} \bar{g}_j^r + \sum_{j \in V_i^{**}} \bar{g}_j^r \tag{19}$$

$$(i \in N_\ell^{**};\ \ell = 1, 2, \ldots, L)$$

$$f_j^r \geq 0 \qquad (j \in A). \tag{20}$$

$$K_j = \sum_{p \neq r} \hat{f}_j^p \qquad (j \in A)$$

This is a single commodity convex cost network flow problem which decomposes into L independent problems, one for each region.

Let $\bar{f}_j^r$, $\bar{Z}_r$ be the resulting solution and $\bar{u}_{ir}^k$ be the dual variables associated with constraints (19).

Set UBD = $\min\left\{\text{UBD}, Z_r(\bar{f}^r)\right\}$.

If $\bar{x}_r^{k-1} \geq \text{UBD} - \varepsilon$, terminate;

Otherwise continue to Step 2.

Step 2 Solve the master problem $P4_r$

$$\min x_r^k \tag{21}$$

subject to

$$x_r^k \geq M^k - \sum_{i \in N_\ell^{**}} \bar{u}_{ir}^k \left(\sum_{j \in W_i^{**}} g_j^r - \sum_{j \in V_i^{**}} g_j^r\right) \tag{22}$$

$$(k = 1, 2, \ldots, K)$$

$$\sum_{i \in N_\ell} \left[h_i^r - \left(\sum_{j \in W_i^{**}} g_j^r - \sum_{j \in V_i^{**}} g_j^r\right)\right] = 0 \tag{23}$$

$$(\ell = 1, 2, \ldots, L)$$

$$g_j^r \geq 0.$$

$$x_r^k \quad \text{unrestricted.}$$

Suppose the solution is $\bar{x}_r^k$, $\bar{g}^{rk}$.

Set $k \leftarrow k + 1$ and go to Step 1.

Comments on the Algorithm

At Step 1, the subproblem $P3_r$ decomposes into L independent problems. They are solved as single commodity uncapacitated network flow problems using the Frank-Wolfe algorithm, see [1, 2], which requires that successive solutions are found with linearized costs. A one-dimensional search is carried out over

successive pairs of solutions using the convex costs to find the minimum cost solution which becomes the initial solution for the next Frank-Wolfe iteration. It is not necessary to solve these problems all the way to optimality. This is allowed by the Generalized Benders Decomposition algorithm (see [12], Theorem 2.1 and [19]).

The problem at Step 2 is solved as a general linear program. The constraints (22) are accumulated as different problems at Step 1 are solved and the index k indicates from which problem the (M^k, u_{ir}^k) values are to be taken. The solution of the master problem is found in the BD computer code by solving its dual so that additional constraints (22) may be readily added. These master problem cuts are generated as follows.

In the notation of appendix 1 the constraints (22) are of the form

$$x_r^K \geq L^*(g^r, \bar{u}^k) \qquad k = 1, 2, \ldots, K$$

where

$$L^*(g^r, \bar{u}^k) = \inf_{f^r \in F^r} \left\{ Z_r(f^r) + \sum_{\ell} \sum_{i \in N_\ell^{**}} \bar{u}_{ir}^k \left[h_i^r - \sum_{j \in W_i^{**}} g_j^r \right.\right.$$

$$\left.\left. + \sum_{j \in V_i^{**}} g_j^r - \sum_{j \in W_i^*} f_j^r + \sum_{j \in V_i^*} f_j^r \right] \right\} \tag{24}$$

$$= \inf_{f^r \in F^r} \left\{ Z_r(f^r) + \sum_{\ell} \sum_{i \in N_\ell^{**}} \bar{u}_{ir}^k \left[- \sum_{J \in W_i^*} f_j^r + \sum_{j \in V_i^*} f_j^r \right] \right\} \tag{25}$$

$$+ \sum_{\ell} \sum_{i \in N_\ell^{**}} \bar{u}_{ir}^k \left[h_i^r - \sum_{j \in W_i^{**}} g_j^r + \sum_{j \in V_i^{**}} g_j^r \right]$$

where the region F^r is defined by constraints (18) and (20) of the problem in Step 1. Now consider the dual of the problem $P3_r$ when $\bar{g}^r = \bar{g}^{r\ k-1}$.

$$\sup_{\substack{u_{ir}^k \\ \text{unrestricted}}} \inf_{f^r \in F^r} \left\{ Z_r(f^r) + \sum_{\ell} \sum_{i \in N^{**}} u_{ir}^k \left[h_i^r - \sum_{j \in W_i^{**}} \bar{g}_j^r + \sum_{j \in V_i^{**}} \bar{g}_j^r - \sum_{j \in W_i^{*}} f_j^r + \sum_{j \in V_i^{*}} f_j^r \right] \right\}$$

$$= \sup_{u} \left[\inf_{f^r \in F^r} \left\{ Z_r(f^r) + \sum_{\ell} \sum_{i \in N^{**}} u_{ir}^k - \left(\sum_{j \in W_i^{*}} f_j^r + \sum_{j \in V_i^{*}} f_j^r \right) \right\} \right.$$

$$\left. + \sum_{\ell} \sum_{\substack{i \in N^{**} \\ \ell}} u_{ir}^k \left(h_i^r - \sum_{j \in W_i^{**}} \bar{g}_j^r + \sum_{j \in V_i^{**}} \bar{g}_j^r \right) \right]. \tag{26}$$

The solution is $\bar{f}^r$, $\bar{u}^r$, $\bar{Z}_r$ so that

$$L^*(g^r, \bar{u}^k) = \bar{Z}_r - \sum_{\ell} \sum_{i \in N_{\ell}^{**}} \bar{u}_{ir}^k \left(h_i^r - \sum_{j \in W_i^{**}} \bar{g}_j^r + \sum_{j \in V_i^{**}} \bar{g}_j^r \right)$$

$$+ \sum_{\ell} \sum_{i} \bar{u}_{ir}^k \left(h_i^r - \sum_{j \in W_i^{**}} g_j^r + \sum_{j \in V_i^{**}} g_j^r \right)$$

$$= \bar{Z}_r + \sum_{\ell} \sum_{i \in N^{**}} \bar{u}_{ir}^k \left(\sum_{j \in W_i^{**}} \bar{g}_j^r - \sum_{j \in V_i^{**}} \bar{g}_j^r \right)$$

$$+ \sum_{\ell} \sum_{i} \bar{u}_{ir}^k \left(- \sum_{j \in W_i^{**}} g_j^r + \sum_{j \in V_I^{**}} g_j^r \right). \tag{27}$$

The constant M^k in (22) may thus be found by adding the quantity

$$\sum_{\ell} \sum_{i \in N^{**}_{\ell}} \bar{u}^k_{ir} \left(\sum_{j \in W^{**}_i} \bar{g}^r_j - \sum_{j \in V^{**}_i} \bar{g}^r_j \right) \tag{28}$$

to the optimal objective value $\bar{Z}_r$ obtained for problem $P3_r$.

In the implementation of the algorithm, each constraint of form (22) has been applied as L separate constraints, one for each subregion, with a corresponding change of form for the objective (21). This is discussed in [19].

4.3 IMPLEMENTATION

The algorithm of Section 4.2 has been implemented in a test bed computer program for the purpose of identifying its operating characteristics and investigating its performance. In this section we give a brief overview of the code and its capabilities. For convenience we refer to this Benders decomposition program as the "BD Program."

Unlike CATNAP (reference [2]) and the SP code (Section 3.3), BD was the initial implementation of an algorithm whose properties were not well understood. For this reason a very flexible, general purpose program was written so that the many different options of the Benders decomposition could be conveniently investigated. Internal array dimensions and external data sets were allocated so that very large problems could be solved without modifying the basic program; this also adds to the flexibility of BD.

Unfortunately, the great generality of present BD implementation leads to extremely poor performance, especially for small and medium sized problems when the overhead is expectedly high. We will discuss this point further in Section 4.4; for now, we note that the system design described

in this section can be substantially improved now that something of its performance is known.

The BD code contains 39 modules which may be considered in four main groups: data input, traffic assignment, linear program and direct access data set management. The code is designed to solve only the optimal traffic assignment problem with fixed demands and system optimal assignment.

The basic problem data structure is similar to that used by the SP code discussed in Section 3.3 except that BD makes no provision for keeping part of the data in main memory at all times; all arrays are stored externally in one of four disk files and are brought into memory as needed. The four disk files contain:

(1) The topography (node and link data) for each region as well as the total flow on each link

(2) The trip table and link flows for each origin node in each region

(3) The master problem data for each origin node

(4) The master problem solution values and generated Benders cuts for each origin node.

To allow maximum flexibility in solving a traffic assignment problem these four disk files are all direct-access files, i.e., one may look at origins or regions in any desired sequence. However, both the direct access file organization and the very large size of the files add greatly to the cost of problem solution.

The traffic assignment modules include various bookkeeping routines (link cost evaluation, cut generation and a linear program interface) and a single subprogram which solves the basic Benders subproblem of $P3_r$, which is to find an optimal assignment of flows for a single origin in one region given:

(1) The supply and demand values for the zones in the region (as originally input)

(2) The required flows across the boundary nodes (as set by the master problem)

(3) The flows from all other origins currently assigned to the arcs of the region.

The subproblem is solved by an application of the Frank-Wolfe algorithm. Because boundary node flows into a region are treated as origins, the basic shortest path routine of CATNAP or SP can no longer be used; instead, a transportation problem must be solved and a code called GNET[15] is used for this purpose. GNET also provides the dual multipliers which are needed for the master problem cuts.

The linear program modules are used to solve the Benders Master problem $P4_r$. The BD code uses a slightly modified version of the Control Analysis Mathematical Programming System (CAMPS) described in reference [16].

4.4 APPLICATION TO SUBAREA FOCUSING

The subarea focusing problem involves the investigation of changes to network topology and/or arc parameters within a given geographic region. The basic assumption made is that the resulting arc flows in other regions are not of major interest to the planner, possibly because the changes are not expected to be significant. Thus, the regions outside the target region (which we call the window) need not be modeled in complete detail.

One approach (which Dial [20] calls the subarea windowing problem) is to merely hold constant the input and output flow values g_j^r at the boundary nodes of the window; the inference is that there is no external response to any of the changes within the window. This seems to be an unnecessarily restrictive assumption. Another approach is to

aggregate the external regions in some fashion so as to reduce the level of detail while hopefully preserving a degree of realistic response. It is very difficult, however, to specify general rules for carrying out the aggregation.

One suggestion for solving the subarea focusing problem involves the Benders Decomposition algorithm described above. An initial solution for the entire network is first obtained; the window is designated as one of the regions, region ℓ' for example. The algorithm, however, must be modified slightly. Recall that changes in the K_j values may invalidate the dual feasibility of $(\bar{f}_j^r, \bar{u}_{ir}^k)$; our technique requires a set of subproblem $P3_r$ solutions which are simultaneously dual feasible for all commodities $r \in R$ and all regions $\ell \neq \ell'$. Thus, for the final pass over the commodities in the P1 algorithm, we modify Step 2 so that the old value of f_j^r is replaced in K_j instead of the new optimal value obtained from problem $P2_r$.

The changes within region ℓ' are then made and the algorithm is repeated.

4.5 COMPUTATIONAL EXPERIENCE AND RESULTS

In addition to small test problems, the BD program has been used to solve the 81-node medium-sized problem described in Section 2.4. The basic approach was to obtain optimal or near-optimal solutions to each subproblem and to repeat the master problem - subproblem iteration until the upper and lower bounds on the optimum value were very close. In this way, the most accurate numerical results could be used for testing the algorithm.

The convergence of the individual subproblems was quite slow. (Recall that a subproblem requires a minimum cost traffic assignment of the trips from a single origin to the links of a single region with the flows from the other

origins held constant.) Since the subproblems are solved by the Frank-Wolfe algorithm this slow convergence is not unexpected; see [2] for further discussion. Note also that since the entire subproblem will fit into the computer's main memory, additional Frank-Wolfe iterations increase only the processor time and do not require additional input or output.

The convergence of the Benders decomposition procedure itself seems to be more rapid than that of the Frank-Wolfe algorithm. The results in Table 4.1 are for the initial problem solved after data input, i.e., the assignment of the demands for origin 1. We note first that only five to six iterations are needed to obtain bounds which are fairly close; the master problem has five decision variables in this case so we conjecture that allowing about one Benders iteration per boundary node should be a reasonable strategy.

The second and third columns of Table 4.1 reflect cases when the subproblems were not solved to optimality; the subproblems were stopped when the Frank-Wolfe bounds on the optimum subproblem solution value were within two percent and ten percent, respectively, of each other. It is apparent that this modification results in a slight degradation of the bounds obtained but in both cases the result is substantially the same.

Inspection of Table 4.1 reveals that the improvement in the upper bound is not monotone. Initially this is due to the fact that the master problem has only a few cuts and the solutions $\bar{g}^r$ passed to the subproblems are not very effective. The effect does not die out, however, as more cuts are accumulated in the master; at iteration 5, for example, a much worse subproblem solution is obtained and it is not until iteration 9 that the situation improves again.

For this reason a feature has been added to the BD code which allows the user to keep only as much of the current master problem solution as will actually improve the subproblem objective function value. This is done by

TABLE 4.1 - CONVERGENCE OF THE BENDER'S DECOMPOSITION ALGORITHM FOR A SINGLE ORIGIN ZONE

Iteraticns of Bender's Algorithm	Subproblem Optimal Solution		Subproblem Within 2% of Optimal		Subproblem Within 10% of Optimal		Best Current Solution (subproblem 2%)	
	Upper Bound	Lower Bound	Upper Bound	Lower Bound	Upper Bound	Lower Bound	Upper Bound	Lower Bound
Initial	38537.6	0	38537.6	0	38537.6	0	38537.6	0
1	56089.3	29347.7	56119.5	29395.3	56246.3	30485.5	38120.5	29395.3
2	41902.7	34510.1	42126.9	34390.9	42187.3	34608.8	38120.5	34383.4
3	38435.5	36466.1	38565.5	36376.8	38560.4	36712.8	38120.5	36373.9
4	37889.2	37124.6	37930.0	37072.8	38169.3	37138.2	37930.0	37073.2
5	37531.6	37366.5	37528.4	37349.2	37561.0	37246.3	37528.3	37350.0
6	37742.1	37474.8	37853.9	37467.1	37585.4	37345.0		
7	37580.2	37497.6	37599.5	37494.6	37604.7	37484.0		
8	37563.2	37507.0	37556.3	37504.9	37532.5	37489.3		
9	37515.6	37509.5	37518.8	37507.2	37644.2	37500.0		
10	37514.4	37513.5	37516.2	37507.2	37528.0	37508.3		

computing the optimum linear combination of the single commodity flow values $\bar{f}^r$ which were found for the current and the previous values of the $\bar{g}^r$ vectors; the weight factor is found by a single dimension Newton-Raphson search:

$$\min_{\alpha} \sum_{j \in A} \left\{ T_j \; K_j + \alpha \bar{f}_j^{rK-1} + (1 - \alpha) \; \bar{f}_j^{rK} \right\}. \tag{29}$$

If the resulting α value is α^* the $\bar{g}^r$ values are adjusted according to

$$\bar{g}_j^{rK} = \alpha^* \bar{g}_j^{rK-1} + (1 - \alpha^*) \; \bar{g}_j^{rK}. \tag{30}$$

The result of applying this heuristic to the 81-node problem is shown in the fourth column of Table 4.1 where it may be seen that there is sometimes a substantial effect (e.g., iterations 1 and 2) and at other times almost no effect (iteration 4). It seems prudent to have available a "best" solution in case of early termination but apparently the method performs best when "bad" solutions are sent to the subproblems so that good cuts can be returned to the master. We feel that the results do not justify the use of the heuristic.

The discussion of computational results for BD has thus far focused on the solution of problems $P3_r$ - $P4_r$ for a single commodity. All of the computer runs in Table 4.1 were carried out for two passes over each of the 31 origins with the results displayed in Table 4.2.

It is clear that the BD code at the present time is inferior to both CATNAP and the SP code. It should be pointed out, however, that the great generality and flexibility of BD involve extensive use of peripheral storage; it is estimated that at least half the CPU time and about three-quarters of the total processing charge for the BD runs is related to input-output operations. It is likely that a properly tuned code with less flexible data structures could substantially reduce this overhead.

TABLE 4.2 - PERFORMANCE OF THE BD TRAFFIC ASSIGNMENT COMPUTER CODE

Run Description	Final Solution	CPU Time	Disk I/O	Processing Cost
Optimal Subproblem	29341.5	108.81 sec	14,067	$49.46
2% Subproblem	29358.9	69.50 sec	14,340	$39.51
10% Subproblem	29349.7	68.27 sec	14,637	$39.53
Heuristic	29511.1	75.25 sec	18,145	$46.19
CATNAP	29322.8	4.41 sec	13	$1.34

It should also be noted that up to ten Benders iterations were allowed for each of the commodity problems in these runs. The results of Table 4.1 indicate that about half that many would probably have been adequate, at least for the first pass over all the origins. Such "tactical" issues as this have not been explored.

In summary, then, the BD algorithm of Section 4.2 has been demonstrated to have reasonable convergence properties in medium-sized traffic assignment problems. The current implementation of the algorithm involves so much overhead, however, that is is not suitable for solving large problems and further work is required to extend the results reported here to practical network problems.

5. GEOGRAPHIC DECOMPOSITION USING THE DANTZIG-WOLFE DECOMPOSITION ALGORITHM

In this section we present a method for the solution of the basic traffic assignment problem using geographic decomposition in conjunction with the Dantzig-Wolfe Decomposition Principle [18].

DANTZIG-WOLFE DECOMPOSITION IN TRANSPORTATION NETWORK ANALYSIS

Assume that the traffic assignment problem has already been decomposed by origin (commodity). The problem for origin r would be:

$\underline{Q1^r}$

$$\text{Minimize} \quad Z1^r = \sum_{j \in A} T_j (K_j^r + f_j^r) \tag{31}$$

$$\text{such that} \quad \sum_{j \in W_i} f_j^r - \sum_{j \in V_i} f_j^r = h_i^r \qquad (i \in N) \tag{32}$$

$$f_j^r \geq 0 \qquad (j \in A) \tag{33}$$

$$\text{where} \quad K_j^r = \sum_{p \neq r} f_j^p \qquad (j \in A) \tag{34}$$

and

A is the set of arcs

N is the set of nodes

W_i is the set of arcs originating from node i

V_i is the set of arcs terminating at node i

T_j is the arc cost for arc j

h_i^r is the "supply" of commodity r at node i.

We now decompose the network geographically by a partition of nodes. Let S be the set of deleted arcs so the network (N, A) is decomposed into a series of disjoint subnetworks (N_ℓ, A_ℓ) $\ell = 1, \ldots, L$, where

$$N = N_1 \cup N_2 \cup \ldots \cup N_\ell \tag{35}$$

$$A = A_1 \cup A_2 \cup \ldots \cup A_L \cup S. \tag{36}$$

Define $W_i^{**} = W_i \cap S$, $W_i^{*} = W_i - W_i^{**}$ (37)

$$V_i^{**} = V_i \cap S, \quad V_i^{*} = V_i - V_i^{**}. \tag{38}$$

Now consider each arc $j \in S$ to consist of two arc segments with flows u_j^r and v_j^r (u_j^r is the flow across the "tail half" of the arc and v_j^r is the flow across the "head half" of the arc (this is equivalent to creating an artificial node "midway" in the arc, as shown in Figure 5.1)).

Figure 5.1 Boundary Arc

We will (arbitrarily) assign the arc cost T_j to u_j^r and get a problem equivalent to $Q1^r$:

$\underline{Q2^r}$

$$\text{Minimize } Z2^r = \sum_{\ell=1}^{L} \sum_{j \in A_\ell} T_j(K_j^r + f_j^r) + \sum_{j \in S} T_j(K_j^r + u_j^r) \tag{39}$$

such that

$$\sum_{j\varepsilon W_i^*} f_j^r - \sum_{j\varepsilon V_i^*} f_j^r + \sum_{j\varepsilon W_i^{**}} u_j^r - \sum_{j\varepsilon V_i^{**}} v_j = h_i^r \quad (i\varepsilon N_\ell;\ \ell=1, \ldots, L) \tag{40}$$

$$u_j^r - v_j^r = 0 \qquad (j\varepsilon S) \tag{41}$$

$$f_j^r,\ u_j^r,\ v_j^r \geq 0 \qquad (j\varepsilon A) \tag{42}$$

where

$$K_j^r = \begin{cases} \sum_{p\neq r} u_j^p & \text{if } j\varepsilon S \\ \sum_{p\neq r} f_j^p & \text{otherwise.} \end{cases} \tag{43}$$

We notice that constraint (41) forms the only connection between regions in $P2^r$. Thus, if (41) was constrained in a Dantzig-Wolfe master problem, we could have L subproblems each of which deals only with flows within a region ℓ as follows:

Subproblem r,ℓ (SP_ℓ^r)

Minimize

$$ZS_{\ell,k}^r = \sum_{j\varepsilon A_\ell} T_j(K_j^r + f_j^r) + \sum_{i\varepsilon N_\ell} \sum_{j\varepsilon W_i^{**}} T_j(K_j^r + u_j^r) + \sum_{j\varepsilon S} D_{\ell,j}(u_j^r, v_j^r) \tag{44}$$

such that

$$\sum_{j\varepsilon W_i^*} f_j^r - \sum_{j\varepsilon V_i^*} f_j^r + \sum_{j\varepsilon W_i^{**}} u_j^r - \sum_{j\varepsilon V_i^{**}} v_j^r = h_i^r \quad (i\varepsilon N_\ell) \tag{45}$$

$$f_j^r \geq 0 \qquad (j \in A_\ell) \tag{46}$$

$$u_j^r \geq 0 \qquad (j \in W_i^{**},\ i \in N_\ell) \tag{47}$$

$$v_j^r \geq 0 \qquad (j \in V_i^{**},\ i \in N_\ell) \tag{48}$$

where
$$D_{\ell,j}(u_j^r,\ v_j^r) = \begin{cases} \hat{\Pi}_{j,k}^r u_j^r & \text{if } j \in W_i^{**} \text{ for some } i \in N_\ell \\ -\hat{\Pi}_{j,k}^r v_j^r & \text{if } j \in V_i^{**} \text{ for some } i \in N_\ell \\ 0 \quad \text{otherwise} \end{cases} \tag{49}$$

and $\hat{\Pi}_{j,k}^r$ is the latest (kth) iterate of the jth dual variable from the master problem (MP^r) below. (Notice that the $\hat{\Pi}_{j,k}^r$ are unrestricted in sign and that we should expect to get unbounded solutions to SP_ℓ^r when using the Frank-Wolfe procedure).

Master Problem r (MP^r)

$$\text{Minimize} \quad ZM_k^r = \sum_{\ell=1}^{L} \sum_{t=1}^{k-1} C_{\ell,t}^r \xi_{\ell,t}^r \tag{50}$$

such that
$$\sum_{\ell=1}^{L} \sum_{t=1}^{k-1} a_{\ell,j,t}^r \xi_{\ell,t}^r = 0 \quad : \Pi_{j,t}^r \quad (j \in S) \tag{51}$$

$$\sum_{t=1}^{k-1} b_{\ell,t}^r \xi_{\ell,t}^r = 1 \qquad (\ell=1, \ldots, L) \tag{52}$$

$$\xi_{\ell,t}^r \geq 0 \qquad (\ell=1, \ldots, L;\ t=1, \ldots, k-1) \tag{53}$$

where the columns of coefficients for primal variable $\xi^r_{\ell,t}$ are derived from the solutions of SP^r_ℓ according to the Dantzig-Wolfe procedure in terms of (41). This can be described as follows:

Assume that SP^r_ℓ at iteration t results in a feasible solution, either finite or unbounded. Denote the relevant $(ZS^r_{\ell,t} - \sum_{j\epsilon S} D_{\ell,j}(u^r_j, v^r_j),\ u^r_j,\ v^r_j)$ elements of either the extreme point or the extreme ray by $(\alpha^r_{\ell,t},\ \beta^r_{j,t},\ \gamma^r_{j,t})$. Then:

$$c^r_{\ell,t} = \alpha^r_{\ell,t} \tag{54}$$

$$a^r_{\ell,j,t} = \begin{cases} \beta^r_{j,t} & \text{if } j\epsilon W_i^{**} \text{ for some } i\epsilon N_\ell \\ -\gamma^r_{j,t} & \text{if } j\epsilon V_i^{**} \text{ for some } i\epsilon N_\ell \\ 0 & \text{otherwise,} \end{cases} \tag{55}$$

$$b^r_{\ell,t} = \begin{cases} 1 & \text{if extreme point} \\ 0 & \text{otherwise.} \end{cases} \tag{56}$$

The overall solution method would be to apply the Frank-Wolfe Algorithm, see for example [1, 2]. For each major Frank-Wolfe iteration the objective form is linearized and a new total flow pattern is computed by solving for each commodity (origin) in turn using the above algorithm. These flows are added to obtain the total flow pattern.

As mentioned earlier, we have not attempted to implement this procedure. It would appear to be of about the same order of complexity as the Bender's Geographic Algorithm. This decomposition method would seem to be unsuitable for nonlinear subarea focusing because the nonlinear

objective is treated by Frank-Wolfe approximations iteratively in an outer loop. We would want to approximate the conditions outside the window by using only old proposals from these regions. If the $c^r_{\ell,t}$ reflected merely a local approximation to the nonlinear subproblem costs, as would be the case with the above procedure, it would appear that the old proposals would be "invalid" as soon as new derivatives were taken. By "invalid" we mean that they would not imply the current objective approximation and it would seem fruitless to use an optimization procedure to select their weights (ξ's). However it would be possible to update the $c^r_{\ell,t}$'s to reflect the changing linear objective so that at least the master problem would be selecting the best of the "suboptimal" proposals from the non-window regions which, perhaps, is all that is wanted in subarea focusing.

6. CONCLUSIONS

When this research was begun it was believed that taking advantage of the special geographic structure of some transportation networks could result in algorithms which were more efficient than the basic Frank-Wolfe algorithm used in UROAD and CATNAP, and that these methods would be useful in studying the problem of subarea focusing.

In this study we have explored the use of three geographic decomposition approaches, one based on decomposition for the shortest path calculation referred to as the SP method, one using the Generalized Benders Decomposition algorithm due to Geoffrion referred to as the BD method, and one using the Dantzig-Wolfe Decomposition algorithm which has been formulated but not coded. The principle problem addressed with all three approaches is the traffic assignment problem with fixed demands. However, as pointed out in the introduction to this report, there are other related problems for which this research could be applicable.

The SP method has proven to be the most successful. There is some flexibility in its use. If enough computer high speed storage is available to hold each subnetwork, it performs better than the CATNAP code which uses the same basic Frank-Wolfe algorithm but without making use of a geographic decomposition of the network. For the largest network used, which is an aggregated network version of the Washington DC highway system, the observed improvement in performance could be as much as about 40 percent. The improvement tends to be most marked at the highest level of decomposition, which was a decomposition into four

geographic subregions in our study. On the other hand, the SP method can be used in a straightforward manner when the available high speed computer storage is sufficient to hold only the largest subnetwork. In the latter case, there is an added overhead cost involved in running the problem.

The BD algorithm has behaved rather poorly, even allowing for the fact that the code is an experimental one. Furthermore, its usefulness for the problem of subarea focusing, we feel, is, at best, marginal.

The Dantzig-Wolfe approach to geographic decomposition has been included as an interesting alternative approach. In a sense, it is a primal solution procedure where the BD approach is a dual procedure. We have no reason to believe that it would be a more effective procedure than the latter.

REFERENCES

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

2. Harvey, R.P., and D.W. Robinson, "Computer Code for Transportation Network Design and Analysis," Department of Transportation Report No. DOT-TSC-OCT-77-39, May 1977.

3. Nguyen, S., "An Algorithm for the Traffic Assignment Problem," Transportation Science, Vol. 8, 1974, p. 203-216.

4. Murchland, J.D., "Road Network Traffic Distribution in Equilibrium," paper for the Tagung ueber "Mathematische Methoden in den Wirtschaftswissenschaften," Mathematisches Forschungsinstitut, Oberwolfach, October, 1969.

5. Dijkstra, E.W., "A Note on Two Problems in Connexion with Graphs," Numer. Math., Vol. 1, p. 269-281, 1959.

6. Floyd, R.W., "Algorithm 97: Shortest Path," Comm. ACM, Vol. 5, p. 345, July, 1962.

7. Bellman, R., "On a Routing Problem," Quarterly Applied Mathematics, Vol. 16, p. 87-90, 1958.

8. Hu. T.C., "A Decomposition Algorithm for Shortest Paths in a Network," Operations Research, Vol. 16, 1968, p. 91-102.

9. Hu, T.C., and Torres, W.T., "Shortcut in the Decomposition Algorithm for Shortest Paths in a Network," IBM Journal of Research and Development, Vol. 13, No. 4, July, 1969.

10. Glover, F., D. Klingman, and A. Napier, "A Note on Finding All Shortest Paths," Trans. Sci., Vol. 8, p. 3-13, February, 1974.

11. Stanford University, FORTUNE and WATTUNE User's Guide, Stanford CA, 1975.

12. Geoffrion, A.M., "Generalized Benders Decomposition," Journal of Optimization Theory and Applications, Vol. 10, No. 4, 1972, p. 237-260.

13. Benders, J.F., "Partitioning Procedures for Solving Mixed-Variables Programming Problems," Numerishche Mathematik, Vol. 4, 1962.

14. Florian, M and S. Nguyen, "A Method for Computing Network Equilibrium with Elastic Demands," Transportation Science, Vol. 8, p. 321-332, 1974.

15. Bradley, G.H., G.G. Brown, and G.W. Graves, "Design and Implementation of Large Scale Primal Transshipment Algorithms," Naval Postgraduate School Technical Report, NPS-55BSBW-76091, Monterey CA, September 1976.

References (Continued...)

16. Harvey, R.P., and R.D. McKnight, Development and Implementation of TAC Resourcer, Control Analysis Corporation Technical Report No. R-68-FG, May 1976.

17. Dial, R.B., "Algorithm 360: Shortest Path Forest with Topological Ordering," Comm. ACM, vol. 12, no. 11, November 1969, pp. 632-633.

18. Dantzig, G.B., and P. Wolfe, 1961-1, "The Decomposition Algorithm for Linear Programming," Econometrica, Vol. 29, No. 4, October 1961.

19. Maier, S.F., and D.W. Robinson, "The Application of Geographic Decomposition to the Solution of Large-Scale Traffic Assignment Problems," Graduate School of Business Administration, Duke University, Durham NC 27706, Working Paper 213.

20. Dial, R.B., "The Development of UMTA's Transportation Planning System," presented at the Annual Meeting of the Highway Research Board, Washington DC, January 1974.

21. Pape, U., "Implementation and Efficiency of Moore-Algorithms for the Shortest Route Problem," Math. Prog., Vol. 7, 1974, p.212-222.

APPENDIX 1

Generalized Benders Decomposition

Consider the problem:

$$\min_{x,y} f(x, y),$$

subject to $G(x,y) = 0$, $x \in X$, $y \in Y$.

Generalized Benders Decomposition allows the solution of such problems by partitioning into a problem in variables x only and a problem in variables y only that must be solved several times. The algorithm, in a form suitable for our purpose, is as follows:

Step 1. Find a $\bar{y} \in Y$ and set $J = 1$.

Solve the problem:

$$\min_{x \in X} f(x, \bar{y}), \text{ subject to } G(x, \bar{y}) = 0.$$

Obtain an optimal (or near-optimal) solution $\bar{x}^J$ and the corresponding multiplier vector $\bar{u}^J$. Select the convergence tolerance parameter $\varepsilon > 0$ and put UBD = $f(\bar{x}^J, \bar{y})$; form the function

$$L^*(y, u) = \inf_{x \in X} \{f(x, y) + u^t G(x, y)\}, \quad y \in Y, \ u \text{ unrestricted.}$$

Step 2. Solve the problem:

$$\min_{yo,\, y} \ y_o \text{ subject to } y_o \geq L^*(y, \bar{u}^j), \ j = 1, \ldots, J,$$

by an applicable algorithm and obtain $\bar{y}_o^J$, $\bar{y}^J$. If $\bar{y}_o^J \geq$ UBD $- \varepsilon$ terminate, otherwise set $J = J + 1$.

Step 3. Solve the problem:

$$\min_{x \in X} f(x, \bar{y}^{J-1}), \text{ subject to } G(x, \bar{y}^{J-1}) = 0,$$

and obtain new $\bar{u}^J$, $\bar{x}^J$; put UBD = min { UBD, $f(\bar{x}^J, \bar{y}^{J-1})$ } . If $\bar{y}_o^{J-1} \geq$ UBD $- \varepsilon$ terminate, otherwise go to Step 2.