REPORT NO. DOT-TSC-RSPD-78-6

TRANSPORTATION NETWORK ANALYSIS AND DECOMPOSITION METHODS

T.L. Magnanti
R.W. Simpson

Massachusetts Institute of Technology
Center for Transportation Studies
Operations Research Center
Cambridge MA 02139

MARCH 1978

FINAL REPORT

DOCUMENT IS AVAILABLE TO THE U.S. PUBLIC THROUGH THE NATIONAL TECHNICAL INFORMATION SERVICE, SPRINGFIELD, VIRGINIA 22161

Prepared for

U.S. DEPARTMENT OF TRANSPORTATION
RESEARCH AND SPECIAL PROGRAMS DIRECTORATE
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.
DOT-TSC-RSPD-78-6		

4. Title and Subtitle	5. Report Date
TRANSPORTATION NETWORK ANALYSIS AND DECOMPOSITION METHODS	March 1978
	6. Performing Organization Code
7. Author(s) T.L. Magnanti and R.W. Simpson	8. Performing Organization Report No. DOT-TSC-RSPD-78-6
9. Performing Organization Name and Address Massachusetts Institute of Technology* Center for Transportation Studies Operations Research Center Cambridge MA 02139	10. Work Unit No. (TRAIS) OS750/R8526 11. Contract or Grant No. DOT-TSC-1058
12. Sponsoring Agency Name and Address U.S. Department of Transportation Research and Special Programs Directorate Office of Transportation Programs Bureau Office of Systems Engineering Washington DC 20590	13. Type of Report and Period Covered Final Report July 1975-September 1976 14. Sponsoring Agency Code

15. Supplementary Notes

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

16 Abstract

This report outlines research in transportation network analysis using decomposition techniques as a basis for problem solutions. Two transportation network problems were considered in detail: a freight network flow problem and a scheduling problem for a transportation system with multiple vehicle fleets. Several different approaches for decomposing the overall problems into smaller subproblems were examined.

A third problem, dealing with the routing of vehicles over a series of demand points, was also examined using a different solution approach.

The report presents detailed mathematical formulations of the different problems, discusses the solution approaches examined and presents computational results of the research.

17. Key Words	18. Distribution Statement
Transportation Network Analysis, Optimization, Decomposition, Routing, Scheduling	DOCUMENT IS AVAILABLE TO THE U.S. PUBLIC THROUGH THE NATIONAL TECHNICAL INFORMATION SERVICE, SPRINGFIELD, VIRGINIA 22161

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	204	

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

PREFACE

This report describes work performed in the first year of a continuing research study aimed at improving large scale analytical techniques in the area of Transportation Network Analysis and Decomposition Methods. The work was carried out by members of a research team drawn from the Operations Research Center and Center for Transportation Studies at Massachusetts Institute of Technology (MIT) under a contract in the Transportation Advanced Research Program (TARP) of the U.S. Department of Transportation (DOT).

The principal investigators were Thomas L. Magnanti and Robert W. Simpson, working with a team of graduate student research assistants -- Bruce Golden, Peeter Kivestu, Arjang Assad, Richard Wong, Hedayat Ashtiani, and Ghazala Sadiq. The contract was supervised by the Transportation Systems Center for the Office of Secretary, DOT. We would like to express our thanks to Louis Fuertes and E. J. Roberts for their efforts and encouragement.

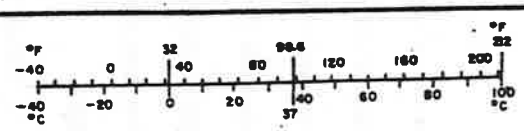
METRIC CONVERSION FACTORS

Approximate Conversions to Metric Measures

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

Approximate Conversions from Metric Measures

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



CONTENTS

CONTENTS (CONTINUED)

LIST OF ILLUSTRATIONS

LIST OF TABLES

1. INTRODUCTION

The formulation of many network problems in transportation exhibits a modular structure which can be exploited in developing a solution technique. The network often can be decomposed in several dimensions such as time, vehicle type, crew base, commodity flow, and so forth, which produce "sub-problems" that can easily be solved separately, and then recombined to achieve an overall optimum. This is in direct contrast to "aggregation" strategies which reduce the scale of the network by removing network structure, and solving an associated "macro" problem using reduced information.

The underlying network structure for transportation (or communication) problems are called "route maps" and "schedule maps". Route maps describe the nodes and links of the system in a three-dimensional space, or describe the "geography" of the system. Schedule maps extend the route map into the time dimension to describe the scheduling and routing of transportation flows over both space and time.

Both types of maps (or networks) are often replicated in the process of modeling transportation processes. Thus, we have "multi-period" maps, "multi-copy" maps which immediately suggest obvious decompositions of this enlarged network structure. Alternatively, given a route or schedule map, there are problems which group sub-portions of that network structure together (e.g. crew scheduling with multiple crew bases, truck delivery with multi-depots). Thus we have a set of "multi-region" networks which again immediately suggest decomposing the problem.

For these problems, specialized network "decomposition" techniques

have been an area of research at MIT over the past several years. Application of these techniques appears to be most fruitful in the transportation area. While the goals of this contract are to evaluate the usefulness of those "decomposition" techniques, we set a higher goal to demonstrate to DOT and the transportation industry that substantial, short-term payoffs are possible in terms of system productivity, efficiency, and improved planning methods from a much larger investment in the research, development and application of these new OR techniques to transportation network problems.

The first year's work involved two phases. The first was a review phase of prior work in this area. This was split into a review of large scale network models and their applications to transportation, and a review of large scale network methodology and computational capability. The first phase produced two bibliographies [1-6,1-8] covering each of these review areas, and led to a decision to study three specific areas in the second phase of the study. The first phase also produced several reports and working papers of a survey nature [1-2,1-9,1-10] which are listed at the end of this section.

The second phase covered the development and testing of improved computational methods in the three problem areas selected: Multifleet Routing, Vehicle Routing, and the generic set of Freight Flow problems. Here detailed mathematical research into algorithms was coupled with development of computer codings to test the performance of new computational methods on sets of small and large scale problems in each of the three areas. This work was a mixture of experimental and theoretical

investigation which met with varying degrees of success and failure [1-1, 1-3,1-4,1-5,1-7]. Work on the variants of the Freight Flow problem is continuing into the second year. This is a summary report describing all the work areas of the study. There will continue to be separate reports and papers on particular topics issued throughout the study which will provide information for readers interested in further detail.

REFERENCES FOR SECTION 1.

1-1 Aashtiani, H. and Magnanti, T., "Implementing Primal-Dual Network Flow Algorithms". MIT/ORC Working Paper OR-055-76, June 1976.

1-2 Assad, A., "Multicommodity Network Flows: A Survey," MIT/ORC Working Paper, OR-045-75, December 1975

1-3 _____, "Solution Techniques for the Multicommodity Flow Problem", S.M. Thesis, M.I.T. Sloan School, May 1976.

1-4 Golden, B., "A Statistical Approach to the Travelling Salesman Problem", MIT/ORC Working Paper OR-052-76, April 1976.

1-5 _____, "Large Scale Vehicle Routing and Related Combinatorial Problems", Ph.D. Thesis, M.I.T. Operations Research Center, June 1976.

1-6 Golden, B., Magnanti, T.L., "Deterministic Network Optimization - A Bibliography", MIT/ORC Working Paper, June 1976.

1-7 _____, and Magnanti, T.L., Nguyen, H.Q., "Implementing Vehicle Routing Algorithms", MIT/ORC Technical Report 115, September 1975.

1-8 Kivestu, P., Simpson, R.W., "A Bibliography of Network Models in Transportation", MIT/CTS Report 76-11, September 1976.

1-9 Magnanti, T.L., "Optimization for Sparse Systems", in Sparse Matrix Computations (J.R. Bunch and D.J. Rose, eds.) Academic Press, New York, 1976.

1-10 Wong, R., "A Survey of Network Design Problems", MIT/ORC Working Paper", OR-053-76, May 1976.

2. TRANSPORTATION NETWORK MODELS

In this section we develop a general mathematical framework for three applications areas of transportation networks:

1) Freight Flow Problems
2) Multifleet Routing Problems
3) Vehicle Routing Problems

These three applications, their generic structures and solution techniques, are summarized in Table 2.1. In the following we elaborate on the generic structures, defining the objective functions and constraints and present the mathematical formulations.

One common feature of these applications is that they all belong to the general class of (mixed) integer programming models. This type of model is associated quite naturally with strategic capital expenditure decisions -- possibilities of adding new depots or adding new roadways to a freight flow system, for example, are modelled by 0 (don't add) or 1 (add) variables. Integer programs also model detailed operational decision making, especially when precedence relationships like those of vehicle routing are dominant features of a system's operation. Certain tactical decision-making problems like the assignment of vehicles to scheduled routes, as in multifleet routing, also lead to integer programming formulations. Consequently, our study of integer programming type applications covers a wide range of strategic, tactical, and operational situations.

Recent advances in the development and application of decomposition techniques for linear and integer programming problems and the recent

TABLE 2.1 CLASSIFICATION OF STUDY PROBLEMS

TRANSPORTATION APPLICATIONS	GENERIC PROBLEMS	SOLUTION TECHNIQUES
FREIGHT FLOW MOVEMENT	NETWORK DESIGN Fixed Charge - Uncapacitated Fixed Charge - Capacitated No Bundles Bundles (MC Flows)	INTEGER PROG. BENDERS DECOMP. Shortest Route Subs. Min Cost Flow Subs. General LP Subs.
MULTIFLEET ROUTING	MULTICOMMODITY FLOW	INTEGER PROG. DANTZIG-WOLFE DECOMP. SUBGRADIENT OPTIM. SPECIAL PURPOSE CODE
URBAN GOODS MOVEMENT	VEHICLE ROUTING	HEURISTICS LIST PROCESSING (Efficient Data Manipulation)

surge in the analysis and implementation of heuristic algorithms combine to provide an attractive climate for considering the selected applications. We discuss the use of these techniques in sections 3, 4 and 5.

2.1 FREIGHT FLOW PROBLEM

The Freight Flow Problem is analogous in many ways to the generic set of problems in urban highway planning which have come to be called Traffic Assignment or Traffic Equilibrium Network Flow Problems. In the latter the flow process of thousands of passengers making individual origin destination trips is modeled.

Little attention, however, has been given to a similar generic set of problems in planning the flow of freight shipments. In these problems, a central manager is aggregating shipments in a set of terminals to reduce line haul costs, and may be using quite indirect routings in the supply network for any particular shipment. For a set of node-pair demands, the problem is to find an optimal pattern of freight flow considering both line haul and terminal costs. Thus, the optimal pattern will also indicate the optimum location and volume of flow for freight terminals on the network.

Examples of typical real-world applications of the Freight Flow problem are found in the consolidation of railroad network problems, in consideration of railroad network operations, in consideration of new roles for freight forwarders and "break-bulk" centers for trucking, and in the study of improved multimodal freight terminals. In the operation research literature these problems generally fall into one of three classifications:

(1) Fixed Charge Transportation Problem

(2) Network Design Problem

(3) Terminal Location Problem.

2.1.1 Fixed Charge Transportation Problem

The fixed charge transportation problem is physically characterized by a bipartite graph $G = G(N,A)$, where N is the set of nodes of the network and A the set of arcs. The node set N is split into two subsets, K and L, denoting respectively suppliers with supplies S_i and customers with requirements R_j. The problem is to determine the freight flow in each supplier-customer pair, with only direct shipments allowed. For each arc in the network we can associated a fixed charge for using it and a linear cost function for the total amount of freight routed along the arc. The fixed charge might correspond to the initial cost of a dispatching vehicle for carrying freight along the route. Our objective is to minimize the total cost of shipping the freight. Mathematically we have the statement (1.1)-(1.5).

2.1.2 Network Design Problem

The network design problem is a more general form of the fixed charge problem. Again we have a graph $G = (N,A)$ with flow requirements R_{ij} for O-D pair (i,j), but we no longer restrict G to be bipartite nor the shipments to be direct. In a sense set A represents only a "candidate" set of arcs which could be built to link the nodes. We still associate fixed charges and linear cost functions for each of the arcs in A.

MIXED INTEGER PROGRAMMING FORMULATION OF THE FIXED CHARGE TRANSPORTATION PROBLEM

$$\text{Minimize:} \quad \sum_{i \in K} \sum_{j \in L} c_{ij} x_{ij} + \sum_{i \in K} \sum_{j \in L} b_{ij} y_{ij} \tag{1.1}$$

subject to:

$$\sum_{j \in L} x_{ij} \leq S_i \qquad i \in K \tag{1.2}$$

$$\sum_{i \in K} x_{ij} \geq R_j \qquad j \in L \tag{1.3}$$

$$x_{ij} \leq [\min (S_i, R_j)] \, y_{ij} \qquad i \in K, \; j \in L \tag{1.4}$$

$$x_{ij} \geq 0 \qquad i \in K, \; j \in L \tag{1.5}$$

where

c_{ij} = the cost of sending one unit of flow between supplier i and customer j,

x_{ij} = the variable denoting the amount of flow between supplier i and customer j,

y_{ij} = a 0-1 variable that will be 1 if the amount of flow between supplier i and customer j is greater than zero,

b_{ij} = the fixed charge of using arc (i, j).

To illustrate the applicability of network synthesis to the Freight Flow Problem consider the simple 2 origin, 2 destination Fixed Charge Problem shown in Figure 2.1. Now consider the same problem with the option of routing freight through distribution center C, as shown in Figure 2.2. In this case we need also to consider the fixed charges and linear throughput costs of operating the distribution center. (The fixed charge for the distribution center could correspond to the cost of constructing such a facility. Possible cost advantages of using the distribution center may be reductions in the total number of vehicles required to transport freight.)

To model the situation of Figure 2.2, consider its "equivalent" expanded network representation in Figure 2.3, We have separated node C into two nodes, C_1 and C_2, and an arc (C_1,C_2) connecting them. With this construction we can associate all the costs of the distribution center, node C, with the arc (C_1,C_2). Therefore, in Figure 2.3 the expanded network has fixed charges and linear cost-flow functions associated with each arc in the network. If all arcs have infinite capacity then the problem we have described becomes a network design problem, which can be mathematically formulated as (2.1)-(2.5). Note that even a 50 node version of this formulation leads to a very large integer program. If we consider arcs joining all nodes, the formulation contains approximately 2500 integer variables y_{ij}, 6,250,000 continuous variables x_{ij}^{kl}, 2500 constraints of the form (2.2) and 6,250,000 constraints of the form (2.3). The last set of constraints can be reduced to about 2500 by a problem reformulation (see subsection 2.1.4). Nevertheless, even if most of the network configuration is set, the arc set A is small, say 200, and we are considering only a small portion of the network (of the possible y_{ij} variables) for design, the

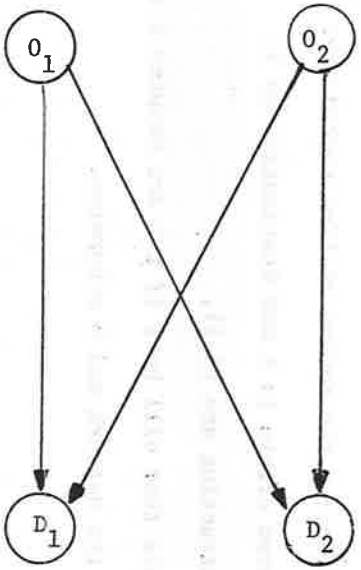


FIGURE 2.1
SIMPLE FREIGHT NETWORK

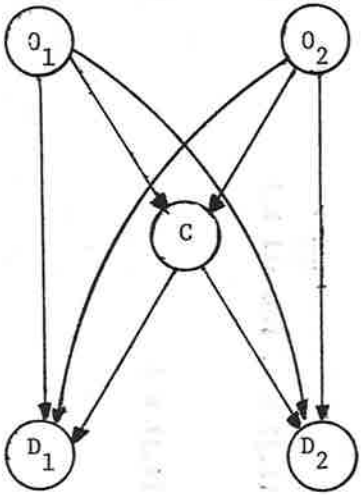


FIGURE 2.2
NETWORK WITH ONE DISTRIBUTION CENTER

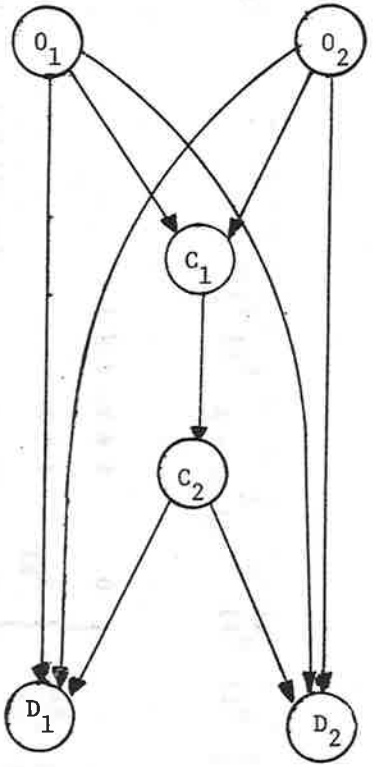


FIGURE 2.3
EXPANDED NETWORK REPRESENTATION

MIXED INTEGER PROGRAMMING PROBLEM FORMULATION OF A NETWORK DESIGN PROBLEM

Minimize:

$$\sum_{(i,j)\in A} \sum_{k} \sum_{l} c_{ij} x_{ij}^{kl} + \sum_{(i,j)\in A} b_{ij} y_{ij}, \tag{2.1}$$

subject to:

$$\sum_{j} x_{ij}^{kl} - \sum_{j} x_{ji}^{kl} = \begin{cases} 0 & i \neq k,\ i \neq l \\ R_{kl} & i = k \\ -R_{kl} & i = l \end{cases} \quad \text{for all } k,l \tag{2.2}$$

$$x_{ij}^{kl} \leq R_{kl}\, y_{ij} \qquad \text{for } (i,j) \in A, \text{ for all } k,l \tag{2.3}$$

$$x_{ij}^{kl} \geq 0 \qquad \text{for } (i,j) \in A, \text{ for all } k,l \tag{2.4}$$

$$y_{ij} = 0 \text{ or } 1 \qquad \text{for } (i,j) \in A \tag{2.5}$$

where:

i,j,k,l = node indices,

c_{ij} = cost of unit flow on arc (i,j),

x_{ij}^{kl} = a variable denoting the amount of flow routed over the arc (i,j) whose origin is k and destination is l,

b_{ij} = cost of constructing arc (i,j),

y_{ij} = a 0-1 variable that will be 1 if the arc between i and j is added to the network and 0 otherwise.

resulting integer program is quite large (about 500,000 continuous variables) when compared with computational capabilities of contemporary general purpose integer programming algorithms.

2.1.3 Terminal Location Problem

The third problem type is called the terminal or facility location problem. We seek to locate a fixed number of facilities, p, to service demand which is generated at an arbitrary subset of nodes of the network N (assume without loss of generality that it is the N set itself). The set of available loactions L for the construction of the facilities is also a subset of N. The associated costs are the facility construction cost at node j, b_j, and the total transportation costs of demand from a node i to a facility at j, c_{ij}. Since the capacity of the facilities is hypothesized to be unlimited, the nature of the problem dictates that the demand of a particular node i will never be fractionally satisfied at any particular facility, i.e., x_{ij}, the fraction of node i's demands that are satisfied at facility j is always 0 or 1. We thus have the integer programming formulation (3.1-3.5). An example of a special case of the terminal location problem, the p-median problem, is shown in Figure 2.4. All node demands (weights) are equal to one.

2.1.4 Modeling Extensions

There are a number of extensions to the fixed charge transportation problem, to the uncapacitated network design problem, and to the terminal location problem which permit broarder modeling capabilities. Let us

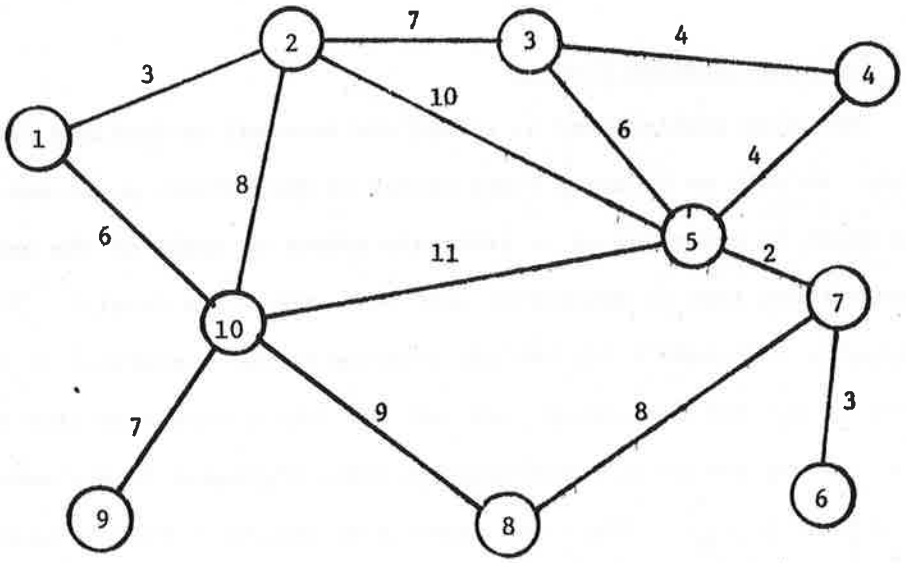


FIGURE 2.4 EXAMPLE OF p-MEDIAN PROBLEM (10 NODES)

MIXED INTEGER PROGRAMMING FORMULATION OF THE FACILITY LOCATION PROBLEM

Minimize:

$$\sum_{i \in N} \sum_{j \in L} c_{ij} x_{ij} + \sum_{j \in L} b_j y_j, \tag{3.1}$$

subject to:

$$\sum_{j \in L} x_{ij} = 1 \qquad i \in N \tag{3.2}$$

$$\sum_{j \in L} y_j \leq p, \tag{3.3}$$

$$x_{ij} \leq y_j \qquad i \in N,\ j \in L \tag{3.4}$$

$$x_{ij} \geq 0 \qquad i \in N,\ j \in L \tag{3.5}$$

where

N = the set of nodes in the problem,

L = the set of potential locations in the problem

y_j = a 0-1 variable that will be 1 if a facility is located at node j and 0 otherwise,

p = the maximum number of facilities that may be located.

For the p-median problem, all b_j are equal to zero. Also, the set N is identical to the set L.

For the simple plant location problem, constraint 3.3 is omitted.

illustrate certain possibilities in the context of network design.

In the capacitated network design model, each arc (i,j) that is constructed has a capacity bound K_{ij}. To accommodate this contingency, we use the constraints

$$\sum_k \sum_l x_{ij}^{kl} \leq K_{ij} y_{ij} \qquad \text{for } (i,j) \in A$$

instead of (2.3). Of course, by using the modelling device introduced previously, we can use constraints of this nature to model throughput capacity of a depot.

Note that the problem with these new constraints in place of (2.3) is equivalent to the formulation (2.1)-(2.5) whenever $K_{ij} \geq \sum_k \sum_l R_{kl}$. Then both the new constraints and (2.3) imply that $x_{ij}^{kl} = 0$ for all k and l whenever $y_{ij} = 0$; and when $y_{ij} = 1$, both the new constraint and (2.3) are redundant. Even though the new formulation appears to be preferred because it contains fewer constraints, several recent studies have shown that most algorithms perform better when applied to the given formulation (2.1)-(2.5). The fact that the linear programming relaxation (with $0 \leq y_i \leq 1$ in place of $y_i = 0$ or 1 for all i) of the given formulation contains more feasible solutions than does the linear programming relaxation of the modified formulation probably accounts for this behavior.

Another way to extend the network design model is by introducing constraints on network configuration. Typically, these can be modelled by linear constraints involving the design variables y_{ij}. For example, the constraint $y_{ij} \leq y_{rs}$ states that arc (i,j) is constructed only if arc (r,s) is. The constraint $y_{ij} + y_{rs} \leq 1$ implies that construction of both

arcs (i,j) and (r,s) is not permitted. Incorporating $y_{ij} + y_{rs} \geq 1$ states that at least one of arcs (i,j) and (r,s) must be constructed. Other configuration constraints can be included in a similar fashion.

None of the modelling variants considered thus far introduce congestion effects. We have described arc costs as linear functions of the arc flow and have defined a fixed capacity limit for arc flow. As arc flow approaches this limit, there may be a non-linear increasing cost due to congestion effects. These non-linear cost functions may readily be included in the model if they are known, perhaps in a step-wise linear fashion.

2.2 MULTIFLEET ROUTING PROBLEM

The Multifleet Routing problem is concerned with optimally routing fleets of different types of vehicles within a given timetable, or "schedule-map", of alternative possible services.

These problems occur in two areas of transportation systems analysis. In the setting of airline systems, these areas are: 1) in schedule planning, where a timetable is being constructed perhaps for next year, or for several years ahead to assist in investment decisions for aircraft and airports; 2) in real-time operations control where today's airline schedule needs revision due to aircraft mechanical failures or bad weather, and where decisions must be made about cancelling flights in a way that minimizes revenue loss or passenger disruption. There are similar applications in any scheduled system of passenger transportation vehicles such as passenger train systems, subway systems or intercity bus systems if vehicles of different size, speed or operating cost are being used.

The network in this case is the schedule map, an example of which is given for an airline application in Figure 2.5. In the map, the vertical dimension represents time, and the period of time measured in this dimension is called one time cycle. We then describe the map as a directed graph $G = (N,A)$, where N is the set of nodes representing specific times at specific locations, and A the set of arcs joining the nodes. More precisely the arc set A is the disjoint union of three sets:

1) Service arcs, A_s, which identify the potential services that can be flown, specified by a departure time t_p, from some origin p, and an arrival time t_q at destination q.

2) Ground arcs, A_g, which connect two nodes at the same station and describe the possibility of an aircraft staying on the ground in some interval of time.

3) Cycle arcs, A_c, which occur because the schedule map is periodic. These are dummy arcs which are required to recycle the fleet back to the beginning of a period.

The Single Fleet Routing Problem (SFRP) is easily solved for large networks. However, it has limited use in situations where there are vehicles of varying size and operating cost (such as in the case of domestic airlines). The Multifleet Routing Problem (MFRP) is considerably more difficult to solve. In the MFRP, many (but not necessarily all) of the services may be flown by different aircraft subject to limitations of aircraft range and seating capacity. Thus we need to establish, for each type of aircraft, a separate schedule map upon which a fleet of vehicles of that type is routed. The service value will vary due to the different costs of operating large or small aircraft, and it is not clear that a

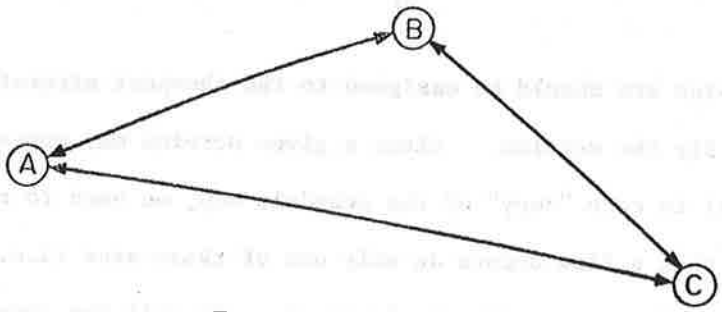


FIGURE 2.5A ROUTE MAP

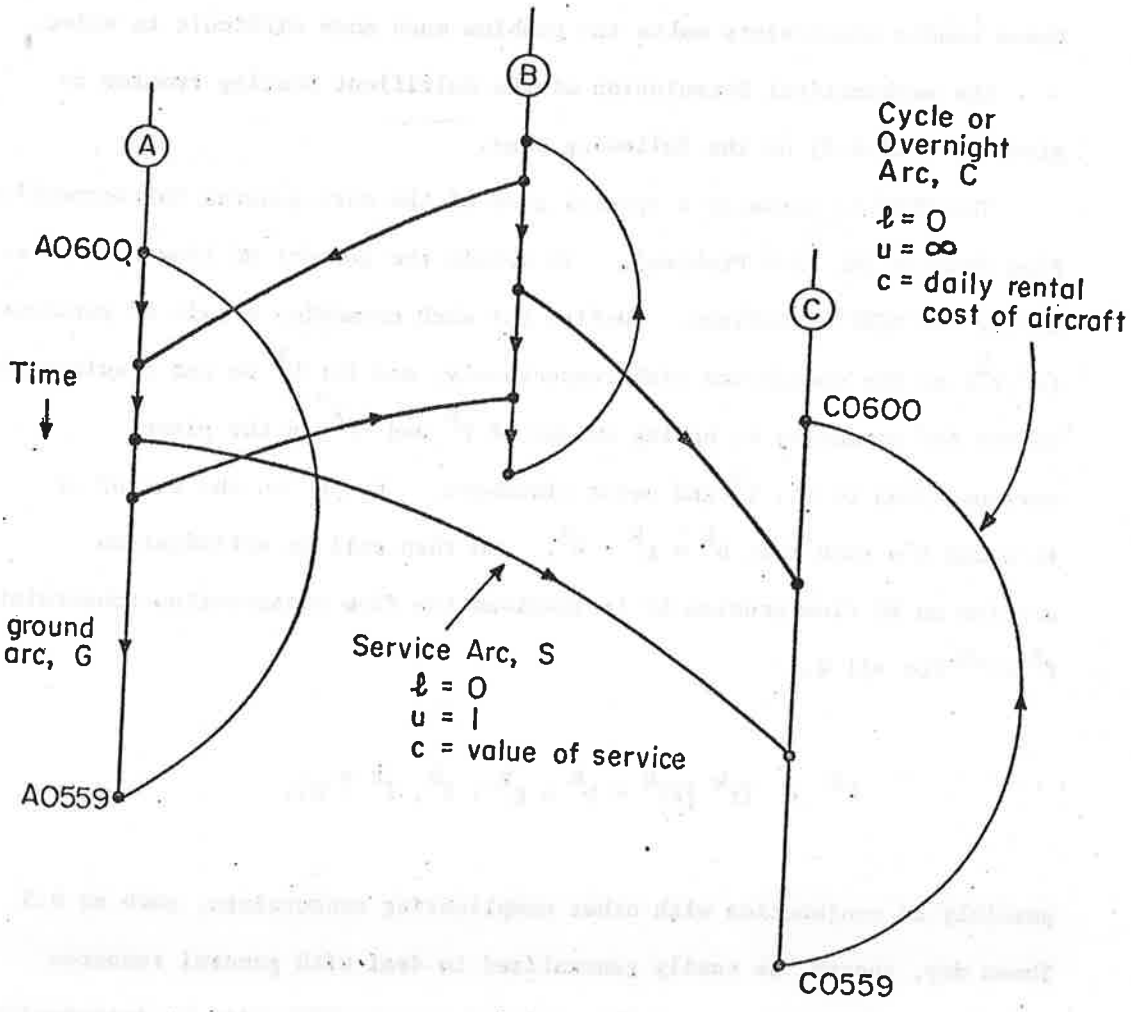


FIGURE 2.5B SCHEDULE MAP

particular service arc should be assigned to the cheapest aircraft which is eligible to fly the service. Since a given service may appear (with different value) in each "copy" of the schedule map, we need to constrain the flows such that a flow occurs in only one of these arcs (i.e., only one type of aircraft may provide the service). We call the constraints (see 4.5) associated with this, "bundle" constraints. The addition of these bundle constraints makes the problem much more difficult to solve.

The mathematical formulation of the Multifleet Routing Problem is given by (4.1-4.5) on the following page.

The MFRP is actually a special case of the more general Multicommodity Flow Problem (MC Flow Problem). To obtain the general MC Flow Problem we modify the MFRP as follows. Define for each commodity a pair of vertices (s^k,t^k) as the source and sink respectively, and let b^k be the requirements vector for commodity k, having values of F^k and $-F^k$ in the places corresponding to s^k, t^k and zeros elsewhere. Let g^k be the vector of +1's and 0's such that $b^k = g^k \cdot F^k$. We then call an optimization problem an MC flow problem if it involves the flow conservation constraints $f^k \in F^k$ for all k,

$$F^k = \{f^k \mid Ef^k = b^k = g^k \cdot F^k,\ f^k \geq 0\},$$

possibly in conjunction with other complicating constraints, such as 4.5. These may, though, be easily generalized to deal with general resource constraints or conversion of flow units into capacity units by introducing matrices D^k where D^k_{ra} = amount of resource r per unit flow of commodity k

A MATHEMATICAL STATEMENT OF THE MULTIFLEET ROUTING PROBLEM

Maximize:

$$Z = \sum_{k=1}^{K} c^k \cdot f^k \qquad (4.1)$$

$$\ell^k \leq f^k \leq u^k \qquad \text{Arc Flow Constraints for each schedule map k,} \qquad (4.2)$$

$$\sum_{a \in A_c} f_a^k \leq F^k \qquad \text{Fleet Size Constraints for each schedule map k,} \qquad (4.3)$$

$$E^k \cdot f^k = 0 \qquad \text{Circulation Flow Conservation Constraints,} \qquad (4.4)$$

$$\sum_{k=1}^{K} f_a^k \leq 1 \qquad \text{Service Bundle Constraints for each } a \in A_s, \qquad (4.5)$$

$$f^k \geq 0$$

where:

f^k = vector of integer circulation flows in arcs of schedule map k, with component f_a^k for arc a.

c^k = arc flow costs of schedule map k, with component c_a^k for arc a.

F^k = number of available vehicles, type k.

ℓ^k = lower bounds on arc flows in schedule map k.

u^k = upper bounds on arc flows in schedule map k.

$E^k = \{e_a^k\}$ = the node arc incidence matrix of schedule map k.

K = number of different types of vehicles in fleet.

on arc a. Thus a generalized resource-constraint MC flow problem can be defined as:

$$\text{Min } z = \sum_{k=1}^{K} c^k \cdot f^k \tag{5.1}$$

$$f^k \in F^k \qquad k = 1,2,\ldots,K \tag{5.2}$$

$$\sum_{k=1}^{K} D^k f^k \leq d. \tag{5.3}$$

Obviously, for $D^k = I$ for all k, and $b^k = 0$ for all k, we recover the original formulation (4.1)-(4.5)†. This problem has a block angular structure which the detached coefficient form below makes conspicuous:

$$\begin{vmatrix} E & & & \\ & E & & \\ & & \ddots & \\ & & & E \\ II & \cdots & & II \end{vmatrix} \cdot \begin{vmatrix} f^1 \\ f^2 \\ \vdots \\ f^K \\ s \end{vmatrix} = \begin{vmatrix} b^1 \\ b^2 \\ \vdots \\ b^K \\ d \end{vmatrix} . \tag{6}$$

The structure makes the problem suitable for the application of both price-directive and resource-directive methods. The basic idea is to do away with the complicating constraints (5.3) so as to decompose the MC problem into K single commodity subproblems where the network structure allows efficient solution techniques. Section 3 gives the details of the algorithms involved.

---

†In this case, the flow constraint set F^k will also include the flow bounds (4.2). Also, note that we have written problem (5.1) - (5.3) in minimization rather than maximization form. This agrees with our later description of multicommodity flow algorithms.

2.3 VEHICLE ROUTING PROBLEM

The allocation and routing of vehicles for the purpose of collecting and delivering goods and services on a regular basis -- the Vehicle Routing Problem (VRP) -- is an essential element of any logistics system. Common examples include newspaper delivery, schoolbus routing, municipal waste collection, fuel oil delivery, and truck dispatching in any number of industries. The system may involve a single depot or multiple depots; the objectives may be aimed at cost minimization (distribution costs, and vehicle or depot acquisition costs) or service improvement (increasing distribution capacities, reducing distribution time, and related network design issues). Constraints may be imposed upon:

i) the numbers, possible locations, and capacities of depots

ii) the numbers of vehicles and capacities

iii) delivery point demand and service constraints

iv) the routing structure -- limit on stops/vehicle route or length of route in time or distribution

v) operator scheduling and assignments.

As we will later see, several approaches to the vehicle routing problem entail solving a number of smaller traveling salesman problems (TSP). In other words, algorithmically the VRP decomposes into the ubiquitous TSP.

In constructing the mathematical formulation of the more general VRP, we start with the simpler TSP, where we seek to form a tour of a set of n nodes, beginning and ending at the origin, node 1, which gives the minimum total distance or cost. A more general version of this is the Multiple Traveling Salesman Problem (MTSP) that comes closer to accommodating many

real world problems. Here there is a need to decide upon the assignment of M salesmen (vehicles) to cover the nodes of the network such that every node (except the origin) is visited exactly once by exactly one salesman, again minimizing the total distance traveled by all M salesmen. An MTSP formulation is displayed below:

$$\text{Minimize:} \quad \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} x_{ij} \tag{7.1}$$

$$\text{subject to:} \quad \sum_{i=1}^{n} x_{ij} = b_j = \begin{cases} M \text{ if } j=1 \\ 1 \text{ if } j=2, 3, \ldots, n \end{cases} \tag{7.2}$$

$$\sum_{j=1}^{n} x_{ij} = a_i = \begin{cases} M \text{ if } i=1 \\ 1 \text{ if } i=2, 3, \ldots, n \end{cases} \tag{7.3}$$

$$X = (x_{ij}) \in S \tag{7.4}$$

$$x_{ij} = 0 \text{ or } 1 \qquad (i = 1, \ldots, n; \; j = 1, \ldots, n), \tag{7.5}$$

for any choice of the set S that breaks subtours which do not include the origin and where

n = the number of nodes in the network

d_{ij} = the distance or cost of arc (i,j).

In this formulation, $x_{ij} = 1$ if a salesman transverses arc (i,j) and $x_{ij} = 0$ otherwise. The set S is selected to prohibit subtour solutions satisfying the assignment constraints (7.2), (7.3), and (7.5). Several alternatives have been proposed for S including,

$$S = \{(x_{ij}): \sum_{i\in Q} \sum_{j\notin Q} x_{ij} \geq 1 \quad \text{for every nonempty proper subset Q of the set of nodes N}\}. \tag{8}$$

$$S = \{(x_{ij}): \sum_{i\in Q} \sum_{j\in Q} x_{ij} \leq |Q| - 1 \quad \text{for every nonempty subset Q of 2, 3, ..., n}\}. \tag{9}$$

$$S = \{(x_{ij}): y_i - y_j + nx_{ij} \leq n-1 \quad \text{for } 2 \leq i \neq j \leq n \text{ for some real numbers } y_i\}. \tag{10}$$

From the MTSP it is possible to create vehicle routing problems incorporating various real world operating constraints. Consider first a fleet of NV (number of) vehicles of various capacities P_k and various total elapsed route time constraints T_k (for instance, a newspaper delivery truck may be restricted from spending more than one hour on a tour in order that the maximum time interval from press to street be made as short as possible). The following problem will then be referred to as the generic vehicle routing problem:

$$\text{Minimize:} \quad \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{NV} d_{ij} x_{ij}^{k} \tag{11.1}$$

$$\text{subject to:} \quad \sum_{i=1}^{n} \sum_{k=1}^{NV} x_{ij}^{k} = 1 (j=2,\ldots,n), \tag{11.2}$$

$$\sum_{j=1}^{n} \sum_{k=1}^{NV} x_{ij}^{k} = 1 \quad (i = 2, \ldots, n), \tag{11.3}$$

$$\sum_{i=1}^{n} x_{ip}^{k} - \sum_{j=1}^{n} x_{pj}^{k} = 0 \qquad (k = 1, \ldots, NV; \; p = 1, \ldots, n) \tag{11.4}$$

$$\sum_{i=1}^{n} Q_i \left(\sum_{j=1}^{n} x_{ij}^{k} \right) \leq P_k \qquad (k = 1, \ldots, NV), \tag{11.5}$$

$$\sum_{i=1}^{n} t_i^k \sum_{j=1}^{n} x_{ij}^{k} + \sum_{i=1}^{n} \sum_{j=1}^{n} t_{ij}^{k} x_{ij}^{k} \leq T_k \qquad (k = 1, \ldots, NV), \tag{11.6}$$

$$\sum_{j=2}^{n} x_{ij}^{k} \leq 1 \qquad (k = 1, \ldots, NV), \tag{11.7}$$

$$\sum_{i=2}^{n} x_{i1}^{k} \leq 1 \qquad (k = 1, \ldots, NV) \tag{11.8}$$

$$X \in S \tag{11.9}$$

$$x_{ij}^{k} = 0 \text{ or } 1 \qquad \text{for all } i, j, k. \tag{11.10}$$

where

n = number of nodes

NV = number of vehicles

P_k = capacity of vehicle k

T_k = maximum time allowed for route of vehicle k

Q_i = demand at node i ($Q_1 = 0$)

t_i^k = time required for vehicle k to deliver or collect at node i ($t_1 = 0$)

t_{ij}^k = travel time for vehicle k from node i to node j ($t_{ii} = \infty$)

d_{ij} = distance from node i to node j

x_{ij}^k = $\begin{cases} 1 \text{ if arc } (i,j) \text{ is traversed by vehicle } k \\ 0 \text{ otherwise} \end{cases}$

X = matrix with components $x_{ij} \equiv \sum_{k=1}^{NV} x_{ij}^k$, specifying connections regardless of vehicle type.

Note that this formulation contains approximately 800,000 0-1 variables and approximately 44,500 constraints (using (10) for S) for a 20-vehicle 200-node problem. We shall report on computational experience with 600-node, 20-vehicle problems in section 5.

We can easily add the investigation of tradeoffs between routing and acquisition costs to this problem by considering the fixed acquisition costs of each vehicle type. Also,it is possible to generalize to the multicommodity case where several different types of products must be routed simultaneously over a network in order to satisfy demands at delivery points for the various products. With somewhat greater difficulty we can also incorporate timing restrictions on vehicle dispatching: by specifying earliest delivery times and absolute delivery deadlines. Last of all, the integer programming formulation of the vehicle routing problem is altered in minor ways to incorporate multiple depots. Letting nodes 1, 2, ..., M denote the depots, we obtain the formulation by changing the index in constraints (11.2) and (11.3) to (j = M + 1, ..., n), by changing constraints (11.7) and (11.8) to

$$\sum_{i=1}^{M} \sum_{j=M+1}^{n} x_{ij}^k \leq 1 \qquad (k = 1, \ldots, NV) \qquad (11.7')$$

$$\sum_{p=1}^{M} \sum_{i=M+1}^{n} x_{ip}^{k} \leq 1 \qquad (k = 1, \ldots, NV), \qquad (11.8'),$$

and by redefining our previous choices for the subtour breaking set S to be:

$$S = \{(x_{ij}): \sum_{i \in Q} \sum_{j \notin Q} x_{ij} \geq 1 \quad \text{for every proper subset Q of N containing nodes 1, 2, ..., M}\}; \qquad (8')$$

$$S = \{(x_{ij}): \sum_{i \in Q} \sum_{j \in Q} x_{ij} \leq |Q| - 1 \quad \text{for every nonempty subset Q of } \{M+1, M+2, \ldots, n\}\}; \qquad (9')$$

$$S = \{(x_{ij}): y_i - y_j + n\, x_{ij} \leq n - 1 \quad \text{for } M+1 \leq i \neq j \leq n \text{ for some real numbers } y_i\} \qquad (10')$$

Following this introduction to the network models and their applications, the next three sections will give a detailed description of solution methodology and computational experience.

3. MULTIFLEET ROUTING PROBLEMS

3.1 METHODOLOGICAL APPROACHES

In this section we describe three algorithms for the multifleet routing problem: price-directive decomposition, resource-directive decomposition, and a special purpose heuristic algorithm. In section 3.2 we discuss implementation details for these algorithms and in sections 3.3 and 3.4 we summarize computational experience with these algorithms for multifleet routing and other multicommodity flow problems.

3.1.1. Dantzig-Wolfe Decomposition

We have described the price-directive or Dantzig-Wolfe decomposition in the survey paper [3-3]. Here, before discussing our implementation of the algorithm, we review its use for the multi-commodity flow problem.

Recall from section 2.2 that the linear multicommodity flow problem can be formulated with flow variables f^k for each of several commodities $k = 1, 2, \ldots, K$ as:

$$\text{Min} \sum_{k=1}^{K} c^k \cdot f^k \tag{1.1}$$

subject to:

$$f^k \in F^k \tag{1.2}$$

$$D \cdot f = \sum_{k=1}^{K} D f^k \leq d \tag{1.3}$$

where

$$F^k = \{f^k | Ef^k = b^k = g^k F^k, \quad f^k \geq 0\}. \tag{2}$$

Here F^k is the required flow value for commodity k from its origin s^k to its destination t^k. Matrix D in (1.3) transforms the flow vector f to a vector Df giving the flows on capacitated arcs only. Thus if the first m_0 arcs are capacitated, then D will be the $m_0 \times m$ matrix [I,0].

Letting $Q_t^k = \{f^{k,i} | i=1, \ldots, I_t^k\}$ be the set of extreme points of F^k on hand at the beginning of iteration t, we may express a tentative solution to (1.1) - (1.3) as:

$$f^k = \sum_{i=1}^{I_t^k} \lambda_i^k \, f^{k,i} \tag{3.1}$$

$$\sum_i \lambda_i^k = 1, \quad \lambda_i^k \geq 0. \tag{3.2}$$

Substituting for f^k in (1.1) - (1.3) and treating the λ_i^k's as decision variables, we obtain the t^{th} "restricted master problem":

$$\text{Min} \sum_{k=1}^{K} \sum_{i=1}^{I_t^k} (c^k \cdot f^{k,i}) \, \lambda_i^k \tag{4.1}$$

subject to:

$$\sum_k \sum_i (Df^{k,i}) \cdot \lambda_i^k \leq d \tag{4.2}$$

$$\sum_i \lambda_i^k = 1 \quad \forall k \tag{4.3}$$

$$\lambda_i^k \geq 0 \quad \forall k, \forall i. \tag{4.4}$$

In the above linear program we have restricted our choice of f^k by considering only a subset Q_t^k of the extreme points of F^k. This set, however, can be enlarged if it is profitable to consider other extreme points. Let π_t and σ_t^k be optimal dual variables associated with (4.2) and (4.3). The criterion for entering a new column associated with an extreme point f^k of F^k is that it has negative reduced cost, i.e.,

$$c^k \cdot f^k - \pi_t D f^k - \sigma_t^k < 0. \tag{5}$$

Thus to find the most favored column (in terms of reduced cost) for entry, we solve the subproblem

$$w^k = \text{Minimum } (c^k - \pi_t D) \cdot f^k \tag{6.1}$$

subject to:

$$Ef^k = b^k \tag{6.2}$$

$$f^k \geq 0, \tag{6.3}$$

for commodity k. The subproblem requires sending the required flow of F^k units of commodity k from s^k to t^k in such a way as to minimize the routing costs with respect to the arc costs given by $(c^k - \pi_t D)$. It may be readily solved by sending all F^k units of flow along the single minimum cost

chain from s^k to t^k. Thus for each commodity k, a shortest chain problem is solved as the subproblem.

If all columns price out nonnegatively for all commodities, then the optimal solution to the current restricted master is also optimal for (1.1) - (1.3). Thus our optimality criterion is

$$\underset{k}{\text{Minimum}} \{ (c^k - \pi_t D) \cdot f^k - \sigma_t^k \} \geq 0 \tag{7.1}$$

or equivalently:

$$w^k \geq \sigma_t^k \quad \forall k. \tag{7.2}$$

If (7.1) is not satisfied then (5) holds for at least one k, meaning that the set $S_t = \{k \mid w^k < \sigma_t^k\}$ is nonempty. For each k in S_t, we augment the set Q_t of extreme points on hand by the extreme point generated by solving (6.1) - (6.3), i.e., add f^k to Q_t^k to form Q_{t+1}^k. Obviously, for $k \notin S_t$ no new extreme point is added to Q_t^k and $Q_{t+1}^k = Q_t^k$.

To complete our description of the algorithm, we should specify the initialization procedure for the master problem. The form given for the master problem in (4.1) - (4.4) does not ensure its feasibility. Since a flow of $\lambda_i^k F^k$ units of commodity k is sent along the chain corresponding to $f^{k,i}$, it is easy to see how the capacity constraints in (4.2) would limit this flow, resulting in the λ_i^k's to sum up to a value less than one. This problem is most pronounced at the initial stages where the number I_t^k of known flow patterns for the commodities k=1, ..., K are not

sufficiently large to allow us to distribute the flow in a manner to avoid the interference of capacity constraints. Thus it is necessary to add artificial variables λ_a^k to ensure feasibility. The restricted master will then read:

$$\text{Min} \sum_{k=1}^{K} \sum_{i} (c^k \cdot f^{k,i}) \lambda_i^k + \sum_{k=1}^{K} c_a^k \cdot \lambda_a^k \tag{8.1}$$

subject to:

$$\sum_k \sum_i (Df^{k,i}) \cdot \lambda^k \leq d \tag{8.2}$$

$$\sum_k \lambda_i^k + \lambda_a^k = 1 \tag{8.3}$$

$$\lambda_i^k, \ \lambda_a^k \geq 0. \tag{8.4}$$

To initialize Q_t^k for t=1 we solve (6.1) - (6.3) for each commodity with respect to the real arc costs c^k, i.e., with $\pi_0 = 0$ in (6.1) to obtain $f^{k,1}$ and we set $Q_1^k = \{f^{k,1}\}$, $I_1^k = 1$. This choice is motivated by the thought that these "real" shortest chains are, a priori, the most attractive candidates for carrying the flow. Consequently, they might carry a good portion of the flow in the optimal solution, so we should enter them as soon as possible. (This conjecture was borne out by the computational results.)

The costs c_a^k for the artificial columns are chosen to be large positive numbers so as to make these columns unattractive and drive the

λ_a^k's out of the basis. To achieve this, the costs c_a^k must be appreciably larger than the average chain costs $c^k \cdot f^{k,i}$ (averaged over all possible chains from s^k to t^k). On the other hand, as the dual variables associated with (8.1) - (8.4) will be of the same order of magnitude as the c_a^k's, we should avoid using very large values of c_a^k. In practice we have chosen c_a^k to be about 10-20 times the chain cost $c^k \cdot f^{k,1}$. Essentially the procedure given above is the "Big-M" method for obtaining a feasible solution. Tomlin [3-30] describes an alternative approach, which involves a "Phase I" procedure for this problem. Our choice may be defended by the computational observation that for cases where capacity constraints (8.2) are nonbinding, we can obtain the optimal solution from the first master problem. Further remarks on this as well as other implementation issues will be made in the subsequent two sections.

Specialization for the Multifleet Routing Problem

To put the Multifleet Routing Problem, in section 2 (4.1 - 4.5), in a form compatible with our code for the general Multicommodity Flow Problem, we proceed as follows: We remove all cycle (or overnight) arcs and instead attach a supersource and a supersink where all aircraft originate and terminate respectively. These act as the common OD pair (s,t) for all commodities k. Thus the topmost node of each vertical line (city) in the schedule map is joined to the supersource s and the lowest node to t. The resulting network is then acyclic. The ownership costs c_o^k attached to cycle arcs $a \in A_c$ may now be attached to the arcs emanating from s. The arcs incident to t are given a cost of zero. The flow value

F^k denotes the total number of vessels of type k utilized. The problem may then be stated as one of minimizing losses:

$$\text{Min } z = \sum_{k=1}^{K} \left(c_o^k \cdot F^k + \sum_{a \in A_g} c_a^k \cdot f_a^k - \sum_{a \in A_s} c_a^k \cdot f_a^k \right) \tag{9.1}$$

subject to:

$$Ef^k - g^k F^k = 0 \quad \forall k \tag{9.2}$$

$$f^k, F^k \geq 0 \quad \forall k \tag{9.3}$$

$$\sum_{k=1}^{K} f_a^k \leq 1 \quad \forall a \in A_s. \tag{9.4}$$

Notice that we have chosen the lower bound $\ell = 0$ and have imposed no upper bounds. We also note that the set

$$P^k \triangleq \{(f^k, F^k) \mid Ef^k = g^k F^k, f^k \geq 0, F^k \geq 0\}, \tag{10}$$

is a cone. Any feasible flow satisfying (9.2-9.3) may be expressed as

$$f^k = \sum_{i=1}^{I^k} \rho_i^k f^{k,i}$$

with $(f^{k,i}, 1)$ an extreme ray of P^k. Thus our master problem may be stated as

$$\text{Min} \sum_{k=1}^{K} \sum_{i=1}^{I^k} c^{ki} \cdot \rho_i^k \tag{11.1}$$

subject to:

$$\sum_{k=1}^{K} \sum_{i=1}^{I^k} \rho_i^k f_a^{k,i} \leq 1 \qquad a \in A_s \tag{11.2}$$

$$\rho_i^k \geq 0 \qquad \forall i, \forall k \tag{11.3}$$

where

$$c^{k,i} = c^k \cdot f^{k,i} = c_\Omega^k + \sum_{a \in A_g} c_a^k \cdot f_a^{k,i} - \sum_{a \in A_s} c_a^k \cdot f_a^{k,i} \tag{12}$$

Note that we have no convexity constraints (as in 4.3) since we are decomposing over a cone [3-19].

As before, attach dual variables π to (11.2). The criterion for entering a column into the restricted master is

$$c^{k,i} - \pi D \, f^{k,i} < 0$$

where D is a matrix transforming $f^{k,i}$ to a subvector of flows on A_s only. Thus the subproblems are, as before, shortest path problems with respect to the arc costs $(c^k - \pi D)$. There are, however, two minor changes: (a) the arc costs are not all nonnegative and some arcs have negative lengths, (b) the network structure is acyclic. We know that condition (a) causes no problems as long as no negative cycles exist, which condition (b) insures, so that the Bellman-Moore shortest route algorithm BELL, which we describe in subsection 3.2.1, may be applied with impunity.

Further the shortest path algorithm may be accelerated by taking the acyclic structure into account. We have not adopted this course due to the small percentage of time our code expends on the subproblems as opposed to the master problem.

3.1.2 Resource-Directive Decomposition

Resource-directive methods (or decomposition by right-hand-side allocation) were discussed in sections 2.3 and 4.2 of the Multicommodity Flow Survey [3-3]. Here we will describe an algorithm based on the idea of subgradient optimization as developed by Held, Wolfe, and Crowder [3-16]. We start by showing how the minimum cost flow problem can be put into a form amenable to subgradient optimization. Consider again the problem:

$$v_1^* = \text{Min} \sum_{k=1}^{K} c^k \cdot f^k \tag{13.1}$$

subject to:

$$f^k \in F^k \qquad k = 1,2,\ldots,K \tag{13.2}$$

$$\sum_{k=1}^{K} f^k \leq d. \tag{13.3}$$

The complicating constraints in (13.3) may be avoided if one knew how much of the scarce capacity resource each commodity requires in the optimal solution. The problem (13.1) - (13.3) can then be decomposed into K separate single commodity capacitated flow problems. This motivates allocating capacities $y^1, \ldots, y^K$ to the commodities and searching for

an optimal allocation of d into the vectors y^k for k=1, ..., K. One may thus reformulate (13.1) - (13.3) as the equivalent problem:

$$v_1^* = \text{Min} \{ \sum_{k=1}^{K} v_1^k (y^k) \} \tag{14.1}$$

subject to:

$$\sum_{k=1}^{K} y^k = d \tag{14.2}$$

$$y^k \geq 0 \quad \forall k \tag{14.3}$$

where

$$v_1^k (y^k) = \text{Min } c^k \cdot f^k \tag{15.1}$$

subject to:

$$Ef^k - g^k F_o^k = 0 \tag{15.2}$$

$$f^k \leq y^k \tag{15.3}$$

$$f^k \geq 0 \tag{15.4}$$

The single-commodity subproblem (15.1) - (15.4) minimizes routing costs given a capacity allocation y^k for commodity k. The "master problem" (14.1) - (14.3) searches over all feasible partitions of the

capacity vector d into allocations $y^1, \ldots, y^K$. The constraint (14.2) simply ensures that on any arc "a" the sum of capacities allocated to different commodities equals the available capacity d_a. Obviously by comparing (15.3) and (15.4) we wish to limit the capacity allocations to nonnegative values as stated in (14.3).

Note that the constraints (15.2) and (15.4) merely restate (13.2) for the required flow value F_o^k and that (15.3) describes a capacity constraint on the flow of commodity k alone. The attractiveness of resource-directive decomposition for (13.1) - (13.3) lies in the fact that each subproblem (15.1) - (15.4) is a single commodity minimum cost flow for any given $y^k \geq 0$. Such a problem may be solved by an efficient Out-of-Kilter algorithm [3-1, 3-2, 3-6] or by any of the recent implementations of simplex-based primal network flow codes [3-11, 3-12, 3-25]. (For a survey of the recent advances in solving minimum cost network flows, see section III of [3-21].)

We now proceed to define a slightly modified but equivalent subproblem:

$$v^k(y^k) = \text{Max } (c_o^k F^k - c^k f^k) \tag{16.1}$$

subject to:

$$Ef^k - g^k F^k = 0 \tag{16.2}$$

$$f^k \leq y^k \tag{16.3}$$

$$F^k \leq F_o^k \tag{16.4}$$

$$f^k,\ F^k \geq 0. \tag{16.5}$$

Here F^k, the total flow of commodity k, is treated as a decision variable. By attaching large costs c_o^k to F^k, this variable is encouraged to assume its upper bound F_o^k, which is the required flow value in (15.2), provided that the capacity constraints (16.3) do not preclude this. Thus, if problem (15.1) - (15.4) is feasible, meaning that there exists a feasible flow with flow value F_o^k, then (15.1) - (15.4) and (16.1) - (16.5) have the same optimal solution $\bar{f}^k$. Moreover, in that case, at optimality we have

$$v^k(y^k) = c_o^k \bar{F}^k - c^k \bar{f}^k = c_o^k F_o^k - v_1^k(y^k). \tag{17}$$

This manipulation shows that our two subproblem formulations are equivalent. Intuitively, expressions (16.1) - (16.5) may be thought of as first striving for maximum flow $\bar{F}^k \leq F_o^k$ through the network with arc capacities y^k; and then searching for the flow $\bar{f}^k$ with optimal routing costs among all flows with flow value $\bar{F}^k$. The advantage of (16.1) - (16.5) over (15.1) - (15.4) is that the former always possesses a feasible solution. In fact, $(f^k, F^k) = (\underline{0}, 0)$ is always feasible in (16.1) - (16.5), whereas infeasibility might easily occur in (15.1) - (15.4) for a "small" capacity allocation y^k yielding a maximum flow value smaller than F_o^k.

To continue the development of the subgradient algorithm, define

$$y = (y^1, \ldots, y^K)$$

$$v(y) = \sum_{k=1}^{K} v^k(y^k) \qquad (18)$$

and

$$S = \{y = (y^1, \ldots, y^K) \mid \sum_{k=1}^{K} y^k = d,\ y^k \geq 0 \ \forall k\}. \qquad (19)$$

We may restate the master problem as:

$$v^* = \text{Max } v(y) \qquad (20.1)$$

subject to:

$$y \in S \qquad (20.2)$$

The dual to (16.1) - (16.5) is

$$\text{Minimize } (\lambda^k \cdot y^k + \gamma_o^k F_o^k) \qquad (21.1)$$

subject to:

$$\pi^k E^k - \gamma^k \leq c^k \qquad (21.2)$$

$$\pi^k \cdot g^k + \gamma_o^k \geq c_o^k \qquad (21.3)$$

$$\gamma^k, \gamma_o^k \geq 0 \qquad (21.4)$$

(π^k free)

where π^k, γ^k, and γ_o^k are dual variables associated with (16.2), (16.3), and (16.4) respectively. By strong duality, at optimality we have:

$$v^k(y^k) = \gamma^{-k} \cdot y^k + \gamma_o^{-k} F_o^k \tag{22}$$

so that

$$v(y) = \text{Min} \{ \sum_{k=1}^{K} \bar{\gamma}^k \cdot y^k + \bar{\gamma}_o^k F_o^k \} \tag{23}$$

subject to the constraints (21.2) - (21.4) for all k.

For each k the above minimization will yield an extreme point of the dual feasible region defined by (21.2) - (21.4). Thus in (23) we may minimize over the set of all extreme points of (21.2) - (21.4) for all k. Define:

$$\gamma^p = (\gamma^{1,p}, \ldots, \gamma^{K,p}), c_p = \sum_{k=1}^{K} \gamma_o^{k,p} F_o^k \tag{24}$$

where for each k, $(\gamma^{k,p}, \gamma_o^{k,p})$ corresponds to an extreme point solution of (21.2) - (21.4) and the index p enumerates the <u>finite</u> set of all possible groupings of subproblem dual extreme points. Thus problem (23) may be stated as the <u>finite</u> minimization problem

$$v(y) = \text{Minimum} \{ c_p + y \cdot \gamma^p \mid p=1, \ldots, n_p \}. \tag{25}$$

The master problem (20.1) - (20.2) with the characterization of v(y) given in (25) is exactly of the form assumed by [3-16] for the subgradient optimization algorithm. We thus arrive at the following algorithm:

Subgradient Algorithm for the Multicommodity Flow Problem

1) Set i = 1. Allocate initial arc capacities to individual commodities, that is, define the vector:

$$y^0 = (y^{1,0}, \ldots, y^{K,0}) \in S. \tag{26}$$

2) Solve the single commodity minimum cost flow problem (16.1) - (16.5) in cycle for k=1, ..., K using $y^{k,i}$ as the right-hand side in (16.3). Obtain dual variables $\gamma^k(y^k_i)$ from the optimal solution.

3) Obtain the step size t_i and form the intermediate vector $\tilde{y}^i = (\tilde{y}^{1,i}, \ldots, \tilde{y}^{K,i})$

$$\tilde{y}^i = y^i + t_i\gamma(y^i). \tag{27}$$

4) Project the intermediate vector on the convex set S to obtain the next capacity allocation vector

$$y^{i+1} = P_S(\tilde{y}^i). \tag{28}$$

5) Stop if termination criterion is met. Otherwise let i ← i+1 and go to step two.

The technical report [3-4] discusses connections between this algorithm and Benders Method (which is described in section 4 of this report). That report justifies using the formulation (16.1) - (16.5) to avoid subproblem infeasibility but does not otherwise affect the implementation issues this report focuses on. Accordingly this material will not be reproduced here.

We conclude the subsection with brief comments about the form of step sizes t_i and the mechanism of performing the projection in step 4. This will serve as a background to our discussion of computational issues in subsequent sections.

Held, Wolfe, and Crowder [3-16] suggest using a sequence of positive numbers λ_i with

$$0 < \epsilon < \lambda_i \leq 2 \tag{29}$$

to define a sequence of step sizes of the form

$$t_i = \lambda_i \frac{v - v(y^i)}{||\gamma(y^i)||^2} \tag{30}$$

where v is an underestimate of the optimal value v^* (i.e., $v < v^*$), $v(y^i)$ is the optimal value of the master at iteration i with right-hand side y^i as in (18); and finally the denominator denotes the Euclidean norm of the dual vector $\gamma(y^i)$ corresponding to y^i, that is,

$$||\gamma(y^i)||^2 = \sum_{k=1}^{K} ||\gamma^k(y^i)||^2 = \sum_{k} \sum_{a \in A} (\gamma_a^k(y^{k,i}))^2 \tag{31}$$

Held and Karp [3-15] and Oettli [3-27] give theoretical justification for the form (31) and $\lambda_i = 1$.

The subgradient approach is made attractive by the ease with which the projection in (28) can be performed: For each arc a in the network, the intermediate capacities $(\tilde{y}_a^1, \ldots, \tilde{y}_a^K)$ obtained from (27) have to be projected upon the set

$$S_a = \{x = (x^1, \ldots, x^K) \in R^k \mid \sum_{k=1}^{K} x^k = d_a, x^k \geq 0 \text{ for all } k\} \tag{32}$$

where d_a is the total capacity of arc a. Specifically, we require the solution $y_a = (y_a^1, \ldots, y_a^K)$ to the minimum norm problem

Minimize:

$$||\tilde{y}_a - x||^2 \tag{33}$$

subject to:

$$x \in S_a.$$

The procedure involves ordering the components of $\tilde{y}_a$ to form a new vector Y whose components Y^k satisfy:

$$Y^1 \leq Y^2 \leq \ldots \leq Y^K.$$

The optimal solution y_a to (33) then satisfies

$$y_a^k = \text{Max}\{Y^k - \lambda^*, 0\}, \tag{34}$$

where

$$\lambda^* = \left(\sum_{k=J+1}^{K} Y^k - d_a\right)/K-J, \tag{35}$$

with J defined as:

$$J = \text{Max}\left\{ j \;\middle|\; \left(\sum_{k=j+1}^{K} Y^k - d_a\right)/K-j > Y^j \right\}. \tag{36}$$

The projection given above need not be performed if the capacities on arc a have not been changed in step Three of the algorithm. Indeed, the set of arcs for which the capacities have been altered in (27) is given by

$$T_i = \{a \in A \mid \gamma_a^k(y^i) > 0 \text{ for some } k\}. \tag{37}$$

The arcs in T_i are flagged for projection. Similarly, the summation over A in (31) can be limited to T_i.

Computationally one expects T_i to be small in comparison with A. In particular, any arc which is not individually saturated by any commodity (i.e., for which $f_a^k < y_a^{k,i}$ for all k) will belong to the set $A-T_i$ and so its capacity allocation will not be altered at iteration i. This computational fact suggests that the optimal solution and dual variables

of subproblem (16.1) - (16.5) at iteration i will serve as a good starting solution for the primal-dual algorithm at iteration i+1 since relatively few components of $y^{k,i+1}$ are different from $y^{k,i}$.

3.1.3 Special Purpose Algorithm

The Multifleet Routing Problem was originally described by Simpson [3-28]; Levin had some partial success in obtaining small-scale solutions using branch and bound techniques. But with the size of networks which describe the typical schedule map for a typical domestic airline being of the order of 1000 nodes and 3000 arcs, there is need for greatly improved computational efficiency if successful applications are to be made. The special purpose "bundle pricing" algorithm which we describe in the following is one such attempt.

Consider first the SFRP, with the objective of finding the optimum number of vehicles and their routing in the schedule map or, alternatively, which services should be operated with a given number of vehicles. We note that since every vehicle in the fleet must recycle, the cost on the cycle arcs can be regarded as a rental or toll corresponding to ownership costs for a vehicle. This cost, λ, can also be used as a Lagrange Multiplier (or a dual "price") to control the number of vehicles in the fleet. Therefore, by increasing the cost on the cycle arcs, vehicles become more expensive to own and less likely to find a profitable routing in the system. Thus system income and the number of aircraft in service decrease as the cycle costs λ increase, as shown in Figure 3.1.

In any solution of a problem of this type, an individual service arc has a value which depends very much upon the values of the other service

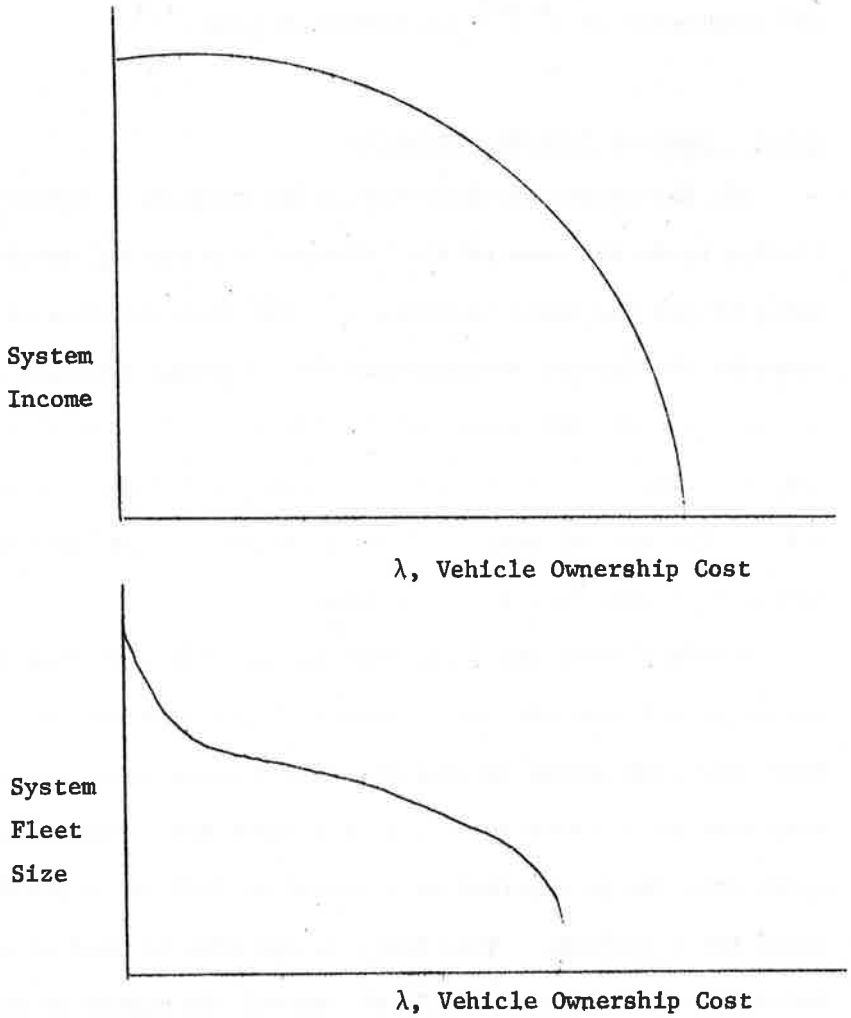


FIGURE 3.1 SYSTEM INCOME AND FLEET SIZE-FLEET ROUTING MODEL

arcs selected by the complete routing pattern. Consider an arbitrary schedule map. We will find that the most profitable routings for two (identical) aircraft may not include services selected in a single aircraft routing. Also, if we insist that a particular service arc be flown (or not flown) in a particular solution, the impact of this constraint on system income will vary in a non-monotone fashion. For example, the reduced cost $\bar{c}_a$ indicates the marginal cost or value of a service to the solution. As λ is varied, the $\bar{c}_a$'s can vary in value from positive to negative to positive, with service occurring or not occurring on the arc, due to the different fleet routing patterns which occur.

In the MFRP, many (but not necessarily all) of the services may be flown by different aircraft. The same routing characteristics of the SFRP, which cause the value of a service arc to vary erratically as the fleet routing solution is changed, may mean that a service arc should be flown, at a low load factor and at an apparent loss, by a large, expensive aircraft, since it may enable the routing of that aircraft to achieve a more profitable pattern of service throughout the rest of the cycle. This implies the need for the addition of the "bundle" constraints which make the problem much more difficult to solve.

If there were only one bundle, we could simply use the Lagrange multiplier technique as described for controlling fleet size for the single fleet problem, i.e., simply artificially decrease the value of the service until only one type of aircraft wished to continue service. But with multiple constraints, we are faced with finding a vector of prices, γ, which will cause 0-1 flows on every bundle of service arcs, so that we cannot conduct a simple search for the correct price vector. Several

versions of the "Bundle Pricing Algorithm" (BPA) were developed as a systematic way of conducting the search for γ.

The strategy involved in the development of the BPA revolves around the relative structural independence of each copy of the schedule map from every other within the framework of the MFRP. This independence is easily seen from the schematic tableau of the MFRP, Figure 3.2, whence it is obvious that the m fleet MFRP consists of m SFRP's coupled only by the Service Arc Bundle Constraints. Therefore the approach is to decompose the problem into m SFRP's which are easily solved using either existing efficient Out-of-Kilter (OKF) algorithm codes, or, as we have mentioned previously, newly improved primal simplex-based codes.

Because of the inherent profitability of some routes (or other reasons such as the requirement of cycle flows), almost inevitably the composite solution to the MFRP obtained from the simple combination of the m SFRP solutions will be infeasible, i.e., some services will be flown by more than one aircraft type, in violation of the MFRP bundle constraints. In order to eliminate these infeasibilities, the BPA seeks to artificially increase the costs of the services to the point where the composite solution of the m separate SFRP's yields 0-1 flows in each arc with a bundle constraint.

To do this we note that at any given iteration when we solve the SFRP (with OKF) we obtain the copy flow, flow cost, and the dual variables, particularly the arc reduced costs, $\bar{c}_a^k = (c_a^k - \alpha^k e_a^k)$, where α is the dual prices, and e the matrix elements in the tableau of Figure 3.2. These $\bar{c}_a^k$ values represent a bound on the change in copy flow cost if the flow in the bundle arc was to be changed, i.e. they give an indication of the

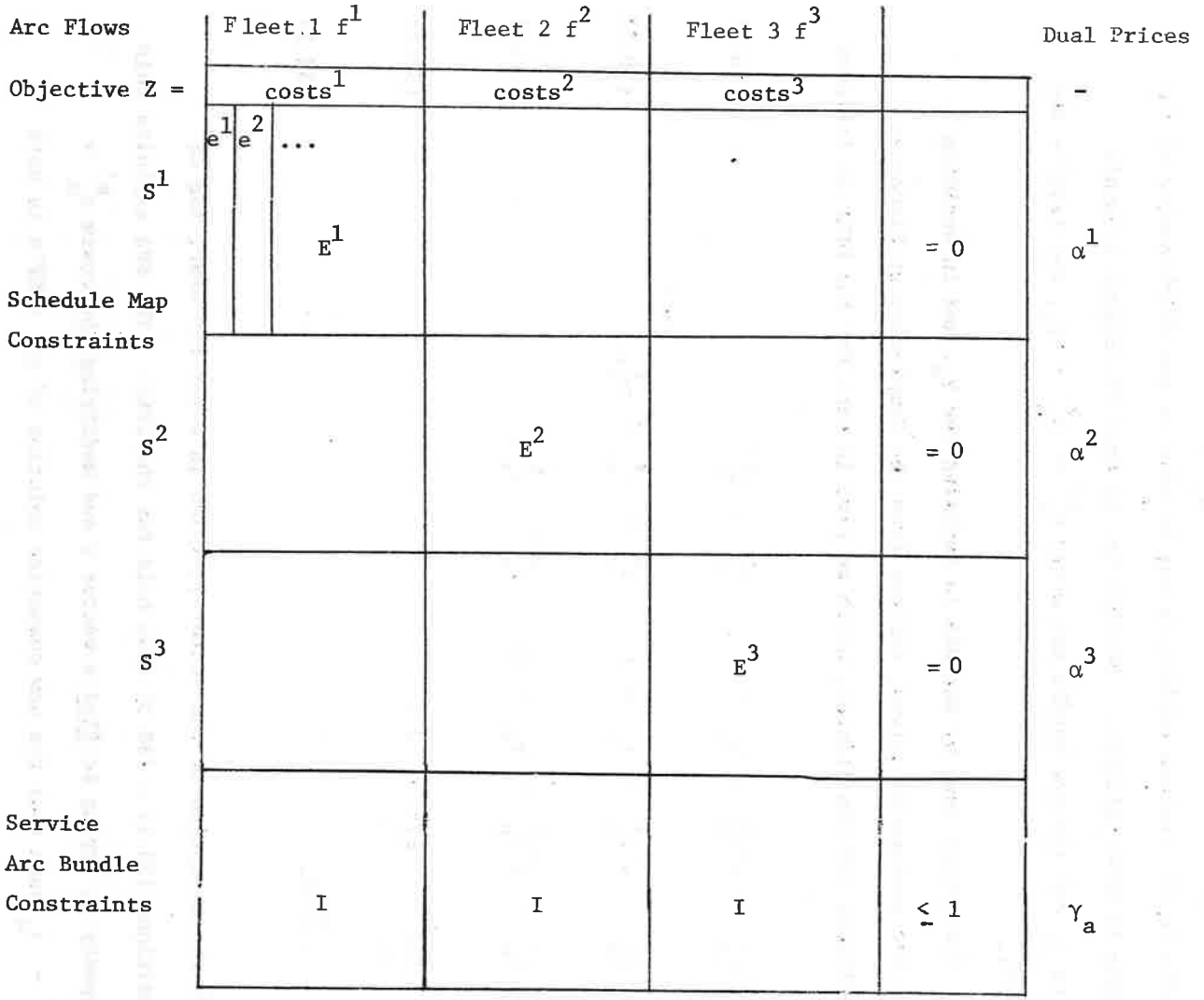


FIGURE 3.2 TABLEAU FOR MULTIFLEET ROUTING PROBLEM

value to the copy flow of the flow in the arc. For those arcs with bundle constraints, by looking at the $\bar{c}_a^k$ values for all members of the bundle in the various copies, we may be able to see which member of the bundle is most valuable. We then may be able to compute a bundle price γ_a and the new bundle arc costs $c_a^{k'} = c_a^k - \gamma_a$, and resolve the SFRP's.

The device used by the BPA in choosing the γ_a, and in deciding when the problem is solved, follows from the Complementary Slackness Conditions (CS conditions), which we give in part for the MFRP as follows:

$$c_a^k - \alpha^k e_a^k - \gamma_a < 0 \qquad f_a^k = \ell_a^k, \tag{38.1}$$

$$c_a^k - \alpha^k e_a^k - \gamma_a = 0 \qquad \ell_a^k \leq f_a^k \leq u_a^k, \tag{38.2}$$

$$c_a^k - \alpha^k e_a^k - \gamma_a > 0 \qquad f_a^k = u_a^k, \tag{38.3}$$

$$(\Sigma_k f^k - \underline{1})\gamma = 0, \tag{38.4}$$

$$\gamma \leq 0. \tag{38.5}$$

With the exception of the vector γ, which is 0 for the SFRP, the CS conditions (38.1) - (38.3) also hold for the SFRP. The BPA exploits this property in trying to find a vector γ and modifying the costs $c_a^{k'} = c_a^k - \gamma_a$ such that the new composite solution of the SFRP's is both feasible and satisfies the MFRP CS conditions (38.4) - (38.5).

In essence BPA will use the $\bar{c}$ values of the decomposed copy flows to search for the vector of bundle prices, γ which produce an optimal feasible set of copy flows for the MFRP. The clue to the correct choice of γ is the CS condition (38.1) - (38.3) applied to the SFRP, since we associate with $\bar{c}_a^k > 0$ a flow $f_a^k = 1$ and $\bar{c}_a^k < 0$ a flow $f_a^k = 0$. Consequently, if a given bundle constraint is violated (more than one $f_a > 0$) we have at least a myopic view of what the value of the $\bar{c}_a^k$'s should be, and hence the direction and magnitude of change of γ_a. Once a γ_a has been chosen for all bundles, the SFRP's are resolved with the new costs and the BPA starts all over again.

In section 3.2.3 we describe several options for selecting the bundle prices α for this algorithmic approach.

3.2 IMPLEMENTATION DETAILS

This section describes the implementation and program structures of price and resource directive algorithms.

3.2.1 Price-Directive Decomposition

The program for the price-directive decomposition algorithm involves a linear programming code to solve the master problem and shortest chain algorithm to solve the subproblems. The main program, called DCMP for decomposition, deals with the master problem and communicates with two subroutines GENCOL and BELL that are described below.

The SEXOP package (Subroutines for Experimental Optimization), developed by R.E. Marsten [3-22], is used to solve the master problem at each iteration. SEXOP consists of a collection of subroutines which

are well suited for column generation and are designed for interactive use. In particular, the number of columns in the linear program are allowed to vary from one iteration to the next. A subroutine ADDCOL can be used to add columns to the master problem. The optimal solution to the master at iteration i serves as a starting solution for iteration i+1 in SEXOP and the convexity constraints (8.3) are treated as GUB constraints. Thus SEXOP may be expected to solve the master problem efficiently at each iteration. The only disadvantage of SEXOP is that the number of regular constraints (8.2), which correspond to the number of capacitated arcs in the network in our formulation, are restricted to be less than or equal to 99 in the current working version. It would not be advisable to use SEXOP for problems with more than around a hundred regular constraints anyway, since the program would be inefficient compared to commercial codes, especially if the problem is sparse. We reiterate that a salient advantage of SEXOP is the ease of interacting with it.

The shortest chains are found by Bellman's algorithm. The version we use, called BELL, was coded by B. Golden and is described in [3-13]. The code uses a "Forward Star" network representation scheme [3-10] in which the arcs $(i,j) \in A$ are ordered lexicographically in increasing order. Thus, for a given node $i \in N$ we list all arcs $(i,j) \in A$ in the order of increasing j and then go on to node i+1 to do the same, continuing until i = NN, where NN denotes the number of nodes in the network. Thus two arrays are sufficient to represent the network topology: a pointer of length NN giving the arc number (in the above ordering) of the first arc starting with node i; and an end-node array of length NA listing the j's for all $(i,j) \in A$. (NA is the number of arcs in the network.) This representation scheme was extended to the rest of our code.

The subroutine BELL finds the shortest distances from a specified origin s to all other nodes in the network. The labeling assigns each node i the cost of the shortest path (s) from s to i and a predecessor node j. The output of BELL is thus a predecessor vector and a cost vector. Given a destination t one may immediately obtain the length of the shortest path from s to t from the t-entry of the cost vector and trace through the predecessor vector (starting at t) to obtain the optimal path in terms of a sequence of nodes $[s,i_1,i_2,\ldots,i_{n-1},t]$.

This procedure can easily be extended to a commodity with more than one destination. Given the source and n^k destinations, we call BELL once and trace through the predecessor vector n^k times to find the shortest paths from s to each of the n^k destinations.

The above is actually done in subroutine GENCOL which generates the column f^k to be added to the master problem in a form specified by ADDCOL. This column has entries equal to F^k in the places corresponding to capacitated arcs on the path from s^k to t^k and zeros elsewhere. More specifically, if an arc $(i_e i_{e+1})$ on the path from s^k to t^k is capacitated and corresponds to the q^{th} capacity constraint, we put F^k in the q^{th} entry of f^k. While tracing through the predecessor vector, GENCOL also finds the "real" chain costs, $c^k \cdot f^k$, which are required by SEXOP as the corresponding column costs.

The interaction between the subprograms is shown in Figure 3.4. The dual variables π_t from the master problem are passed on to GENCOL by DCMP. GENCOL uses π_t to obtain the modified arc costs $(c^k-\pi_t D)$ which are then passed on to BELL with the specified origin s^k. The length of the chain from s^k to t^k with respect to modified arc costs, which is w^k in

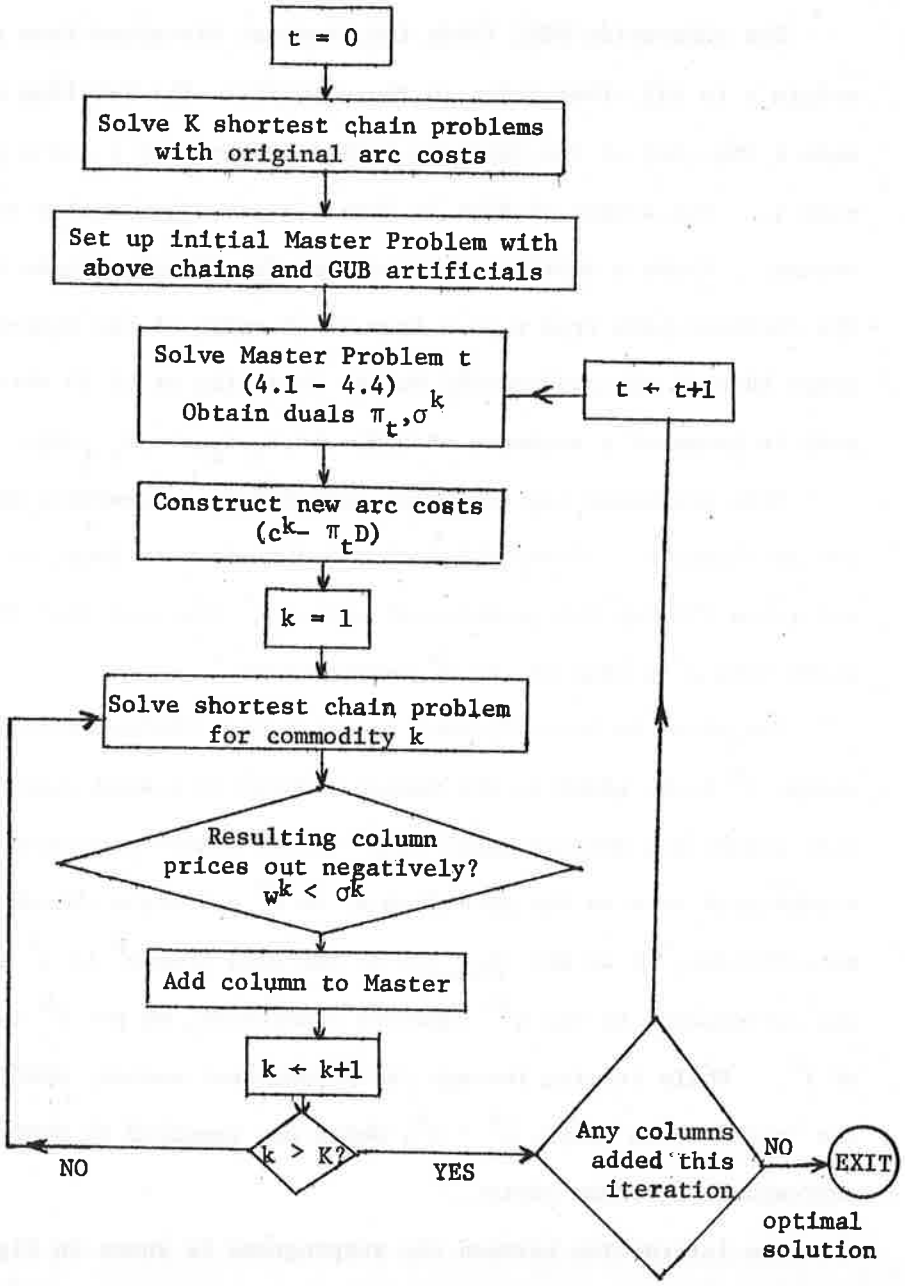


FIGURE 3.3 FLOW-CHART FOR DANTZIG-WOLFE DECOMPOSITION

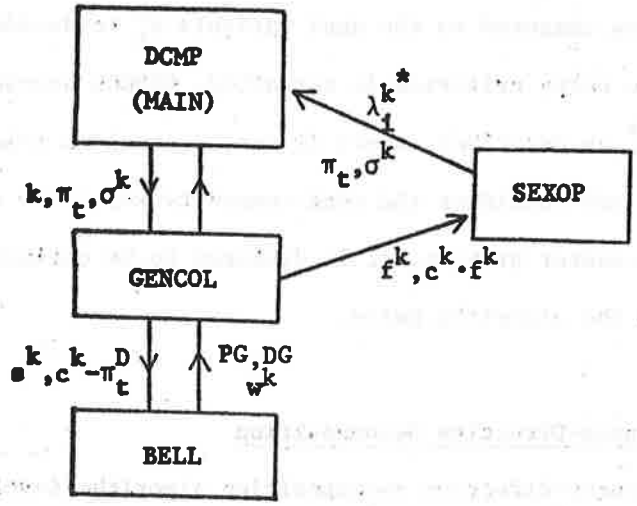


FIGURE 3.4 INTERACTION BETWEEN DCMP AND ITS SUBPROGRAMS

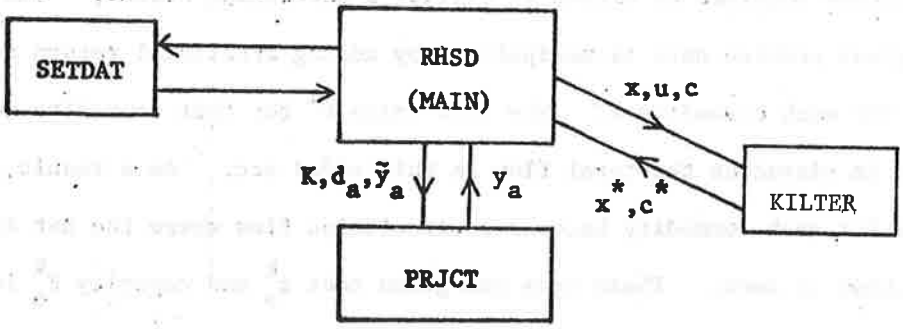


FIGURE 3.5 INTERACTION BETWEEN RHSD AND ITS SUBPROGRAMS

(6.1), is then compared to the dual variable σ_t^k to decide upon entry. If the column entry criterion is satisfied, GENCOL proceeds to generate the column f^k as described above; if not, control is transferred back to DCMP, which then considers the next commodity k+1. If no columns are added to the master problem, it is declared to be optimal by virtue of (7.1) and the algorithm halts.

3.2.2 Resource-Directive Decomposition

The resource-directive decomposition algorithm involves a master problem, the main component of which is a projection algorithm called PRJCT; and subproblems which are solved by an Out-of-Kilter code called KILTER. The main program is called RHSD (for right-hand side decomposition). We now describe each component of the code in detail.

The main program reads in the problem data and sets up the data structure required by KILTER by calling a subroutine SETDAT, The original problem data is manipulated by adding artificial return arcs (t^k, s^k) for each commodity k. The flow value F^k for that commodity may thus be viewed as the total flow on this added arc. As a result, the flow for each commodity becomes a circulation flow where the net flow at any node is zero. These arcs are given cost c_o^k and capacity F_o^k in accordance with the subproblem definition in (16.1) - (16.5). SETDAT also transforms multiple arcs between a pair of nodes -- i.e. (i,j) and (j,i) -- by inserting an additional artificial node in one of the arcs, thereby adding a new arc to the network. Thus the total number of arcs in the new network structure is

$NA^* = NA + NCOM + NMULT,$

where NMULT is the number of multiple arcs (corresponding to undirected arcs in our case) in the original network. This transformation is required by KILTER. Several pointer arrays needed by KILTER are also constructed by SETDAT.

KILTER is an efficient Out-of-Kilter algorithm which uses recent ideas in data manipulation and is found to be computationally superior to a number of commercial codes. The algorithm, as well as the required data structure, is discussed in detail in [3-1, 3-2]. As used by our code, KILTER requires the specification of three vectors x, u, and c, of length NA^*, which contain the flow, the capacity (upper bound on flow), and the reduced cost for every arc in the network. Naturally, as for all primal-dual methods, flow values need not be feasible (i.e., may have $f_a > u_a$ on some arc a). The algorithm then looks for an optimal solution by maintaining dual feasibility and complementary slackness and striving for primal feasibility. The subroutine KILTER returns the optimal flows and reduced costs x^*, c^*.

PRJCT is the name given to the subroutine performing the projection (33) by using the algorithm outlined in equations (35) - (36). As noted there, the projection requires an initial ordering of the components of the input vector $\tilde{y}_a$ for arc a. This is done by using the heap-sort algorithm given in Fortran in [3-26]. To call PRJCT one need only specify the parameters K, d_a and the capacity allocation vector $\tilde{y}_a$ for any arc a. PRJCT returns the solution y_a.

Figure 3.5 shows the interaction between program components. Upon constructing the data structures and the augmented network structure to

be used by all subproblems, the main program calls KILTER to solve each subproblem with the initial values of zero flow, the capacity vector y^0, and the original arc costs. Thereupon at any given iteration i, the subproblems are solved with right-hand side $(u =) y^{k,i}$ for k=1, ...K. The dual variables $\gamma^k(y^{k,i})$ are obtained from the optimal reduced costs of the subproblem by the relation:

$$\gamma_a^k = \text{Max}\{0, -\bar{c}_a^k\}, \tag{39}$$

where $\bar{c}_a^k$ is the optimal reduced cost on arc a. The dual variables in (39) determine the set of arcs T_i to be flagged for projection [see (37)] for which the subroutine PRJCT is called.

Note that at any iteration, the only change in each subproblem k is its right-hand side, and this change occurs in relatively few entries of y_i^k. Thus the solution x^* and reduced costs c^* to the subproblem at iteration i are saved to serve as the starting solution at iteration i+1. This strategy may demand substantial storage for large K, 2K vectors of length NA, but is believed to speed up the solution of subproblems substantially. Clearly, when solving subproblem k, only one artificial commodity return arc (t^k, s^k) is given a nonzero upper bound F_o^k. All the other return arcs have a capacity of 0 for commodity k. This simple rule allows us to use a single data structure -- that of the full network -- horizontally across the subproblems.

One problem in using KILTER to solve the subproblems was a mismatch of data types. KILTER accepts only integer values as data and is written in terms of integer arithmetic. The capacity allocation resulting

from the projection routine are obviously real variables, and as such should be rounded off before they could be passed to KILTER as the vector u. To achieve this, we simply multiplied the capacities by an appropriate power of ten (depending upon the degree of accuracy required) and placed the result in the integer array u. We note that as a result, the optimal dual variables γ_a^k defined in (39) above will also be integer and differ slightly from the real dual variables for the problem. Originally we were concerned whether this would affect the convergence properties of the algorithm. To check this, we coded an earlier version of RHSD using SEXOP, in place of KILTER, to solve the subproblems.

A small test problem with 10 arcs, 6 nodes, and 2 commodities was chosen to carry out the comparison. All arcs were capacitated although on 5 arcs capacities were large enough not to be binding. Below we list the optimal value of the master and the step size obtained at the end of each iteration from the two versions S and K of the algorithm using SEXOP and KILTER respectively. In rounding off, only 4 significant digits were used.

Iteration	v_S	t_S	v_K	t_K
1	753.20	0.0140	752.82	0.0141
2	873.40	0.0072	875.70	0.0070
5	917.55	0.0062	917.49	0.0061
10	941.06	5.8941	941.07	5.8931
11	942.00	3.6250	942.00	5.8000

Convergence is obtained at iteration 11 for both algorithms. The intermediate results are also in close agreement as seen above. We also

compared the two versions on a larger test problem of 26 arcs, 9 nodes, and 4 commodities. Again the discrepancies were insignificant.

Given that rounding off is no problem, we believe that the SEXOP option is inferior since it currently offers no possibility of saving the previous subproblem solution in core for more efficient re-optimization. The best one can do is to use the dual simplex method on the solution for subproblem k as a starting solution for subproblem k+1. When using KILTER, however, we save the solution of each subproblem to start off the Out-of-Kilter algorithm for the same subproblem in the next iteration.

3.2.3 Bundle Pricing Algorithm

In this section we give a more formal statement of one version of the Bundle Pricing Algorithm which we have called "ALLNOT.0". Although the algorithm is explicitly written for a 3-fleet problem, the logic is applicable to an arbitrary number of copy flows. Several other variants operated in much the same framework. The underlying principle of all of them is to modify the costs c_a^k of the copy flows by looking at the reduced costs $\bar{c}_a^k$ and applying one of the variants of the myopic decision rule. That is, we compute a new vector of bundle prices γ' by a rule such as

$$\gamma' = \min\{0, \gamma + \bar{c}^*\},$$

where γ is the old vector of bundle prices and the $\bar{c}^*$ values measure the extent to which the reduced costs must be changed to achieve a flow satisfying the MFRP CS conditions. Note that this rule restricts $\gamma \leq 0$, which is a dual feasibility condition of the MFRP.

As experience with the algorithms has been achieved, they were continually modified in trying to improve their performance. For example, some decision rules seemed to cause more drastic changes in the copy flows at the following iteration than others. One variation in this type of problem has been to stop putting $\gamma=0$ in step five of ALLNOT.0. Instead, a lower bound of unity was placed on each bundle arc to "flood" the bundle and create some nonzero $\bar{c}$ values in a manner similar to the process of step four.

One algorithm, SPLIT.1, seemed to have more success in arriving at a final solution with, however, penalties in the rate of convergence. Here, in an attempt to choose the correct bundle price γ', we define $\bar{c}^*$ not as the maximum or minimum value of $\bar{c}$ in the bundle, but rather as their average

$$\bar{c}^* = \frac{\bar{c}_{max} + \bar{c}_{min}}{2} .$$

This was motivated by the observation that a unique γ' does not exist. Rather a range exists for each bundle price in the optimal vector and we should try to stay in the middle of the range.

All of the algorithms, including the OKF algorithm, were coded in Fortran and maintained on the IBM timesharing system at the MIT Information Processing Center. By having the problem solved on TSO the analyst can monitor the performance of the algorithm from iteration to iteration, and modify the decision rules as necessary.

Let us now give a formal statement of the algorithmic steps for the ALLNOT.0 version of the bundle pricing algorithm. Consider a three-fleet case where the flows in each copy of the schedule map are given by $\underline{x}$, $\underline{y}$, and z. We use superscript b to indicate arcs belonging to a bundle, e.g.

γ^b is the common bundle price arising from the bundle capacity constraint.

STEP 1) Given a new bundle price γ^b, use the OKF Network Flow Algorithm to solve each copy for sub-optimal flow, $\underline{x}$, $\underline{y}$, $\underline{z}$, and $\underline{\bar{c}}$ (the corresponding reduced costs). If $u^b = 0$, put $u^b = 1$ after solving.

STEP 2) Scan all bundles, computing $f^b = x^b + y^b + z^b$ = total bundle flow.

a) If $f^b > u^b$, and the smallest $\bar{c}^b$ is negative, denoted $\bar{c}^*$, go to (3).

b) If $f^b < u^b$, and $\gamma^b < 0$, and the largest $\bar{c}^b$ is positive, denoted $\bar{c}^*$, go to (3).

c) If $f^b > u^b$, and the smallest $\bar{c}^b = 0$, go to (4).

d) If $f^b < u^b$, and $\gamma^b < 0$, and the largest $\bar{c}^b = 0$, go to (5).

e) Otherwise, go to (6).

STEP 3) Change bundle price, $\gamma^{b'} = \text{MIN}\{0, \gamma^b + \bar{c}^*\}$. Go to (6).

STEP 4) Change upper bounds to zero, $u^b = 0$. Go to (6).

STEP 5) Change bundle price, $\gamma^{b'} = 0$. Go to (6).

STEP 6) If all bundles have been scanned, without any changes from (3), (4), or (5), go to (7). Otherwise go to (1).

STEP 7) STOP, optimal solution satisfying complementary slackness.

i.e. for all bundles, $f^b = u^b$, $\gamma^b \leq 0$ or

$f^b < u^b$, $\gamma^b = 0$.

3.3 COMPUTATIONAL RESULTS FOR MULTIFLEET ROUTING PROBLEMS

In this section we present numerical results for the MFRP using three different computational techniques:

1) Dantzig-Wolfe Decomposition
2) Bundle Pricing Algorithm
3) Linear Programming.

One problem was attempted with all three: this is the Tech Airways Problem originally described in Cohen [3-7] and given in Figure 3.6. There are 58 flights which could be flown by either of two different types of aircraft. Each resultant network copy consists of 58 service arcs, 85 ground arcs, 5 cycle arcs and 90 nodes. Revenue from the flight arcs vary by aircraft type, ground arcs have zero cost, and cycle arcs, reflecting aircraft ownership expense, have costs of $1500 and $2000 for aircraft types 1 and 2 respectively. Many other smaller problems were tried with the Bundle Pricing Algorithm. These will be discussed in 3.3.2.

Note that we did not test the subgradient optimization on this problem because of its inferior performance when compared to Dantzig-Wolfe decomposition for other types of multicommodity flow problems. The details of these numerical comparisons are given in section 3.4.

3.3.1 Dantzig-Wolfe Decomposition

As we discussed in subsection 3.1.1, the multicommodity flow decomposition code was modified slightly because of the special features of the multifleet routing problem. We shall refer to this version of the code as DCMP2. Note that, in contrast to general purpose version DCMP

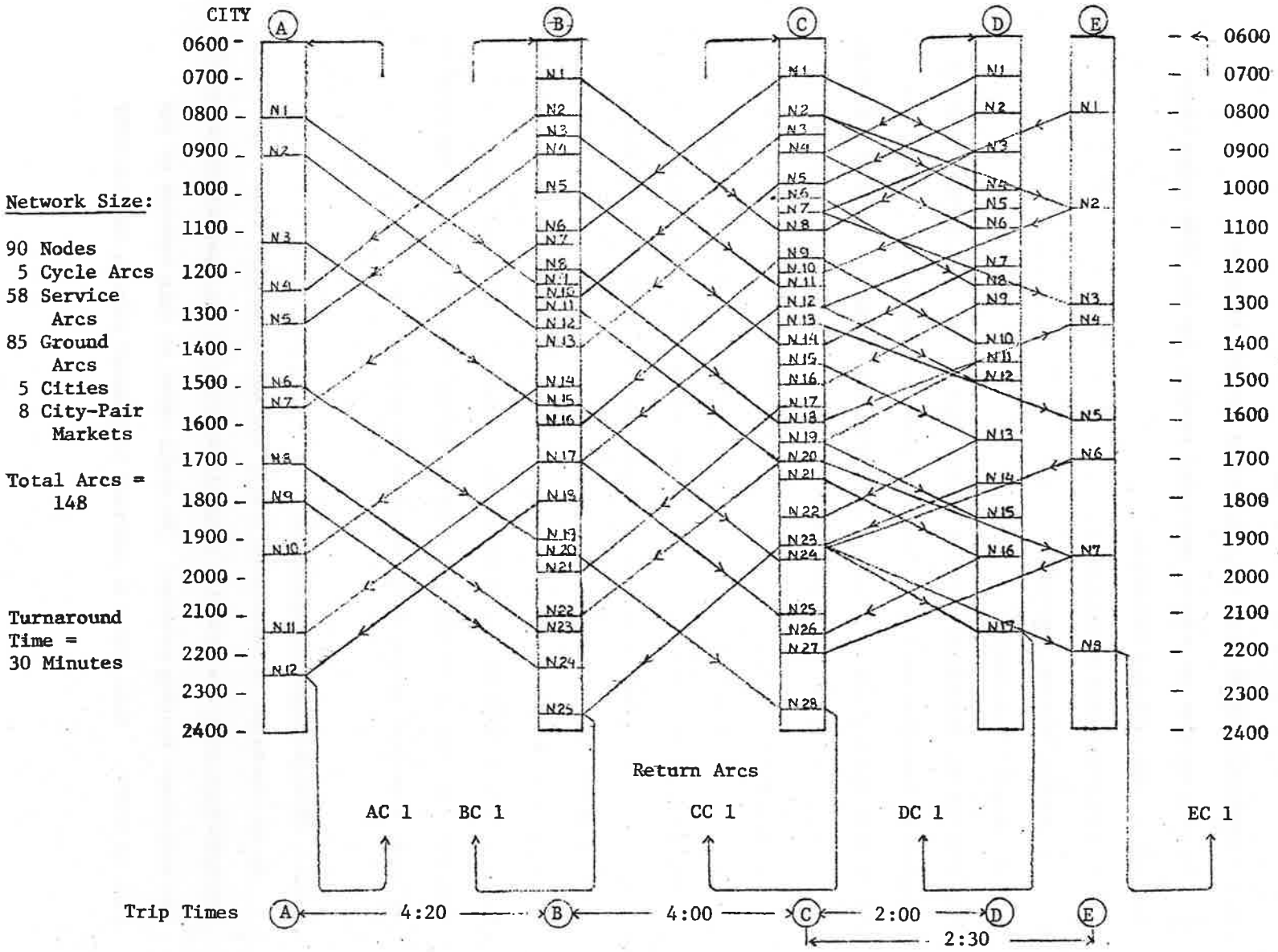


FIGURE 3.6 TECH AIRWAYS SCHEDULE MAP

of the code, the flow values F^k are decision variables in DCMP2.

First we modify the Tech Airways problem to the format indicated at the end of subsection 3.1.1. Upon adding supernodes and replacing the cycle arcs for each city in Tech Airways by arcs emanating from the supersource and terminating at the supersink, we will have two more nodes and ten arcs. Obviously the number of ground and service arcs remain unchanged. Thus in our notation the problem parameters are:

NN = 92, NA = 58 + 85 + 10 = 153, NCOM = 2.

Only two aircraft types are used (K = NCOM = 2) and the only capacitated arcs are the service arcs NCAP = $|A_s|$ = 58.

We shall now describe the solution obtained by DCMP2: We start by noting that each colum ($f^{k,i}$, 1) describes a chain from s to t with unit flow which fully describes the schedule of an aircraft of type k during the planning cycle. Thus each vessel starts from a certain city at the beginning of our time cycle and, upon traversing a combination of service and ground arcs (corresponding to flight and waiting periods), terminates its activity in some city at the end of the time cycle. To describe the optimal solution we need only list those basic columns at optimality with strictly positive flow. Note further that for integrality we need each ρ_i^k to be integer; otherwise, fractional flows will enter the picture. The following schedules combine to form the optimal solution:

(S1) C1 - D3 - C8 - C13 - E5 - E6 - C23 - D17

(S2) D1 - D2 - C5 - B13 - B17 - A11 - A12

(S3) B1 - B2 - A4 - A8 - B23 - B25

(S4) D1 - C4 - D6 - D14 - C23 - E8

(S5) B1 - B5 - C14 - C21 - D16 - C26 - C28

(S6) C1 - C2 - E2 - C12 - B17 - C25 - C28

(S7) E1 - C7 - C9 - B16 - B18 - A12

(S8) C1 - B6 - B7 - A7 - A9 - B24 - B25

(S9) A1 - B9 - B11 - C20 - E7 - C27 - C28.

To explain the notation we examine (S4) more carefully: The aircraft starts in city D at node D1, then flies to C4 followed by another flight back to D6. Then there is a waiting period in city D characterized by D6 - D14 (all intermediate nodes are deleted for simplicity), followed by two flights first to C23, then to E8. Thus the schedule of the aircraft is completely specified during the planning horizon of one cycle.

The aircraft type is also specified for each schedule (the index k gives the type) and in our solution schedule (S2) corresponds to a type 2 aircraft). All otheraircrafts (8 in number) are of type one. The total fleet size is $F^1 + F^2 = 8 + 1 = 9$. Altogether 31 services are flown with a resultant optimal value for the objective function of z = \$17,000.

Table 3.1 shows the convergence of DCMP2 for this problem. The optimality criterion is satisfied after 22 iterations of the master problem. The CPU time excluding I/O on the IBM 370/168 machine was 0,67 seconds, which is most encouraging for this type of problem. Another encouraging (though not unexpected) result was the integrality of the optimal solution.

TABLE 3.1

CONVERGENCE BEHAVIOR FOR THE MULTIFLEET PROBLEM

Iteration	Objective Function
1	6,400
2	6,400
3	6,400
4	8,250
5	8,250
6	10,050
7	11,550
8	11,550
9	11,550
10	11,650
11	11,650
12	12,950
13	12,950
14	13,550
15	14,650
16	15,650
17	15,850
18	16,000
19	16,000
20	17,000
21	17,000
22	17,000 (Terminates)

3.3.2 Special Purpose Algorithm

All of the variants of the Bundle Pricing Algorithm have successfully solved some small two-copy MFRP's after a varying number of iterations. The schedule map and costs for a typical network copy (which we call OKFDATA) are shown in Figure 3.7. It describes the network for scheduled service between 3 cities and has a total of 14 service arcs. For the second copy of a two-fleet routing problem we usually chose a network with costs and arc capacities which differed only slightly from the first copy (OKFDATA). This selection reflects the fact that it is most difficult for the Bundle Pricing Algorithm to generate a dual price vector γ when many of the same flight arcs want to enter the solution of each copy at each iteration.

To test the sensitivity of the algorithm to these kinds of solutions, we generated a variety of revisions of OKFDATA, which we called OKFDATA2, OKFDATA3, etc. Since the first copy is always constant, we will henceforth refer to a particular MFRP by the name of the second copy data only (i.e., a problem having first copy OKFDATA and second copy OKFDATA2 would be called simply OKFDATA2).

Essentially we found a wide variability in the solubility and number of iterations required, even within the framework of our small problems. For example, the algorithm with lower bounds, ALLNOT.2, solved the problem OKFDATA2, Figure 3.7, in 17 iterations, finding that the optimal bundle price vector was (solution given in Figure 3.8):

$$\gamma_2 = 50$$

$$\gamma_{12} = 17{,}750$$

$$\gamma_{14} = 7{,}200$$

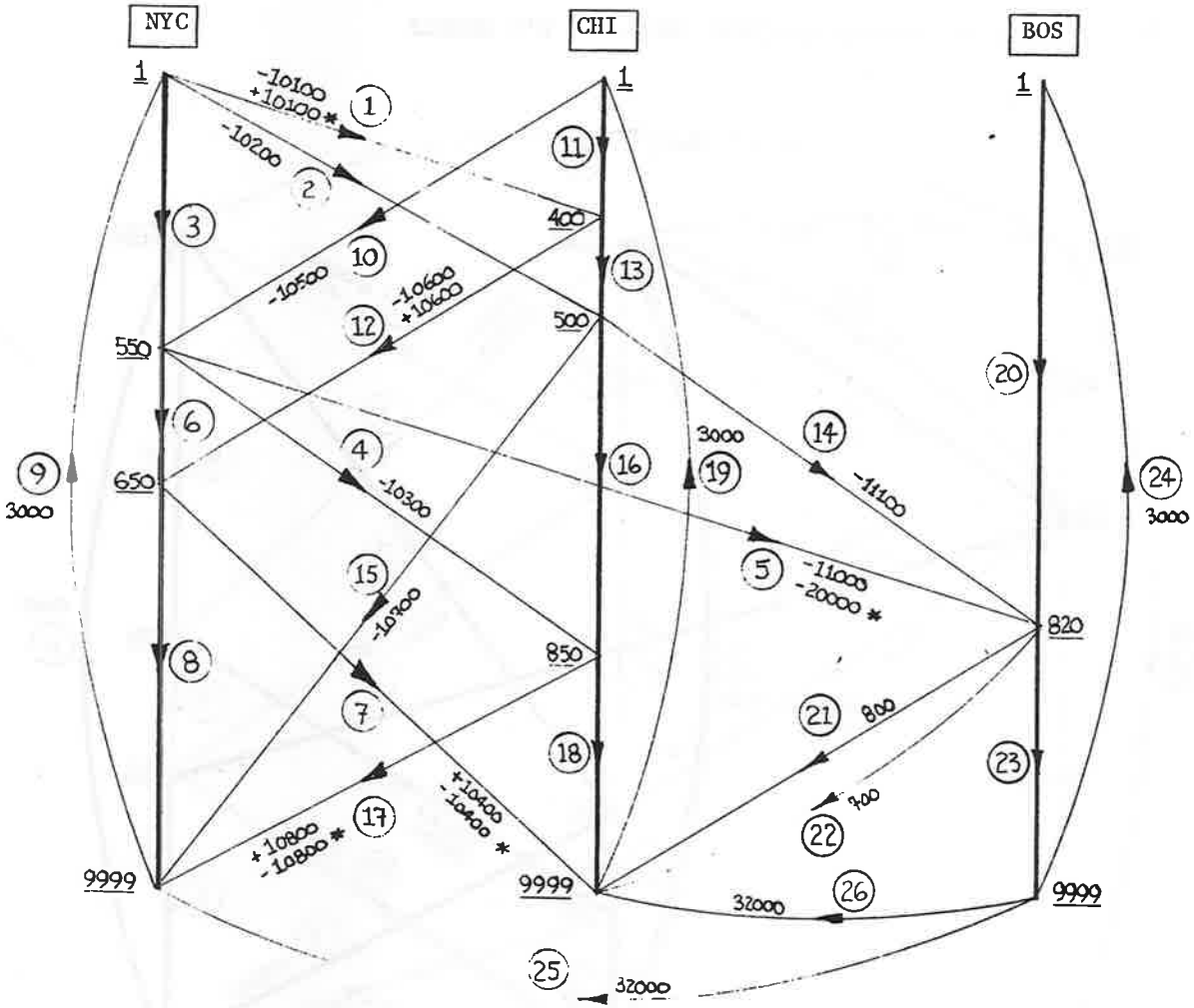
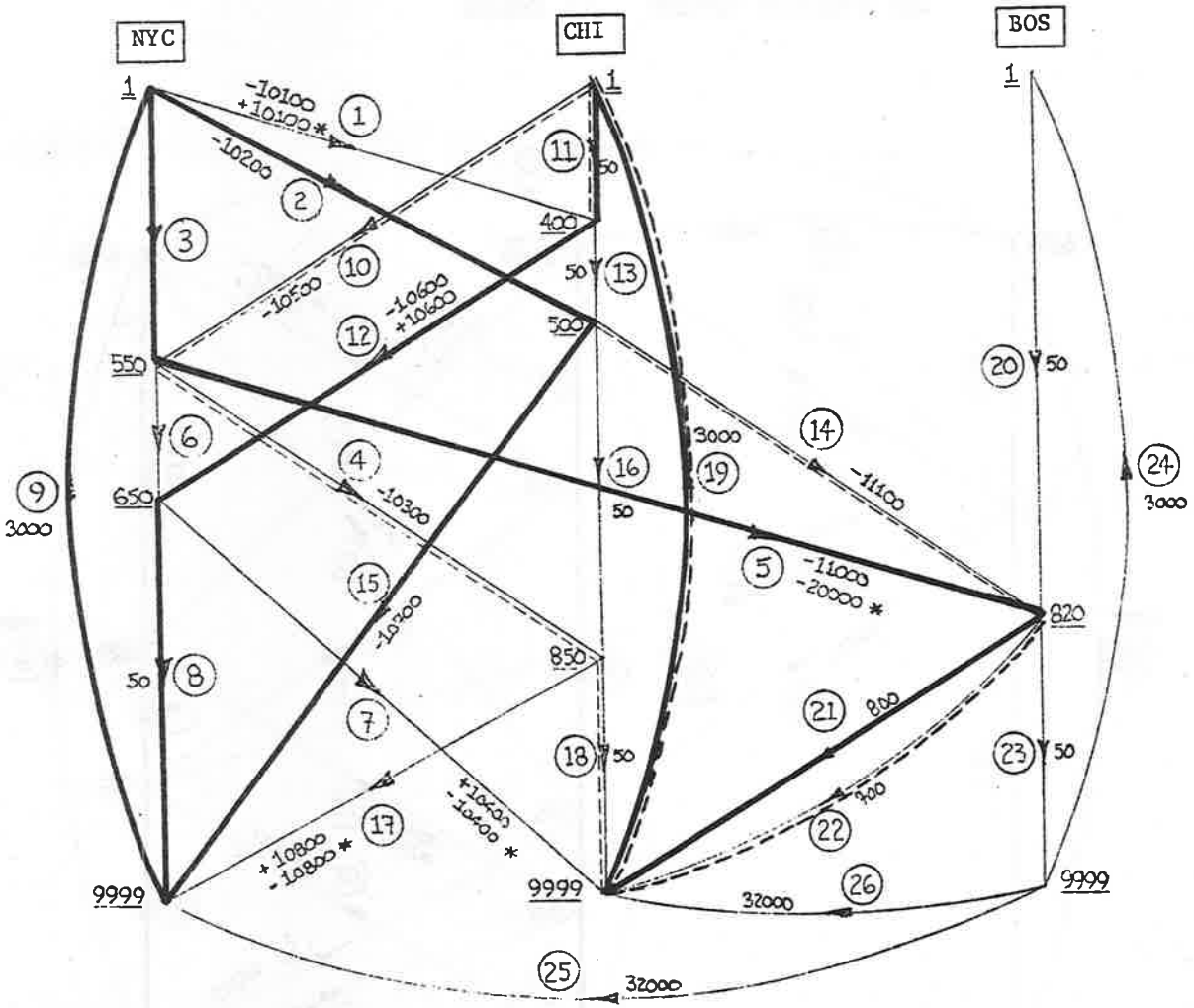


FIGURE 3.7 OKFDATA SCHEDULE MAP

Second copy costs which are different from first copy results are marked with an asterisk.



Notation

i) --- Flown by copy flow I

ii) ▬▬ Flown by copy flow II

iii) —— Flown by neither

FIGURE 3.8 OPTIMAL SOLUTION-PROBLEM OKFDATA2

$\gamma_{15} = 17{,}850$

$\gamma_{22} = 100$

All other $\gamma = 0$.

In contrast, a problem of similar size, OKFDATA3 (see Table 3.2 for comparison) was solved by both the SPLIT.1 and ALLNOT.1 algorithms in 3 iterations. However, after the arc CH 1-NY 550 was removed from the second copy, the problem would not solve at all.

The problem OKFDATA4 (Table 3.3) was solved by ALLNOT.1 in 3 iterations and ALLNOT.2 in 4. However, when OKFDATA4 is modified slightly to yield OKFDATA6 (Table 3.4), the problem OKFDATA6 solved with ALLNOT.1 in 10 iterations but remained unsolved with ALLNOT.2 after 20 iterations. It should be noted that in one problem of this size the algorithm SPLIT.1 converged after 86 iterations (of which 83 had only 1 bundle constraint violated!)

We have also been using a Tech Airways schedule map for a larger problem which consists of 5 cities, and 58 bundles. It is shown in Figure 3.6. So far this problem with two copy flows and a given set of arc costs has resisted solution by the bundle pricing algorithms. They very quickly bring all but several of the bundles into a satisfactory state, but then a random walk commences amongst the bundle prices and a total optimal solution is not achieved. Further analysis of these conditions may reveal the problem and lead to an algorithm with better final convergence. Because the algorithm does seem to obtain solutions satisfying all but a few bundle constraints quickly, though, it may be attractive to use, even in its present form, for large networks to generate starting solutions for other algorithms such as Dantzig-Wolfe decomposition.

TABLE 3.2 A COMPARISON OF OKFDATA WITH OKFDATA3

ARC NAME				COSTS		CAPACITIES	
				OKFDATA	OKFDATA3	OKFDATA	OKFDATA3
NY	1	CH	400	-10100	-10100	1	0*
NY	1	CH	500	-10200	-8000*	1	1
NY	1	NY	550	50	50	100	100
NY	550	CH	850	-10300	10300*	1	1
NY	550	BO	820	-11000	-15000*	1	1
NY	550	NY	650	50	50	100	100
NY	650	CH	9999	10400	5000*	1	1
NY	650	NY	9999	50	50	100	100
NY	9999	NY	1	3000	3000	100	100
CH	1	NY	550	-10500	-7500*	1	0*
CH	1	CH	400	50	50	100	100
CH	400	NY	650	-10600	2300*	1	1
CH	400	CH	500	50	50	100	100
CH	500	BO	820	-11100	-21100*	1	1
CH	500	NY	9999	-10700	6700*	1	1
CH	500	CH	850	50	50	100	100
CH	850	NY	9999	10800	10800	1	1
CH	850	CH	9999	50	50	100	100
CH	9999	CH	1	3000	3000	100	100
BO	1	BO	820	50	50	100	100
BO	820	CH	9999	800	800	1	1
BO	820	CH	9999	700	700	1	1
BO	820	BO	9999	50	50	100	100
BO	9999	BO	1	3000	3000	100	100
BO	9999	NY	9999	32000	32000	1	1
BO	9999	CH	9999	32000	32000	1	1

* Arcs in which costs or capacities are different from OKFDATA

TABLE 3.3 A COMPARISON OF OKFDATA WITH OKFDATA4

ARC NAME				COSTS		CAPACITIES	
				OKFDATA	OKFDATA4	OKFDATA	OKFDATA4
NY	1	CH	400	-10100	-10100	1	0*
NY	1	CH	500	-10200	-102000	1	1
NY	1	NY	550	50	50	100	100
NY	550	CH	850	-10300	10300	1	1
NY	550	BO	820	-11000	-15000*	1	1
NY	550	NY	650	50	50	100	100
NY	650	CH	9999	10400	5000*	1	1
NY	650	NY	9999	50	50	100	100
NY	9999	NY	1	3000	3000	100	100
CH	1	NY	550	-10500	-7500*	1	0*
CH	1	CH	400	50	50	100	100
CH	400	NY	650	-10600	-10600	1	1
CH	400	CH	500	50	50	100	100
CH	500	BO	820	-11100	-21100*	1	1
CH	500	NY	9999	-10700	6700*	1	1
CH	500	CH	850	50	50	100	100
CH	850	NY	9999	10800	10800	1	1
CH	850	CH	9999	50	50	100	100
CH	9999	CH	1	3000	3000	100	100
BO	1	BO	820	50	50	100	100
BO	820	CH	9999	800	800	1	1
BO	820	CH	9999	700	700	1	1
BO	820	BO	9999	50	50	100	100
BO	9999	BO	1	3000	3000	100	100
BO	9999	NY	9999	32000	32000	1	1
BO	9999	CH	9999	32000	32000	1	1

* Arcs in which costs or capacities are different from OKFDATA

TABLE 3.4 A COMPARISON OF OKFDATA4 WITH OKFDATA6

ARC NAME				COSTS		CAPACITIES	
				OKFDATA4	OKFDATA6	OKFDATA4	OKFDATA6
NY	1	CH	400	-10100	-10100	0*	1
NY	1	CH	500	-102000*	-8000*	1	1
NY	1	NY	550	50	50	100	100
NY	550	CH	850	-10300	10300*	1	1
NY	550	BO	820	-15000*	-15000*	1	1
NY	550	NY	650	50	50	100	100
NY	650	CH	9999	5000*	5000*	1	1
NY	650	NY	9999	50	50	100	100
NY	9999	NY	1	3000	3000	100	100
CH	1	NY	550	-7500*	-7500*	0*	1
CH	1	CH	400	50	50	100	100
CH	400	NY	650	-10600	-10600	1	1
CH	400	CH	500	50	50	100	100
CH	500	BO	820	-21100*	-21100*	1	1
CH	500	NY	9999	6700*	6700*	1	1
CH	500	CH	850	50	50	100	100
CH	850	NY	9999	10800	10800	1	1
CH	850	CH	9999	50	50	100	100
CH	9999	CH	1	3000	3000	100	100
BO	1	BO	820	50	50	100	100
BO	820	CH	9999	800	800	1	1
BO	820	CH	9999	700	700	1	1
BO	820	BO	9999	50	50	100	100
BO	9999	BO	1	3000	3000	100	100
BO	9999	NY	9999	32000	32000	1	1
BO	9999	CH	9999	32000	32000	1	1

* Arcs in which costs or capacities are different from OKFDATA

— Arcs that have different costs or capacities between OKFDATA4 and OKFDATA6

3.4 ADDITIONAL COMPUTATIONAL EXPERIENCE

To compare the price-directive and resource-directive algorithms for general multicommodity flow problems, we constructed four networks to test the two codes which we refer to as problems P1 through P4. The notation for problem parameters is as follows:

NA = number of arcs in the network.

NN = number of nodes in the network.

NCAP = number of capacitated arcs.

NCOM = number of commodities.

The basic factors which are varied in the test problems are network topology and the ratio of capacitated arcs to total number of arcs, i.e. NCAP/NA.

As noted earlier, the current implementation of SEXOP limit NCAP to be less than 100, which explains why the maximum number of capacitated arcs in our test problems is 98. In Problems P1 and P2 all arcs are capacitated, whereas in P3 and P4 the ratio NCAP/NA is roughly 0.48 and 0.38, respectively.

The network topologies were chosen with a view towards the modeling applications of the multicommodity flow problem other than the multifleet routing problem. The network structure in problems P1 and P3 is essentially acyclic (with only a very few cycles) and may be expected to arise in applications to multicommodity multiperiod, production-distribution problems or more generally in the construction of time space diagrams for scheduline vehicles on a network [3-8]. In our examples we conceive of certain arcs as denoting passage of time in a fixed spatial location, whereas the others denote spatial transportation from one point to another.

Thus in a production-distribution example the "time-like" arcs represent keeping a unit in inventory for one time period, thereby incurring an "arc" holding cost. The "space-like" arcs, however, stand for shipment of the items in the distribution network and the arc costs for these shipments could reflect transportation costs.

The other network topology -- corresponding to P2 and P4 -- corresponds to an undirected communications network. Thus for nodes i and j in the network $(i,j)\in A$ implies $(j,i)\in A$. Such networks would arise naturally in intercity transportation problems as well as in the currently expanding studies of computer networks. We note that the degree of nodes in this structure is greater than in the above and the networks contain more chains. The two network structures are different enough to warrant experimentation. We note that our choice of these structures was to some extent influenced by the work of Swoveland [3-29], whose test problems have the above two structures. All data were generated manually, and the arc costs basically satisfy the triangle inequality (with a few exceptions in any network). For listings of the problem data see [3-5].

3.4.1 Dantzig-Wolfe Decomposition

For each of the test problems P1 through P4, we conducted a series of runs to investigate the effects of

a) the number of commodities NCOM, and

b) the total flow FTOT of all commodities

on the number of iterations of the decomposition algorithm (i.e. the number of master problems solved to reach optimality) and the total time required

for the problem solution (excluding input-output). The total flow FTOT is defined to be the sum of the fixed flow requirements F_o^k over all commodities; i.e.

$$\text{FTOT} = \sum_{k=1}^{K} F_o^k .$$

That NCOM is an essential variable for experimentation deserves little explanation. Obviously in the decomposition code, this parameter determines the number of subproblems to be solved, the number of GUB constraints in the master problems, as well as the rate of growth of the columns in the master problem from one iteration to another. The role of FTOT may be explained as follows: Clearly the complicating mutual capacity constraints (8.2) interfere to the extent that commodities compete for capacity. If FTOT is small compared to the average arc capacity, the flow of each commodity may be sent on a single chain unimpeded by the capacity constraint, whereas if FTOT is large, the flow of a commodity k must be apportioned over several chains. Thus the total load on the network is expected to affect solution times.

In order to have some control on the interaction between the effects of varying NCOM and FTOT, the runs for each problem were divided into groups:

(a) Hold the total flow fixed while increasing the number of commodities. In this group of runs one would start with a few commodities (i.e. OD pairs) and then successively break down the flow requirement for each commodity into flow requirements for a number of new commodities. Such runs were motivated by the idea of aggregation of commodities: A commodity k (which

is specified completely by its origin-destination-OD-pair (s^k,t^k)) can be broken up into a number of "more detailed" commodities with OD pairs (s_i^k,t_i^k) for i=1, ..., n_k, each with a required flow value of F_i^k subject to the restriction

$$F_o^k = \sum_{i=1}^{n_k} F_i^k ,$$

where F_o^k is the required flow for commodity k. Obviously FTOT remains constant in such a process of disaggregation and the load pattern on the network does not change much either, since care will be taken to choose the s_i^k's and t_i^k's from the "vicinity" of s^k and t^k respectively.

(b) Increase the number of commodities increasing the total flow proportionally. Here a basic "per commodity" required flow value FAV is chosen and will be the same for all commodities k. Thus it is clear that we will have

$$F_o^k = \text{FAV for all k,} \qquad \text{and FTOT} = \text{NCOM} * \text{FAV.}$$

In this case the network traffic is becoming more congested as new commodities (OD pairs) are added.

We shall refer to these two strategies as (a) and (b) when discussing the computational results below.

Tables 3.5-3.9 summarize the results of our experimentation. All runs were made on the IBM 370/168 computer at the MIT Information Processing Center. The runs P1.1, P1.2, and P1.3 follow the abovementioned strategy

TABLE 3.5 TEST PROBLEMS, P1, P2, P3, P4

Problem Name	Number of Arcs (NARC)	Number of Capacitated Arcs (NCAP)	Number of Nodes (NN)	Average Capacity per capacitated arc (AVCAP)
P1	98	98	47	6.0
P2	96	96	25	13.0
P3	204	95	83	12.0
P4	268	98	44	16.0

TABLE 3.6 RESULTS FOR P1

Run Name	No. of Commodities (NCOM)	Total Flow (FTOT)	Optimal Value (VOPT)	Iterations to Optimality (ITOPT)	Time (sec.)
P1.1	3	28.0	486.0	7	.27
P1.2	5	28.0	478.0	5	.39
P1.3	7	28.0	492.0	6	.60
P1.4	6	30.0	518.0	7	.53
P1.5	10	30.0	498.0	5	.85
P1.6	15	30.0	497.0	5	.90
P1.7	3	28.0	482.0	12	.63
P1.8	3	28.0	470.0	8	.39

TABLE 3.7 RESULTS FOR P2

Run Name	No. of Commodities (NCOM)	Total Flow (FTOT)	Optimal Value (VOPT)	Iterations to Optimality (ITOPT)	Time (sec.)
P2.1	4	120.0	4836.0	6	0.50
P2.2	10	130.0	5364.0	7	1.73
P2.3	20	150.0	5852.0	6	4.31
P2.4	34	150.0	5840.0	4	3.13
P2.5	10	60.0	2304.0	1	0.04
P2.6	14	84.0	3192.0	1	0.06
P2.7	20	120.0	4522.0	3	0.57
P2.8	24	144.0	5602.0	4	1.37
P2.9	30	180.0	Infeasible	5	4.13

TABLE 3.8 RESULTS FOR P3

Run Name	No. of Commodities (NCOM)	Total Flow (FTOT)	Optimal Value (VOPT)	Iterations to Optimality (ITOPT)	Time (sec.)
P3.1	4	105.0	2271.0	10	1.51
P3.2	6	105.0	2269.0	10	1.41
P3.3	12	105.0	2213.0	9	3.48
P3.4	18	105.0	2088.0	6	3.11
P3.5	6	36.0	712.0	2	0.12
P3.6	12	72.0	1444.0	4	0.81
P3.7	18	108.0	2155.0	9	2.80

TABLE 3.9 RESULTS FOR P4

Run Name	No. of Commodities (NCOM)	Total Flow (FTOT)	Optimal Value (VOPT)	Iterations to Optimality (ITOPT)	Time (sec.)
P4.1	6	90.0	3640.0	2	0.20
P4.2	10	150.0	6006.0	4	0.58
P4.3	14	210.0	8858.0	5	1.44
P4.4	20	300.0	12488.0	3	1.70
P4.5	30	450.0	18162.0	4	3.98

(a) with a fixed value of total flow equal to 28.0. The same strategy is used in runs P1.4 through P1.6 with a total flow of 30.0. An interesting pattern presents itself in both cases. The number of iterations (number of master problems solved to reach optimality) -- denoted ITOPT -- decreases as the number of commodities increase for a fixed value of total flow. This change is due to the fact that the number of chains generated at each iteration is equal to the number of commodities. Thus many more chains are available at the end of a given iteration for a run with large NCOM compared to small NCOM. Moreover, the average flow per commodity decreases in strategy (a) as the number of commodities increases (F^k equals FTOT/NCOM on the average). Since each column in the master problem represents a chain with all the flow F^k assigned to it, capacity constraints are less likely to interfere when F^k grows smaller in comparison with the chain capacity.

The run P1.7 was included to test the effect of uneven loading on the network. We feel that a problem will require more iterations if the flow requirements are distributed in an uneven fashion, compared to a problem with even distribution of the same total flow value. In P1.8 we consider three OD pairs with flow requirements 9, 10, and 9 respectively. In P1.7 we change the requirements to 14, 12, and 2. The total flow is the same for both runs. We see that the number of iterations as well as the solution time increases substantially.

Table 3.7 presents the results of DCMP on P2. It may be recalled that the structure of P2 corresponds to an undirected graph. In the interest of symmetry we chose our OD pairs symmetrically as well. This means after

assigning an OD pair (s^k, t^k) to an odd-numbered commodity k, commodity k+1 is given the OD pair (t^k, s^k) with the same flow value F_o^k.

The series of runs P2.5-P2.8 follow strategy (b). The average per commodity flow value chosen is FAV = 6.0. Since the total flow value in runs P2.5 and P2.6 is small compared to the network capacity, the first master problem is found to be optimal. This shows the advantage of starting out with good chains in conjunction with a "Big-M" method. As the number of commodities, and hence the total flow, increases, more iterations are required.

Comparing runs P2.3 and P2.4 which share the same value of total flow, once again we see the effect of increasing NCOM in terms of decreasing the number of iterations. The runs for P2 show that ITOPT is not necessarily a good indicator of the solution time required by the problem. For example, P2.3 has a lower ITOPT than P2.2 but requires a much greater solution time. This, by the way, should be attributed to the SEXOP routine solving the master problem, which seems to slow down when NCOM is large (thereby having a large number of columns in each master as well as a greater number of GUB constraints); and not to the shortest chain routine solving the subproblems as well shall see.

In the runs for P3 strategies (a) and (b) were used in groups P3.1 - P3.4 and P3.5 - P3.7 respectively. Again, increasing the number of commodities with fixed total flow decreases ITOPT but the solution time tends to increase nevertheless. For the second group it can be seen that both ITOPT and the solution time increase with the number of commodities.

Finally for P4, strategy (b) was employed. We remark that the capacity constraints are rather sparse in this problem, in the sense that there is

a good chance of being able to find an uncapacitated chain between an arbitrary pair of nodes. P4 corresponds to an undirected graph as well. A per commodity flow value of 15.00 (=FAV) was used. We note a steady increase in solution times with the number of commodities.

In empirical timing of decomposition algorithms with a master problem interacting with subproblems possessing special structure, it is a good idea to investigate how time-consuming the subproblems are to solve. In addition to location the portion of the algorithm where most of the effort is expended, it throws some light on the effect of varying the number of subproblems -- NCOM in our case. For our Dantzig-Wolfe Decomposition code, the subroutine BELL used to solve the shortest chain problems should be timed. To this end the network and cost structures of P1 through P4 were used to obtain an average time per shortest-chain calculation for the three structures. The results are shown below:

Problem Name	Number of Arcs	Number of Nodes	Avg. Time (in 10^{-3} sec.)
P1	98	47	1.47
P2	96	25	1.25
P3	204	83	2.89
P4	268	44	3.03

Thus, for an average network of 200 arcs and 10 commodities which may require 10 iterations on the master level, we may expect an upper limit of 0.3 sec. on the total time expended in solving the shortest chain subproblems. Given that the total time for the solution of such a problem may be 2-3 seconds; this is not a substantially time-consuming activity of the code.

As a further illustration of the use of this information, we compare the runs P2.2 and P2.3. The number of shortest path problems solved in each run is (ITOPT+1) · NCOM · T, where T is the average time per shortest chain calculation. Thus in going from P2.2 to P2.3 a total time increase of 60T = 0.075 seconds may be attributed to the additional time spent in solving shortest path problems. This shows that increasing NCOM affects the LP solution time much more than the subproblems.

We wish to conclude by noting that our price-directive decomposition code has not yet been subjected to a second level effort aimed at shortening its running time. We can easily visualize detailed portions of the code to which such an effort could be applied. In particular, the subroutine GENCOL could be much more efficient in using the inputs from BELL to generate columns. Also, one may experiment with different strategies for column generation. It may be advantageous to have added columns replace some other nonbasic columns in the master problem. Another possibility is to ignore a subproblem which has priced optimally at a certain iteration for a number of subsequent iterations, since the set $\bar{S}_t$ of subproblems optimal at iteration t, does not change much from one iteration to the next.

3.4.2 Resource-Directive Decomposition

In this subsection we report computational experience with the resource-directive algorithm, some of which directly affected the choice of our particular implementation of the subgradient algorithm. The algorithm we presented in subsection 3.1.2 has a number of controllable

factors which we may vary when searching for the best algorithmic performance. These factors may be listed as:

1) Choice of an initial capacity allocation $y^o \in S$.

2) Choice of the artificial costs c_o^k in the subproblem objective function.

3) Choice of the estimate $\hat{v}$ to be used in (30).

4) Choice of the step-size factors $\{\gamma_i\}$ in (30).

Most of our computational experimenting focused on issues 3 and 4 of the above. We shall comment on 1 and 2 rather briefly.

1) The initial capacity allocation $y^o = (y^{1,o}, \ldots, y^{K,o})$ was set by the following simple formula:

$$y^{k,o} = \frac{F_o^k}{FTOT} \cdot d \ , \tag{40}$$

where FTOT is defined, as before, to be

$$FTOT = \sum_{k=1}^{K} F_o^k . \tag{41}$$

This choice is not a very sophisticated one but is definitely easy to compute. A more complex alternative would involve solving each of the subproblems with the original arc capacities (i.e. $y^k = d$), thus obtaining the solution to the multicommodity problem with the mutual capacity constraints (13.3) relaxed in favor of single-commodity constraints $f^k \leq d$ for all k. This "interaction-free" flow pattern can then be used as a basis for determining y^o. This choice seems attractive, particularly

when the flow-carrying chains for different commodities are expected to be essentially diverse from each other, since one may then expect the optimal flow pattern to be close to the interaction-free pattern discussed above.

2) The artificial costs c_o^k reflect the extent to which we are encouraging the variable F^k to achieve its upper bound F_o^k. Essentially we chose c_o^k values consistent with the artificial costs in the Dantzig-Wolfe decomposition algorithm. That is, we chose the costs per unit flow c_o^k to be around 10-20 times the average routing cost per unit flow. We had only one occasion to test the sensitivity of our code to this choice: When running RHSD on Problem P1.5 with 10 commodities we noticed that many flow values F^k were consistently below their upper bound ($F^k < F_o^k$) with a value of 200 for c_o^k. To encourage greater values of F^k, c_o^k was increased to 2000, and as a result, 4 commodities attained the required flow value F_o^k. The ratio VBEST/v* (where VBEST denotes the best value obtained for the master problem) also increased from .79 to .88. We also found that increasing c_o^k leads to a decrease in the values of t_i's. However, the general behavior of the algorithm is not affected by the choice of c_o^k.

3) The subgradient algorithm requires an underestimate $\hat{v}$ of the optimal value v* to be used in the step-size formula (30). Held, Wolfe, and Crowder, however, report using an overestimate $\hat{v} > v^*$ for lack of a readily available underestimate. In our case, too, we decided to use an underestimate. We recall from subsection 3.1.2 that

$$v^* = \sum_{k=1}^{K} c_o^k \cdot F_o^k - v_1^* , \qquad (42)$$

where v_1^* are the minimal routing costs defined in (13.1). Assuming that c_o^k's are chosen to be equal ($c_o^k = c_o$ for all k), we may write (33) as

$$v^* = c_o \cdot FTOT - v_1^* . \tag{43}$$

We see that by (43), any underestimate $\hat{v}_1$ of v_1^* ($\hat{v}_1 < v_1^*$) will define an overestimate $\hat{v} = c_o \cdot FTOT - \hat{v}_1$ of v^*. A very crude overestimate may be obtained immediately by choosing $\hat{v}_1 = 0$, so that

$$\hat{v} = c_o \cdot FTOT . \tag{44}$$

A much better estimate may be obtained by solving the interaction-free version of (13.1) - (13.3) discussed above, where (13.3) is replaced by $f^k \leq d$ for all k. Denoting the objective function thus defined by $\hat{v}_D$, we clearly have $\hat{v}_o < v_1^*$ and obtain an overestimate

$$\hat{v} = c_o \cdot FTOT - \hat{v}_o . \tag{45}$$

We have experimented with both types of overestimates (44) - (45) and obtained significantly better results with the tighter overestimate in (45). In two otherwise identical runs on P1.1, the best value for the master (VBEST) increased from 5097.4 to 5110 in 80 iterations as a result of using (45) instead of (44) ($v^* = 5114$ for this run). The improved behavior of the algorithm is especially apparent when values close to v^* are obtained.

4) A critical issue in using a step-size sequence defined by (29)

and (30) is the choice of the λ_i's values. Held, Wolfe, and Crowder [3-16] state that the choice of step sizes is imperfectly understood. The strategy they used for the multicommodity max flow problem was to set $\lambda = 2$ for 2N iterations (where N depends on problem size) and then halving both the values of λ and the number of iterations until the latter reaches a threshold value N_o, whereupon λ is halved every N_o iterations.

In order to understand the behavior of the algorithm for different choices of the $\{\lambda_i\}$ sequence, we made a number of runs for the problem P1.1. Table 3.10 summarizes the results. We shall now comment on the results according to the general strategy used for defining the λ_i's:

Constant λ.

We initially tried runs with a constant λ and obtained rapid convergence for small test problems with 10 and 26 arcs (2 and 4 commodities respectively). However, for the larger P1.1 problem (of 98 arcs) runs 1 and 2 show that such a strategy does not result in values close enough to the optimum of 5114. An overestimate of 5600 was chosen according to (44), thus the values for VBEST are expected to improve if we use $\hat{v} = 5116$ following (45).

Successive Halving of λ.

Here we followed the suggestion of Held, Wolfe, and Crowder [3-16] discussed before. Runs 3 and 4 show marked improvement over runs 1 and 2 though they also use $\hat{v} = 5600$. Note that comparing runs 3 and 4 shows the iteration numbers chosen for halving λ will also affect the best objective function value obtained. The improvement of run 5 over run 4 is due only to employing the better $\hat{v}$ value of 5116.

TABLE 3.10 STEP SIZE STRATEGIES FOR RHSD

Run Name	Strategy	Number of Iterations	Best Value Obtained	Iteration Number of Best Value
1	= 2 constant	120	5084.9	54
2	= 1 constant	60	5080.6	52
3	halved at 60 & 90	120	5088.2	70
4	halved at 50, 75	80	5097.4	73
5	halved at 50, 75, 90 (used better v)	120	5112.4	120
6	Adaptive with = 0.01	120	5103.7	120
7	Adaptive with = 0.3	120	5112.9	95
8	Adaptive with = 0.5	120	5111.8	120

Adaptive Control of λ.

The above strategy of halving λ depends upon the choice of a sequence of iteration numbers at which λ is halved. The choice of such a sequence (or N and N_o in our earlier discussion) requires some hindsight which is problem-dependent. We shall now consider a scheme which ensures decreasing the size of λ and does not require the prespecification of such a sequence.

To motivate our adaptive strategy, we present the interation values and step sizes resulting from a constant $\lambda = 2$ value in Table 3.11. The last column represents the ratio of current t_i to the last step size obtained. We see from Table 3.11 that as soon as a good value of v_i, the value of the objective function at iteration i ($v_i \equiv v(y^i)$), is obtained for the master problem, the corresponding t_i becomes relatively large. This is probably due to the fact that most γ_a^k's become small or vanish, thus making the denominator of (30) small. The net effect is that the next perturbation in the capacity vector y^i will be too large, causing it to move away to a bad value, resulting in a sudden drop in the objective function value v_i. This happens at iterations 29, 35, and 54 of Table 3.11. It is thus necessary that λ_i's get smaller with increasing i to offset such large fluctuations. This phenomenon in part explains the efficacy of the halving strategy discussed above. The adaptive strategy takes two measures to control the step sizes:

a) It decreases the current t_i when it is deemed to have too large a value.

b) It decreases the value of λ be used in the successive steps.

A closer examination of run number 1 (see Table 3.8) shows that the value of v_i decreases significantly when the ratio t_i/t_{i-1} becomes large.

TABLE 3.11 RESULTS OF RUN 1 OF RHSD

Iteration	v_i	t_i	t_i/t_{i-1}
1	2494.2	0.039	-
10	4312.8	0.027	1.70
20	4813.4	0.017	1.74
28	4934.1	0.005	0.35
29	5080.7	1.927	368.09
30	2421.4	0.017	0.01
34	4927.0	0.009	0.53
35	5076.9	1.718	191.0
36	2661.2	0.011	0.006
53	4743.9	0.027	0.64
54	5084.9	7.518	278.45
55	1440.2	0.018	0.002

This suggests using this ratio as a tracking signal for the adaptive strategy. If the ratio $t_i/t_{i-1} \geq R$ (some threshold); reset

$$t_i \leftarrow a \cdot t_i \tag{46}$$

$$\lambda \leftarrow \alpha \cdot \lambda. \tag{47}$$

Some results for adaptive strategies are shown in runs 6-8. In all cases $\hat{v} = 5600$, and $a = 0.01$. α should not be too small since then two or three triggerings of the scaling operation (47) would result in an overly small value of λ resulting in a very small sequence t_i. This, in turn, would mean that each time the vector y^i is perturbed by an insignificant amount, so that the v_i values will change very slowly with i. The choice of $\alpha = 0.01$ in run 6 resulted in this behavior.

To us, the adaptive strategy seems to be a computationally attractive alternative to the halving strategy. In all cases it resulted in a better sequence of values v_i which came closer to the optimum v^*.

Timing the Projection Routine

As in the case of the Dantzig-Wolfe decomposition code, we found it useful to time the subroutine which solves the projection problem to identify whether much time is spent performing the projections.

Recall that the input parameters for PRJCT were NCOM (or K) and d_a, the sum of the components of the projection vector y_a. In timing PRJCT we varied NCOM and set d_a = NCOM. Thus we project K-vectors on the hyperplane defined by

$$\{x \in R^K \mid \sum_{k=1}^{K} x^k = K \mid x^k \geq 0 \text{ for all } k\} .$$

The results are shown below:

NCOM	3	5	10	20	50
Average time (in 10^{-4}sec.) per projection	1.13	1.92	4.08	9.34	27.08

As remarked earlier, not all arcs are projected. Let NP_i be the number of arcs projected at iteration i. We observed that NP_i/NARC decreases with i once we enter a reasonable neighborhood of the optimum. Again this is due to many γ_a^k's vanishing once the optimum is neared. Also one would expect NP_i to depend on NCOM: the more commodities there are, the greater the probability of γ_a^k being nonzero for some k.

In a test problem with 26 capacitated arcs and 4 commodities we computed an average of $\overline{NP_i} = 15$ for $1 \leq i \leq 10$, and $\overline{NP_i} = 8$ for $10 \leq i \leq 19$. The algorithm converged in 19 iterations. Assuming the ratio NP_i/NARC to be 0.5, at worst, for the 3-commodity problem P1.1, we would have to spend approximately 0.7 seconds for projections in the course of 120 iterations. We have timed RHSD on P1.1 to take approximately 6 seconds. So approximately 12 percent of the solution time is attributable to projections.

General Behavior of the Algorithm

We have described the effect of various strategies on the convergence exhibited by the right-hand side allocation algorithm (RHSD). We noted that with a good overestimate $\hat{v}$ and an adaptive strategy for the step size one may get close approximations to the optimal value v^*. It may be argued,

however, that the output of interest in the optimal flow pattern f^{k*} for each commodity k, rather than merely the optimal value. Moreover, if the capacities are integral we would like to obtain integral values for the optimal flow of each commodity on each arc. The Dantzig-Wolfe decomposition code exhibits this property regularly. Unfortunately resource-directive decomposition will generally fail to do this, since the arc capacities allocated to each commodity result from a projection and may rarely be expected to be integral. To give only one example, the optimal flow pattern obtained by Dantzig-Wolfe decomposition on P1.1 for commodity 1 loads three chains C_1, C_2, and C_3 with chain flows 5, 3, and 1 respectively. The load pattern obtained from the best solution to P1.1 with RHSD places the values 4.4618, 2.9937, and 1.5445 on C_1, C_2, and C_3; and in addition chooses a fourth chain C_4 with a flow value of .0063. Thus RHSD does not yield integral solutions though it essentially chooses the correct chains. Here there may be some hope of reallocating the flows, once the chains are identified, to achieve integrality if needed.

The behavior of RHSD worsens as the number of commodities increases. For 10 commodities the objective function value is far from the optimum and half the commodities fail to meet the flow requirements. Thus RHSD should be limited to networks with a small number of commodities although largest network sizes are not expected to affect the convergence of RHSD much. The solution times are of the order of 6 seconds for 120 iterations on a 3-commodity version of P1.

Postscript

Upon completion of the computational experience on which this subsection is based [3-5], we discovered two papers by Kennington who has solved multicommodity capacitated transportation problems by two algorithms: a primal GUB approach [3-17] and a subgradient procedure similar to ours [3-18]. The networks he experiments with are somewhat smaller, but the results for the subgradient algorithm generally resemble our conclusions.

REFERENCES FOR SECTION 3.

3-1 Aashtiani, H.Z., "Solving Large Scale Network Optimization Problems by the Out-of-Kilter Method," M.S. Thesis, MIT Sloan School of Management, February 1976.

3-2 _____, and T.L. Magnanti, "Implementing Primal-Dual Network Algorithms," Working Paper OR 055-76, Operations Research Center, MIT, June 1976.

3-3 Assad, A.A., "Multi-commodity Network Flows - A Survey," Working Paper OR 045-75, Operations Research Center, MIT, December 1975.

3-4 _____, "Multicommodity Network Flows - Computational Experience," Working Paper No. OR 058-76, MIT Operations Research Center, October 1976.

3-5 _____, "Solution Techniques for the Multicommodity Flow Problem", unpublished M.S. Thesis, MIT Operations Research Center, June 1976.

3-6 Barr, R.S., F. Glover, and D. Klingman, "An Improved Version of the Out-of-Kilter Method and a Comparative Study of Computer Codes," Math. Prog., 7, 50-86, 1974.

3-7 Cohen, R., Term paper for the course "Flight Transportation Analysis," May 1972.

3-8 Florian, M. et al, "The Engine Scheduling Problem in a Railway Network," INFOR J., 14, 121-138, 1976.

3-9 Geoffrion, A.M., "Primal Resource Directive Approaches for Optimizing Nonlinear Decomposable Systems," Oper. Res., 18, 375-403, 1970.

3-10 Gilsinn, J. and C. Witzgall, "A Performance Comparison of Labelling

Algorithms for Calculating Shortest Path Trees," National Bureau of Standards, Technical Note 772, May 1973.

3-11 Glover, F., D. Karney, D. Klingman, and A. Napier, "A Computations Study on Start Procedures, Basis Change Criteria, and Solution Algorithms for Transportation Problems," Man. Sci. 20, 5, 793-813, 1974.

3-12 _____, D. Klingman, and J. Stutz, "An Augmented Threaded Index Method for Network Optimization," INFOR, 12, 293-813, 1974.

3-13 Golden, B, "Shortest Path Algorithms: A Comparison,"(to be published in Oper. Res.).

3-14 Grinold, R.C., "Steepest Ascent for Large Scale Linear Programs," SIAM Review, 14, 3, 447-463, July 1972.

3-15 Held, M. and R.M. Karp, "The Traveling-Salesman Problem, Part II," Math. Prog., 1, 1, 6-25, Oct. 1971.

3-16 Held, M., P. Wolfe, and H.P. Crowder, "Validation of Subgradient Optimization," Math. Prog., 6, 1, 62-88, February 1974.

3-17 Kennington, J.L., "Solving Multicommodity Transportation Problems Using a Primal Partitioning Simplex Technique," Dept. of Ind. Eng. & Oper. Res., Southern Methodist Univ. Tech. Report CP 75013, revised May 1976.

3-18 _____, and M. Shalaby, "An Effective Subgradient Problem for Minimal Cost Multicommodity Flow Problems," Southern Methodist Univ. Tech. Report IEOR 750010, revised March 1976.

3-19 Lasdon, L.S., Optimization Theory for Large Systems, Section 3.2, Macmillan, New York, 1970.

3-20 Levin, A, "Some Fleet Routing and Scheduling Problems for Air Transportation Systems," Flight Transportation Lab Report FTL-R68-5, MIT, January 1969.

3-21 Magnanti, T.L., "Optimization for Sparse Systems," pp. 147-176 in Sparse Matrix Computations (J.R. Bunch and D.J. Rose, eds.), Academic Press, NY, 1976.

3-22 Marsten, R.E., "Users Manual for SEXOP," Release 4, MIT Sloan School of Management, February 1974.

3-23 _____, "The Use of BOXSTEP Method in Discrete Optimization," Math. Prog. Study 3, 1975.

3-24 _____, W.W. Hogan, and J.W. Blankenship, "The BOXSTEP Method for Large-Scale Optimization," Oper. Res. 23, 389-405, 1975.

3-25 Mulvey, J, "Developing and Testing a Truly Large Scale Network Optimization Code," presented at ORSA/TIMS National Meeting, Chicago, May 1975.

3-26 Nijenhuis, A. and H.S. Wilf, Combinatorial Algorithms, Academic Press, NY, 225-231, 1975.

3-27 Oettli, W., "An Iterative Method, Having Linear Rates of Convergence, for Solving a Pair of Dual Linear Programs," Math. Prog. 3, 302-311, 1972.

3-28 Simpson, R.W., "Scheduling and Routing Models for Airline Systems," Flight Transportation Lab Report FTL-R68-3, MIT, December 1969.

3-29 Swoveland, C.; "Decomposition Algorithms for the Multicommodity Distribution Problem", Western Management Science Institute, Univeristy of California (Berkeley), ORC 72-5, February 1972.

3-30 Tomlin, J.A.; "Minimum Cost Multicommodity Network Flows", Oper.Res. 14, 45-51, 1966.

4. FREIGHT FLOW PROBLEMS

In Section 2 we presented mixed integer programming formulations for a series of freight flow models, ranging from the uncapacitated depot model to more elaborate models incorporating multicommodity distribution, capacitated depots, and congestion costs. As noted, we can view these models conceptually as network design problems; accordingly, we can use computational techniques which have been developed for this class of problems as planning tools for freight flow management. In this section, we describe the initial efforts in an ongoing research program aimed at evaluating the potential of this approach.

We have used a decomposition technique, known as Benders decomposition, as the focal point of our analysis. This method has been used most successfully in the past for designing an industrial distribution system [4-5] and has been applied to scheduling railway engines [4-2] and also to routing aircraft [4-10].

These applications, which share many of the combinatorial complexities involved in freight flow management, suggest that Benders Method might be attractive for transportation system design.

When applied to network design problems, Benders algorithm is a decomposition technique that, in essence, decouples the routing decisions from the design decisions. This decomposition leads to a series of linear programs and integer programs that are solved alternately. The linear program decides upon the best routing strategy with respect to the given design. The integer program uses the routing solution to guide the choice of a new design.

Rather than applying Benders algorithm as it has been used in the

past, we propose new methodology for accelerating the algorithm. This methodology should result in fewer integer programs being solved within the iterative scheme of Benders Method, and therefore should help to mitigate part of the greatest computational burden of this algorithmic approach.

To test this methodology, we have applied the modified Benders algorithm to several p-median location models; we also illustrate the method on a small uncapacitated network design model (further experimentation is underway for this class of problems). The p-median and uncapacitated network design models provide good testbeds for assessing our methodology for several reasons:

1) They exhibit the combinatorial complexities inherent in network design applications like freight flow management problems. New theory emerging in computer science [4-7,4-8,4-12] shows that these models are as difficult to solve as more elaborate network design models;

2) A number of test problems in the Transportation Science and Operations Research literatures are readily available; and

3) Because of their relative simplicity, the p-median and uncapacitated network design models are easier to work with than more intricate models in the context of a research program. Programming solution techniques and gathering data for these models, while still far from trivial, are unencumbered by the details involved in more complicated models.

For these reasons, we feel that analyzing these two models is a good starting point for our research.

For completeness and to set notation, we begin by reviewing Benders decomposition for mixed integer programs. We then introduce modifications to the procedure, which apply, in fact, to any "cutting plane" type algorithm

such as Benders decomposition or Dantzig-Wolfe decomposition. In the succeeding subsections, we show how this methodology specializes for the p-median and uncapacitated network design problems to exploit the network structure of these models. The final part of this section summarizes our computational experience to date.

4.1 BENDERS DECOMPOSITION

Each of the network design models formulated in Section 2 can be stated in matrix notation as mixed integer programs of the form:

$$
\begin{aligned}
v \equiv \text{Min} \quad & cx + dy \\
\text{subject to:} \quad & Ax + Dy = b \\
& x \geq 0 \\
& y \in Y ,
\end{aligned}
$$

where the n-dimensional column vector x and the k-dimensional column vector y are problem variables; the row vectors c and d and the column vector b are given data with dimensions n, k and m, respectively; the m by n matrix A and m by k matrix D are given data as well. The set Y captures the integer (or configuration) features of the problem; typically

$$Y = \{y \in R^k : y \geq 0 \text{ and integer}\}$$

although the set Y could include side constraints imposed upon the integer variables. For example, for the p-median problem

$$Y = \{ y \in R^k : \; y_1 + y_2 + \ldots + y_k = p,$$

$$y \geq 0 \text{ and integer} \} .$$

Benders algorithm decomposes the mixed integer problem by viewing the model sequentially as first choosing the "configuration" variables y and then choosing the continuous variables x; that is, by rewriting the mixed integer program in the equivalent form:

$$v = \min_{y \in Y} \; \min_{x \geq 0} \; \{ (cx + dy) : Ax = b - Dy \} .$$

The advantage of this formulation is that the inner minimization over the continuous variables is a linear program which usually will be <u>much</u> easier to solve than the original mixed integer program; it will, in fact, be a network flow problem for many of the applications that arise in transportation planning.

If we make the simplifying assumption that this linear program is feasible and has an optimal solution (i.e. its objective function is bounded from below over the feasible region) for every choice of $y \in Y$, then we can replace the linear program by its dual* and rewrite the problem in the equivalent form:

$$\min_{y \in Y} \; \max_{u} \; \{ [dy + u(b - Dy)] : uA \leq c \} . \qquad (1)$$

---

* Minor algorithm changes are required if we relax these assumptions; see for example [4-9]. Moreoever, the methodology extends to nonlinear problems; see [4-4].

Since the inner minimization will always have a solution at one of its extreme points $u^1, u^2, \ldots, u^K$ of its feasible region $\{u \in R^m: uA \leq c\}$, we can restate the problem as:

$$\underset{y \in Y}{\text{Minimize}} \quad \underset{1 \leq j \leq K}{\text{Maximize}} \{dy + u^j(b - Dy)\}, \tag{2}$$

or in the alternative form:

$$\begin{aligned} \dot{v} = \underset{y \in Y}{\text{Minimum}} \; & Z \\ \text{subject to } Z \geq & \; dy + u^j(b - Dy) \qquad j = 1, 2, \ldots, K. \end{aligned} \tag{3}$$

For any given choice of y, the constraints in this formulation ensure that Z is at least as large as each of the numbers $dy + u^j(b - Dy)$ for $j = 1, 2, \ldots, K$. The minimization over Z then insures that Z equals the maximum of these numbers. Consequently problem (3) is equivalent to problem (2).

Observe that problem (3) is an integer program in the variables y (and the single continuous variable Z), which has one constraint for each of the extreme points $u^1, u^2, \ldots, u^K$. Since in general the number of such extreme points will be enormous, it is not possible to solve this problem directly. Benders approach is to relax most of the constraints and solve the following approximation to (3):

$$v^J = \underset{y \in Y}{\text{Minimum}} \; Z$$

$$\text{subject to } Z \geq dy + u^j(b - Dy) \quad j \in J \,, \tag{4}$$

where J is a "small" subset of the set {1, 2, ..., K}. This problem is then much more amenable to solution by integer programming algorithms.

If the solution $\hat{Z}$, $\hat{y}$ to the relaxed problem (4) satisfies each of the constraints in problem (3), that is

$$\hat{Z} \geq d\hat{y} + u^j(b - D\hat{y}) \quad \text{for all } j = 1, 2, \ldots, K \tag{5}$$

then this solution solves (3) as well. Consequently, $y = \hat{y}$ is the vector of optimal integer variables for the original mixed integer program. With these in hand, we can easily solve for x in the original program by linear programming, or equivalently solve the inner minimization (an LP) in (1) which has x as its dual variables.

If the solution $\hat{Z}, \hat{y}$ to the mathematical program (4) does not satisfy each of the constraints (5), then we have not solved problem (3). We append the most violated constraint in problem (3) at $Z = \hat{Z}$ and $y = \hat{y}$ to problem (4) and repeat the procedure. Since the number of constraints in (3) is finite, the procedure must terminate eventually, possibly when we have exhausted each of the extreme points $u^1, u^2, \ldots, u^K$ by appending the corresponding constraints to the index set J in problem (4).

Observe that to check condition (5) or determine the most violated constraint when $Z = \hat{Z}$ and $y = \hat{y}$, we must determine

$$v(y) \equiv \underset{1 \leq j \leq K}{\text{Maximum}} \{dy + u^j(b - D\hat{y})\}. \tag{6}$$

If $v(\hat{y}) \leq Z$, then condition (5) is satisfied; otherwise, the solution u^k to problem (6) corresponds to the most violated constraint

$$\hat{Z} < d\hat{y} + u^k(b - D\hat{y}),$$

and k is added to the index set J in problem (4). The new constraint

$$Z \geq dy + u^k(b - Dy)$$

added to the problem is called a Benders cut.

Since solving problem (6) is equivalent to finding an extreme point solution to the linear program:

$$v(\hat{y}) \equiv \text{Maximum}\{[d\hat{y} + u(b - D\hat{y})] : uA \leq c\}. \tag{7}$$

Benders algorithm alternates by solving the integer programs (4) and linear programs (7) until the optimality condition (5) is satisfied.

Since problem (4) relaxes some of the constraints of (3), we know that the value v^J to problem (4) underestimates the value v to the original problem. Also, $\hat{y}$ in problem (7) together with the optimal dual variables

$\hat{x}$ to this linear program are feasible in the original problem. Consequently, we have the bounds

$$v^J \leq v \leq v(\hat{y}) \quad \text{for each } J \text{ and } \hat{y}.$$

Knowing these bounds, we may terminate prior to finding an optimal solution with an error bound on our solution.* This feature of the algorithm is useful since the values v^J may "tail-off" when approaching v, coming close to v quickly and then increasing slowly to achieve the final improvements.

There are several ways to modify this statement of the algorithm. For example, it may not pay to solve the integer problem to completion at each stage. Rather we can search for a solution $\hat{y}$ that simply improves upon the last value v^J generated at the previous step, possibly by setting a number $\varepsilon > 0$ as the target value for the level of improvement. Geoffrion and Graves [4-5]discuss several implementation devices such as this.

4.2 ACCELERATING BENDERS DECOMPOSITION

In this section we describe a method for accelerating Benders algorithm. The modification applies when the subproblem (7) has multiple optimal solutions, which results from degeneracy in its dual problem:

$$\text{Minimize}_{x \geq 0} \{(d\hat{y} + cx) : Ax = b - D\hat{y}\}. \tag{8}$$

---

* Since the set J increases from step to step, the bounds v^J increase monotonically to v. The bounds $v(\hat{y})$ need not be monotonic, though, and we may choose the smallest such value generated so far in this iterative procedure as the upper bound on v.

Because network optimization problems are reknowned for their degeneracy, we can expect the algorithmic modifications that we propose to be applicable to many of the models used for transportation planning.

4.2.1 Example

Before presenting the theory of our methodology, let us illustrate the phenomenon that leads to our proposal in a numerical example. Consider the optimization problem with a single integer variable y:

$$v = \text{Minimum} \quad 3x_1 + 3x_2 + x_3 + x_4 + x_5 + 4x_6$$

$$\text{subject to:} \quad 3x_1 + x_2 + x_3 - x_4 - x_5 + x_6 = 4 - y$$

$$8x_1 + 8x_2 + 2x_3 - x_4 + x_6 = 8 - y$$

$$x_j \geq 0 \quad (j = 1, 2, \ldots, 6), \; y \geq 0 \text{ and integer.}$$

In this case, $d = 0$ and $D = \binom{1}{1}$ are the coefficients of the integer variable in the objective function and constraints. Y is given by $\{y : y \geq 0 \text{ and integer}\}$.

Graphing the feasible region to the dual of this problem, we find that it has five extreme points:

Extreme Point u^j	Constraint Coefficient $Z \geq dy + u^j(b - Dy)$
$u^1 = (0, 3/8)$	$Z \geq 3 - 3/8\ y$
$u^2 = (1, 0)$	$Z \geq 4 - y$
$u^3 = (7, -3)$	$Z \geq 4 - 4y$
$u^4 = (-1, 1/2)$	$Z \geq 1/2\ y$
$u^5 = (-1, 0)$	$Z \geq -4 + y$

Figure 4.1 illustrates the feasible region to the integer programming problem (3), i.e.

$$v = \underset{y \in Y}{\text{Minimum}}\ Z$$

$$\text{subject to } Z \geq dy + u^j(b - Dy) \quad (j = 1, 2, \ldots, 5),$$

for this example. The region R above the highlighted curve v(y) corresponds to points Z and y satisfying the inequalities in this problem. The point (Z,y) in this region with y integer and Z as small as possible is a solution, i.e. $(Z,y) = (\frac{15}{8}, 3)$ solves the problem.

Suppose that we initiate Benders algorithm with the single extreme point $u = u^4$ and the corresponding constraint $Z \geq \frac{1}{2} y$ in the relaxed problem (4). The solution to the relaxed problem is $\hat{y} = 0$ and $\hat{Z} = 0$. At the point $\hat{y} = 0$ both the extreme points u^2 and u^3 solve the linear programming subproblem (7). We can visualize this situation in Figure 4.1 by looking along the Z-axis to find the constraint $Z \geq dy + u^j(b - Dy)$ most violated by the point $(\hat{y} = 0, \hat{Z} = 0)$. Both the lines $Z = dy + u^2(b - Dy)$ and $Z = dy + u^3(b - Dy)$ intersect the Z-axis at $Z = 4$ and are most violated. Therefore, in this case we have two optimal solutions, u^2 and u^3, to the linear programming subproblem (7). We also see that the optimal solution $x_3 = 4$, $x_1 = x_2 = x_4 = x_5 = x_6 = 0$ to the linear program (8) with $\hat{y} = 0$ is degenerate, since only one basis variable x_3 is positive.

We now have two choices for the Benders cut to add to the relaxed problem, either that associated with u^2, or that associated with u^3. (We could add both for this simple example; in general, though, we won't know all the optimal solutions to the linear programming subproblem [4-7]).

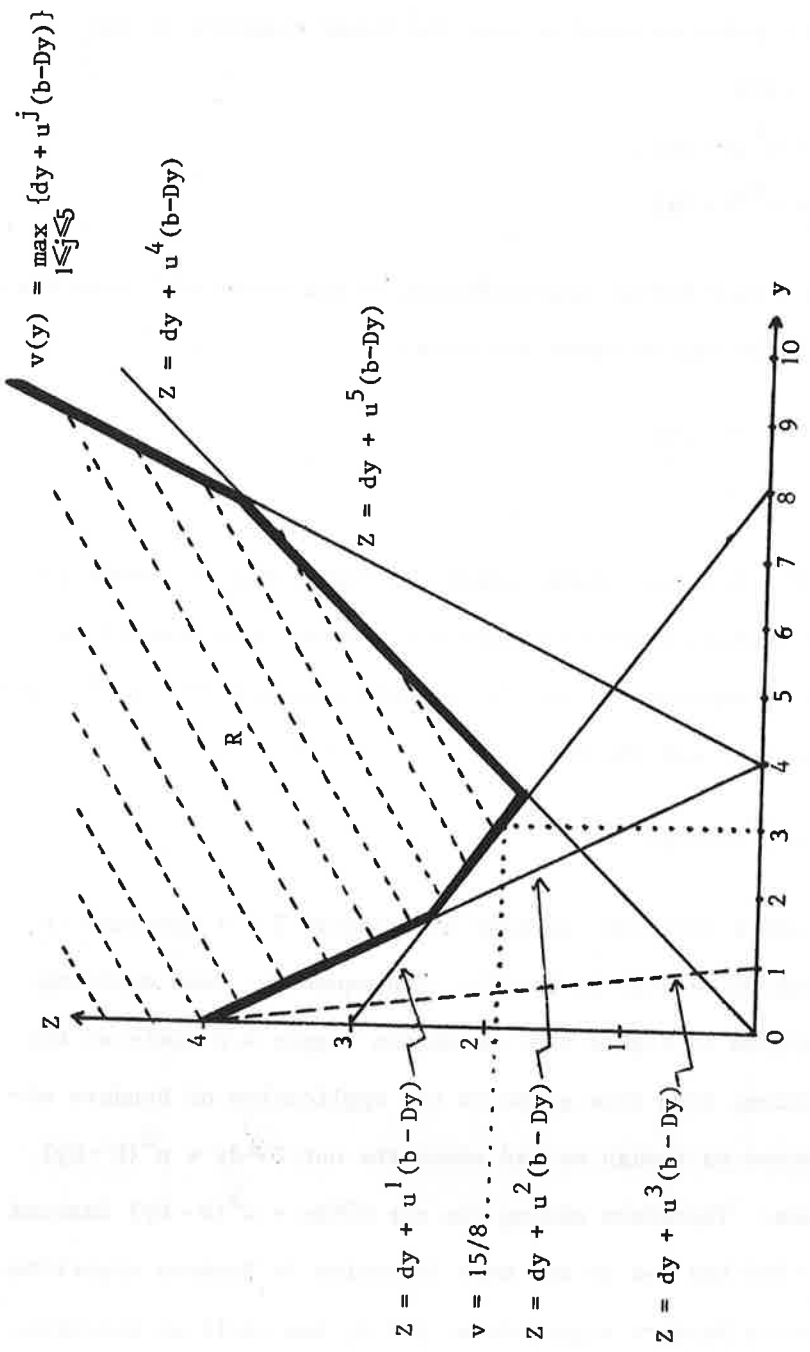


FIGURE 4.1 STRONG CUTS IN BENDERS DECOMPOSITION

Figures 4.2 and 4.3 show the relaxed problems obtained after adding each of these alternative cuts.

The important point to note is that the lower boundary of the region above the cuts

$$Z \geqslant dy + u^2(b - Dy),$$

and $$Z \geqslant dy + u^4(b - Dy)$$

in Figure 4.1 is a much better approximation to the curve v(y) than the lower boundary of the region above the cuts

$$Z \geqslant dy + u^3(b - Dy),$$

and $$Z \geqslant dy + u^4(b - Dy),$$

in Figure 4.3. In fact, the shaded region in Figure 4.2 is actually contained in the shaded region in Figure 4.3. Also, note that if we continue Benders decomposition from the optimal solution ($\hat{y} = 1$, $\hat{Z} = 1/2$) in Figure 4.3, we next add the cut

$$Z \geqslant dy + u^2(b - Dy)$$

to the problem, since this cut is most violated at $\hat{y} = 1$ (because it coincides with the curve v(y) at $\hat{y} = 1$). Consequently, when applying Benders decomposition to Figure 4.3, we obtain Figure 4.2 again as the next relaxed problem; from this point on the application of Benders algorithm will proceed as though we had added the cut $Z \geqslant dy + u^2(b - Dy)$ in the first place. Therefore adding the cut $Z \geqslant dy + u^3(b - Dy)$ instead of $Z \geqslant dy + u^2(b - Dy)$ has led to one more iteration in Benders algorithm.

When we started Benders algorithm at $\hat{y} = 0$, how could we determine

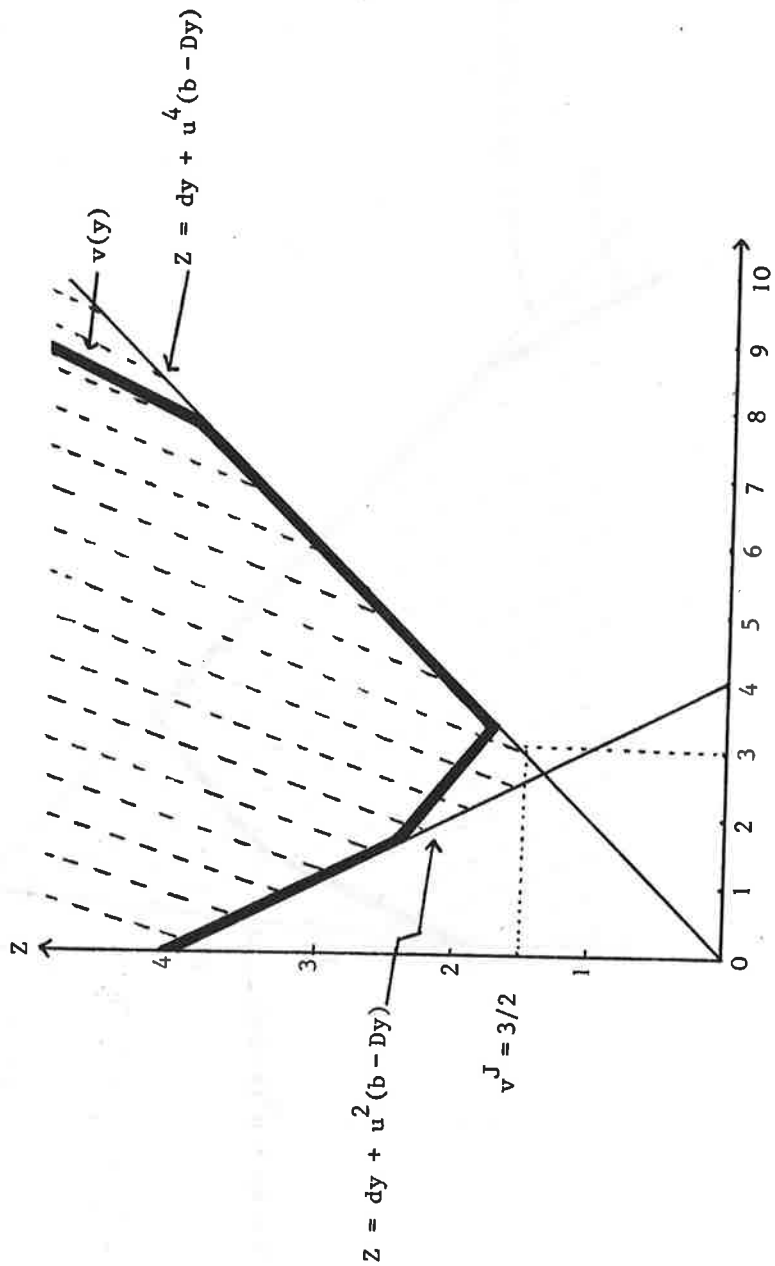


FIGURE 4.2 ADDING A CUT FOR EXTREME POINT u^2

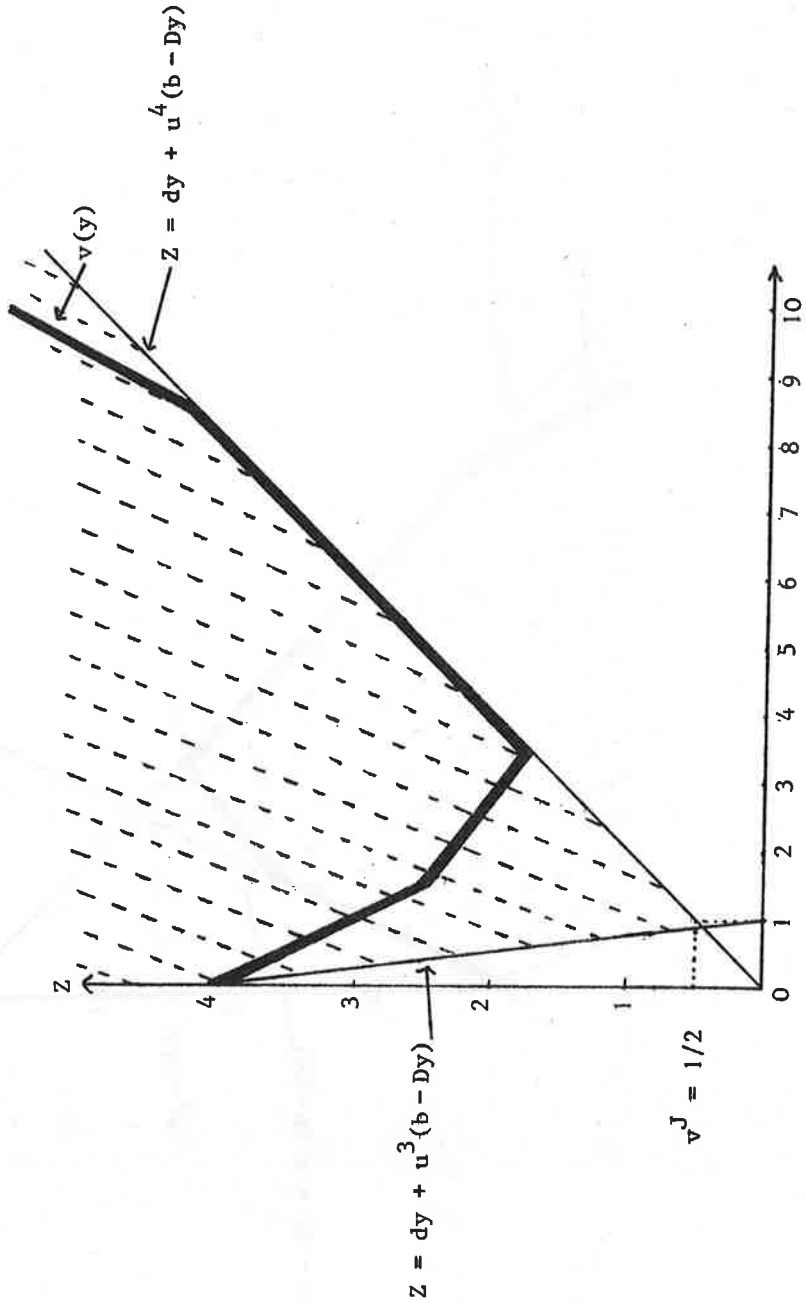


FIGURE 4.3 ADDING A CUT FOR EXTREME POINT u^1

that the cut associated with u^2 is preferred to the cut associated with u^3 ? Looking at Figure 4.1 gives the answer, at least geometrically. Note that the line

$$Z = dy + u^2(b - Dy) \tag{9}$$

always lies on or above the line

$$Z = dy + u^3(b - Dy) \tag{10}$$

in the region $y \geqslant 0$. Therefore whenever

$$Z \geqslant dy + u^2(b - Dy)$$

in this region, we know that

$$Z \geqslant dy + u^3(b - Dy)$$

as well; the second cut gives only redundant information. Note that we can determine this fact geometrically in Figure 4.1 as follows. Both of the lines (9) and (10) cross the Z-axis at the same point $\hat{Z} = 4$, because both u^1 and u^2 solve the linear programming subproblem (7) for determining the most violated constraint. Suppose that we plant our left foot at this intersection point and move our right foot counter clockwise as far as possible so that it still touches one of the two lines (9) and (10); then we must lie on the line (9) which corresponds to the preferred cut. An equivalent procedure is to select any $y^o > 0$ (note that any such point is an interior point to the set $\{ y \geqslant 0 \}$) and to find

$$\underset{j = 2 \text{ or } 3}{\text{maximize}} \{ d y^o + u^j(b - Dy^o) \}.$$

As we shall see, a generalization of this observation will permit us to generate preferred or "strong" cuts for any cutting plane type of algorithm.

4.2.2 Formalization

Consider the minimax problem

$$\underset{y \in Y}{\text{Minimize}} \quad \underset{u \in U}{\text{Maximize}} \quad \{ f(u) + y\, g(u) \} \tag{11}$$

where Y and U are subsets of R^k and R^m, f(u) is a real valued function and g(u) is an m-dimensional vector for any u. If we let $f(u) = u\,b$, $g(u) = (d - u\,D)$ and $U = \{ u \in R^k : u\,A \leqslant c \}$, then this problem arises as formulation (7) when applying Benders decomposition. The problem also arises when dualizing mathematical programs of the form

$$\begin{aligned} &\underset{u \in U}{\text{Maximize}} && f(u) \\ &\text{subject to} && g(u) \leqslant 0 \end{aligned} \tag{12}$$

by attaching dual multipliers y with the constraints $g(u) \leqslant 0$. Formulation (11) then becomes the saddlepoint dual problem associated with this constrained optimization problem.

Relaxation methods such as Benders decomposition replace the equivalent form of problem (11)

$$\begin{aligned} &\underset{y \in Y}{\text{Minimize}} && Z \\ &\text{subject to} && Z \geqslant f(u) + y\, g(u) \quad \text{for all } u \in U \end{aligned} \tag{13}$$

by the approximation problem

$$\begin{aligned} &\underset{y \in Y}{\text{Minimize}} && Z \\ &\text{subject to} && Z \geqslant f(u) + y\, g(u) \quad \text{for all } u \in \hat{U} \end{aligned} \tag{14}$$

where $\hat{U}$ is a finite, and usually "small", subset of U. The approximation problem is an integer program when Benders method is applied to mixed integer programs and is a linear program (the dual of the master program) when Dantzig-Wolfe decomposition or generalized programming is applied to the optimization problem (11).

We will say that the cut (or constraint)

$$Z \geq f(u^1) + y\, g(u^1)$$

in problem (11) *dominates* or is *stronger* than the cut

$$Z \geq f(u) + y\, g(u)$$

in this problem, if

$$f(u^1) + y\, g(u^1) \geq f(u) + y\, g(u)$$

for all $y \in Y$ with a strict inequality for at least one point $y \in Y$. We call a cut *pareto optimal* if no cut dominates it. Since a cut is determined by the vector $u \in U$, we shall also say that u^1 dominates (is stronger) than u if the associated cut is stronger, and we say that u is pareto optimal if the corresponding cut is pareto optimal.

In the example worked out in the previous subsection, we showed how to generate a pareto optimal cut by solving an auxiliary problem in terms of any point $y^o > 0$. Note that any such point is an interior point of the set $\{ y : y \geq 0 \}$. This set, in turn, is the convex hull of the set $Y = \{ y : y \geq 0$ and integer $\}$. The following theorem shows that this observation generalizes to any problem of the form (11). Again, we consider the convex hull of Y, denoted Y^c, but now we will be

more delicate and consider the relative interior (or core) of Y^c, denoted $ri(Y^c)$, instead of its interior. The result will always be applicable since the relative interior of the convex set Y^c is always nonempty, even though the set may contain no interior points. For notation, let us call any point y^o contained in the relative interior of Y^c, a *core* point of Y.

Theorem : *Let* y^o *be a core point of* Y, *i.e.* $y^o \in ri(Y^c)$, *let* $U(\hat{y})$ *denote the set of optimal solutions to the optimization problem*

$$\underset{u\in U}{\text{Max}} \{ f(u) + \hat{y}\, g(u) \}, \tag{15}$$

and let u^o *solve the problem :*

$$\underset{u\in U(\hat{y})}{\text{Max}} \{ f(u) + y^o g(u) \}. \tag{16}$$

Then u^o *is pareto optimal.*

Proof : Suppose to the contrary that u^o is not pareto optimal; that is, there is a $\bar{u} \in U$ that dominates u^o. We first note that since

$$f(\bar{u}) + y\, g(\bar{u}) \geqslant f(u^o) + y\, g(u^o) \qquad \text{for all } y \in Y \tag{17}$$

that

$$f(\bar{u}) + z\, g(\bar{u}) \geqslant f(u^o) + z g(u^o) \qquad \text{for all } z \in Y^c. \tag{18}$$

To establish the last inequality, recall that any point $z \in Y^c$ can be expressed as a convex combination of a finite number of points in Y, i.e.

$$z = \Sigma \{ \lambda_y y \ : y \in Y \}$$

where $\lambda_y \geqslant 0$ for all $y \in Y$, at most a finite number of the λ_y are positive,

and $\Sigma \{\lambda_y : y \in Y\} = 1$. Also note from the inequality (17) with $y = \hat{y}$, that $\bar{u}$ must be an optimal solution to the optimization problem (15), that is, $\bar{u} \in U(\hat{y})$. It then follows from (16) that

$$f(\bar{u}) + y^o g(\bar{u}) = f(u^o) + y^o g(u^o) . \tag{19}$$

Since $\bar{u}$ dominates u^o,

$$f(u^o) + \bar{y} g(u^o) < f(\bar{u}) + \bar{y} g(\bar{u}) \tag{20}$$

for at least one point $\bar{y} \in Y$. Also, since $y^0 \in ri(Y^c)$ there exists (see [4-11, Theorem 6.4]) a scalar $\theta > 1$ such that

$$z \equiv \theta y^o + (1-\theta)\bar{y}$$

belongs to Y^c. Multiplying equation (19) by θ and multiplying inequality (20) by $(1-\theta)$, which is negative and reverses the inequality, and adding gives :

$$f(u^o) + z g(u^o) > f(\bar{u}) + z g(\bar{u}) .$$

But this inequality contradicts (18), showing that our supposition that u^o is not pareto optimal is untenable. This completes the proof.

When $f(u) = u b$, $g(u) = (d - u D)$ and $U = \{u \in R^k : u A \leqslant c\}$ as in Benders decomposition for mixed integer programs, problem (15) is a linear program. In this case, $U(\hat{y})$ is the set of points in U satisfying the linear equation

$$u(b - D\hat{y}) = -d\hat{y} + \hat{Z}$$

where $\hat{Z}$ is the optimal value of problem (14). Therefore to find a pareto optimal point among all the alternate optimal solutions to problem (15),

we solve problem (16) which is the linear program :

$$\begin{aligned} &\text{Maximize} && \{\, dy^o + u(b - Dy^o) \,\} \\ &\text{subject to} && u(b - D\hat{y}) = \hat{Z} - d\hat{y} \\ &\text{and} && uA \leqslant c. \end{aligned} \tag{21}$$

We should note that varying the core point y^o might conceivably generate different pareto optimal cuts. Also, when implementing a strong cut version of Benders algorithm we have the option of generating pareto optimal cuts at every iteration, or possibly, of generating pareto optimal cuts periodically. The tradeoff will depend upon the computational burden of solving problem (16) as compared to the number of iterations that it saves.

In many instances, it is easy to specify a core point y^o for implementing the pareto optimal cut algorithm. If, for example, $Y = \{ y \in R^k : y \geqslant 0$ and integer $\}$ then any point $y^o > 0$ will suffice; if $Y = \{ y \in R^k : y_j = 0$ or 1 for $j = 1,2,\ldots,k \}$ then any vector y^o with $0 < y^o_j < 1$ for $j = 1,2,\ldots,k$ suffices; and if

$$Y = \{ y \in R^k : \sum_{j=1}^{k} y_j \leqslant p,\ y \geqslant 0 \text{ and integer} \}$$

as in the inequality version of the p-median problem, then any point y^o with $y^o > 0$ and $\sum_{j=1}^{k} y^o < p$ suffices. In particular, if $p > \frac{k}{2}$, then $y^o = (\frac{1}{2}, \frac{1}{2}, \ldots, \frac{1}{2})$ is a core point.

4.3 ACCELERATING BENDERS METHOD FOR NETWORK OPTIMIZATION

Although solving the linear program (21) always generates pareto optimal cuts when Benders method is applied to mixed integer programs, it might be possible to generate strong cuts more efficiently in certain situations. In particular, when the Benders subproblem is a network optimization problem, special purpose network algorithms can be developed for solving the subproblem. The situation is similar to solving shortest path problems on a network. Even though the problem can be solved by general purpose linear programming algorithms, specialized algorithms are much more efficient.

In this section we describe special network algorithms for generating strong cuts for the p-median and network design problems. The algorithms aim to find cuts that dominate those that would ordinarily be generated by straightforward implementation of Benders algorithm. We do not, in every case, strive to find pareto optimal cuts. Rather, we discuss several algorithms which range from those that are easy to describe and implement, but that are not guaranteed to find pareto optimal cuts, to more elaborate algorithms that we feel do guarantee cuts that are pareto optimal.

4.3.1 Strong Cuts for the p-Median Problems

We begin by illustrating the notion of strong cuts, in their most rudimentary form, for the p-median problem. Recall that the p-median problem is formulated as :

$$v = \text{Minimize} \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} x_{ij}$$

$$\text{subject to :} \quad \sum_{j=1}^{n} y_j = p$$

$$\sum_{j=1}^{n} x_{ij} \geqslant 1 \qquad (i = 1,\ldots,n)$$

$$x_{ij} \leqslant y_j \qquad (i, j = 1,\ldots,n)$$

$$x_{ij} \geqslant 0 \qquad (i, j = 1,\ldots,n)$$

$$y_j = 0 \text{ or } 1,$$

where d_{ij} = the distance (or cost) from site i to site j and

$$y_j = \begin{cases} 1 \text{ if site j is selected} \\ 0, \text{ otherwise.} \end{cases}$$

We must select p sites to minimize total distribution costs from each site to the closest of the sites selected.

For any given feasible solution to this problem, we say that site j is opened if $y_j = 1$ and that site j is closed if $y_j = 0$. This terminology corresponds to viewing y_j as a decision variable indicating whether or not a depot is located at site j.

Note that if the y_j are given, the problem becomes a linear program which in this case is trivial to solve; for each index i, the optimal values for the decision variables x_{ij} are given by :

$$x_{ij'} = \begin{cases} 1 \text{ if j' is the index j with} \\ \qquad y_j = 1 \text{ giving the minimum } d_{ij} \\ 0 \text{ otherwise} \end{cases} \tag{22}$$

and the routing cost for this selection of y_j set at value one is $w = \sum_{i=1}^{n} d_{ij'}$.

The shadow prices, or dual variables, for the locations with $y_j = 0$ also are easy to compute :

$$\Pi_{ij} = \max(d_{ij'} - d_{ij}, 0).$$

Π_{ij}, which is the negative of the dual variable for the constraint $x_{ij} \leqslant y_j$, indicates the savings in routing from site i, when y_j is changed from 0 to 1. If $d_{ij} < d_{ij'}$, then routing from i to j is cheaper than the current routing from i to j' used in (22) by $d_{ij'} - d_{ij}$ units. If $d_{ij} \geqslant d_{ij'}$, then setting y_j from 0 to 1 does not improve the current routing from i to j' at all, and the savings Π_{ij} equals 0.

The total savings for all sites i from changing y_j from 0 to 1 is

$$\mu_j = \sum_{i=1}^{n} \Pi_{ij}. \tag{23}$$

Let $\mu_j = 0$ for every site j with $y_j = 1$. The usual application of Benders decomposition uses the quantities μ_j to derive a bound on the optimal routing cost v :

$$v \geqslant w - \sum_{j=1}^{n} \mu_j y_j, \tag{24}$$

where w is the routing cost in the current solution.

For reference purposes, we shall refer to the cut in (24) as a type B cut. (We define type A cuts later in this section.)

We can interpret type B cuts in the following way. $\mu_j y_j$ is the savings in routing cost for site j, considered independent of other sites. If two sites j and k are opened, we can never improve upon the current routing cost w by more than $\mu_j + \mu_k$. We might not achieve this total

because of interactions between the two sites. For example, setting both y_j and y_k from 0 to 1 may have produced savings for routing from site i. Setting both y_j and y_k to 1 won't give us the combined savings, though, since site i must be assigned to either site j or site k, but not both. In general, $\Sigma\, \mu_j\, y_j$ is the best savings that we could obtain from the current routing cost w. We might not be able to achieve this savings because of interaction effects.

We can obtain stronger "cuts" or bounds than (24) by looking at the structure of the p-median problem more closely. When deriving the Benders cut (24), we considered savings from opening a new site, i.e. increasing y_j from 0 to 1. We did not, however, consider added costs produced by closing a site, i.e. decreasing some variable y_j currently at value 1 to value 0. Since the number of sites that we select is fixed at p, whenever some new site is opened, some site selected previously must be closed and some of these added costs must be incurred. Let

$$\sigma_{ij} = \max\ (\min_{k\neq j}(d_{ik}) - d_{ij}, 0).$$

Note that $\min_{k\neq j}(d_{ik})$ is the best routing cost from site i to all sites except for site j. If this cost exceeds d_{ij}, then node i is closest to site j and closing site j must incur at least the additional routing cost $\min_{k\neq j}(d_{ik}) - d_{ij}$. If this cost does not exceed d_{ij}, then setting y_j from 1 to 0 doesn't necessarily add any new routing costs. Consequently, σ_{ij} is the cost that must be incurred from site i if site j is closed, and

$$\nu_j = \Sigma_i\, \sigma_{ij}$$

is the total cost incurred from all sites i by setting y_j from 1 to 0. If we replace $\mu_j y_j$ in the Benders cut (24) with the term $\nu_j(1 - y_j)$, we will obtain sharper lower bounds on v and hopefully accelerate the steps of Benders algorithm.

So our new cut, which we will refer to as a type C cut, can be written as :

$$v \geq w + \sum_{i \in Y_1} (1 - y_i)\, \nu_i - \sum_{i \in Y_0} u_i\, y_i \qquad (25)$$

$$v \geq (w + \sum_{i \in Y_1} \nu_i) - \sum_{i \in Y_1} \nu_i\, y_i - \sum_{i \in Y_0} u_i\, y_i \qquad (26)$$

where Y_0 is the set of sites closed in the current solution, i.e. $i \in Y_0$ implies $y_i = 0$

and Y_1 is the set of sites opened in the current solution, i.e. $i \in Y_1$ implies $y_i = 1$.

As a numerical example, consider a two-median problem on the following network :

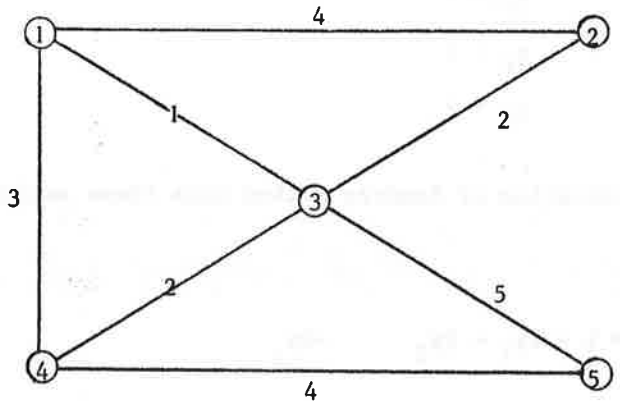


FIGURE 4.4 p-MEDIAN EXAMPLE

The number next to each arc is the distance (or cost) between the nodes incident to that arc. Using this data, we can write the distance matrix d_{ij} of shortest distances, or routing distances, between the nodes :

Distance Matrix $[d_{ij}]$					
	1	2	3	4	5
1	0	3	1	3	6
2	3	0	2	4	7
3	1	2	0	2	5
4	3	4	2	0	4
5	6	7	5	4	0

If we start with $y_3 = y_5 = 1$, the current routing cost is $w = 5$ since we route from nodes 1,2,3 and 4 to node 3 and from node 5 to node 5 to minimize routine costs. Adding node 1 will only alter this routing pattern by routing from node 1 to itself, saving us the cost $d_{13} = 1$. Thus the saving μ_1 for node 1 equals 1 unit. Similarly, we find

$$\mu_1 = 1$$
$$\mu_2 = 2$$
$$\mu_4 = 2.$$

The usual application of Benders method uses these savings to obtain the bound

$$v \geq 5 - 1y_1 - 2y_2 \qquad -2y_4 \tag{27}$$

from (24) on the optimal routing cost v.

If we reduce y_3 from 1 to 0, the second best routing *from* node 3 is to node 1 at a cost of 1 unit. Since we currently are routing node 3 to node 3 at a cost of 0, we incur an additional cost of 1 unit. Similarly, setting y_5 from 1 to 0 incurs a cost of at least 4 units. The strong Benders cut

$$v \geqslant 5 - 1y_1 - 2y_2 + 1(1 - y_3) - 2y_4 + 4(1 - y_5) \tag{28}$$

accounts for these additional costs.

Benders method will next select a new configuration of variables y_j to set to 1 to find the best lower bound on v subject to the p-median constraint

$$\begin{aligned} &y_1 + y_2 + y_3 + y_4 + y_5 = 2 \\ &y_j = 0 \text{ or } 1 \quad (j = 1, 2, \ldots, 5) \end{aligned} \tag{29}$$

The usual method will minimize v subject to (27) and (29) giving $y_2 = y_4 = 1$ and $v \geqslant 1$. The modified method will minimize v subject to (28) and (29) giving $y_4 = y_5 = 1$ and the better lower bound $v \geqslant 4$. In either case, Benders method or its modification, will repeat the previous steps starting from the new site selections.

Tables 4.1 and 4.2 summarize the remaining iterations of Benders method or its modification to solve this simple problem. In this case, the usual application of Benders method requires four iterations while its modification only requires two iterations.

TABLE 4.1 BENDERS METHOD FOR THE EXAMPLE

Iteration 1

$$v \geq 5 - y_1 - 2y_2 \qquad - 2y_4$$

$$y_1 + y_2 + y_3 + y_4 + y_5 = 2$$

set $y_2 = y_4 = 1$, $\longrightarrow 1 \leq v \leq 5 \leftarrow$ value of best known solution

lower bound generated

Iteration 2

$$v \geq 5 - y_1 - 2y_2 \qquad - 2y_4$$

added cut $\longrightarrow v \geq 9 - 4y_1 \qquad -4y_3 \qquad - 4y_5$

$$y_1 + y_2 + y_3 + y_4 + y_5 = 2$$

set $y_1 = y_3 = 1$, $\qquad 4 \leq v \leq 5$

Iteration 3

$$v \geq 5 - y_1 - 2y_2 \qquad - 2y_4$$

$$v \geq 9 - 4y_1 \qquad - 4y_3 \qquad - 4y_5$$

added cut $\longrightarrow v \geq 8 \qquad - 2y_2 \qquad - 2y_4 - 5y_5$

$$y_1 + y_2 + y_3 + y_4 + y_5 = 2$$

set $y_1 = y_5 = 1$, $\qquad 4 \leq v \leq 5$

Iteration 4

$$v \geq 5 - y_1 - 2y_2 \qquad - 2y_4$$

$$v \geq 9 - 4y_1 \qquad - 4y_3 \qquad - 4y_5$$

$$v \geq 8 \qquad - 2y_2 \qquad - 2y_4 - 5y_5$$

added cut $\longrightarrow v \geq 7 \qquad - 3y_2 - 3y_3 - 3y_4$

$$y_1 + y_2 + y_3 + y_4 + y_5 = 2$$

Set $y_3 = y_5 = 1$, $\qquad 5 \leq v \leq 5$ OPTIMAL

TABLE 4.2 MODIFIED BENDERS FOR THE EXAMPLE

Iteration 1

$$v \geqslant 5 - y_1 - 2y_2 + (1 - y_3) - 2y_4 + 4(1 - y_5)$$

or $$v \geqslant 10 - y_1 - 2y_2 - y_3 - 2y_4 - 4y_5$$

$$y_1 + y_2 + y_3 + y_4 + y_5 = 2$$

set $y_4 = y_5 = 1$, $\qquad 4 \leqslant v \leqslant 5$

Iteration 2

$$v \geqslant 10 - y_1 - 2y_2 - y_3 - 2y_4 - 4y_5$$

added cut $\longrightarrow$ $v \geqslant 15 - 5y_1 - 2y_2 - 6y_3 - 2y_4 - 4y_5$

$$y_1 + y_2 + y_3 + y_4 + y_5 = 2$$

set $y_3 = y_5 = 1$, $\qquad 5 \leqslant v \leqslant 5$ OPTIMAL

The essential idea underlying the type C cut is to compute the penalty that results from setting a variable y_i from to 1 to 0. In the network, this corresponds to finding the second closest neighbor to node i (node i is the closest neighbor to itself) and computing the added distance that *the demand at node i* must travel if site i is closed. This idea can be generalized to compute the penalty of traveling to the third closest neighbor, fourth closest neighbor and so on. Before giving an algorithm for implementing this idea, let us consider an example of this new type of cut.

For the network of Figure 4.4, recall that the type C cut generated when $y_3 = y_5 = 1$ is :

$$v \geqslant 5 - 1y_1 - 2y_2 + 1(1 - y_3) - 2y_4 + 4(1 - y_5).$$

Now consider the penalty of having the demand at node 3 diverted to the third nearest neighbor (either node 2 or node 4) instead of its first or second closest neighbor (i.e. node 3 and node 1). The penalty for setting both $y_1 = 0$ and $y_3 = 0$ becomes 2; that is, whenever both y_1 and y_3 are zero, the demand at node 3 must travel at least 2 additional units. We accomodate this observation in the type C cut displayed previously by changing the term $(1 - y_3)$ to $2(1 - y_3)$; then if both y_1 and y_3 are zero, we incur the penalty of 2 units. If $y_3 = 0$ and $y_1 = 1$, though, we do not incur any additional penalty (since the demand at node 3 does *not* travel to its third closest neighbor). To insure that the new cut agrees with the type C cut in this instance, we change the coefficient of y_1 to -2. Therefore, for both the new and old cuts, the terms $-y_1 + 1(1 - y_3)$ and $-2y_1 + 2(1 - y_3)$

agree whenever $y_1 = 1$ and $y_3 = 0$. (Note that the terms also agree when $y_1 = 0$ and $y_3 = 1$. We consider the only remaining case $y_1 = 1$, $y_3 = 0$ when discussing the general procedure later in this section).

After making similar changes to the coefficients of y_5 and y_4 by considering the third closest neighbor to node 5 (which has $y_5 = 1$ in the current solution), we obtain the following cut :

$$v \geq 5 - 2y_1 - 2y_2 + 2(1 - y_3) - 3y_4 + 5(1 - y_5)$$

or

$$v \geq 12 - 2y_1 - 2y_2 - 2y_3 - 3y_4 - 5y_5.$$

As the reader can verify, this cut dominates the type C cut over the region

$$Y = \{ y : y_1 + y_2 + \ldots + y_5 = 2,\ y_j \geq 0 \text{ and integer } \}.$$

We will refer to this new type of cut as a type A cut. Note that when forming this cut, we only apply this generalized penalty to those nodes that are opened in the current solution, but adding the penalty might produce changes in the coefficients of variables y_j for other nodes as well.

To construct the type A cuts, in general, we proceed as follows. We start with the type C cut

$$v \geq w - \sum_{i \in Y_0} \mu_i y_i + \sum_{j \in Y_1} \nu_j (1 - y_j) \qquad (30)$$

containing the maximal savings coefficient μ_i for opening any node i that is currently closed, and the penalty coefficient ν_j of diverting the demand

at node j to its second closest neighbor whenever node j is closed. Next consider neighbors of any opened node $j \in Y_1$, further away than its second closest neighbor. As above let ν_j denote the distance from node j to its second closest neighbor and define

$$N_j(\delta) = \{ \text{nodes } i : d_{ij} \leq \nu_j + \delta \} .$$

$N_j(\delta)$ contains all nodes that are no more than δ units further away from node j than its second closest neighbor. Figure 4.5 illustrates the $N_j(\delta)$

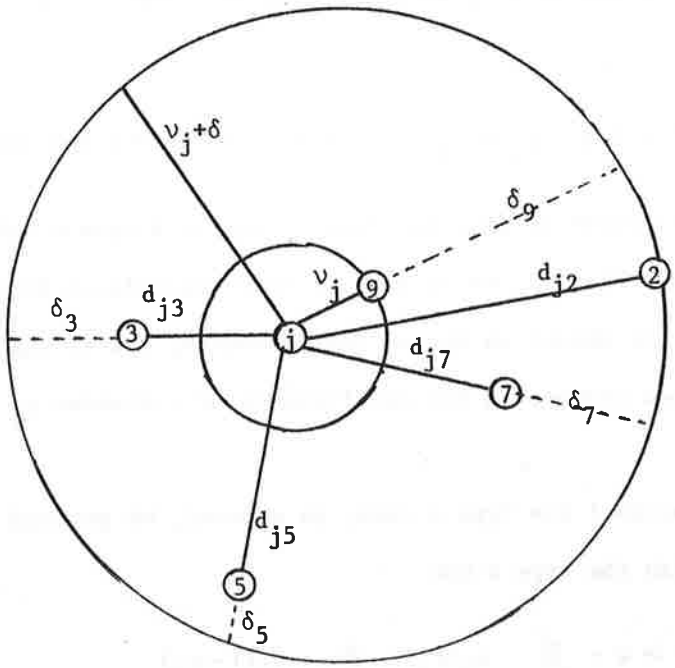


FIGURE 4.5 THE $N_j(\delta)$ NEIGHBORHOOD OF NODE (j)

neighborhood of node j. The outer ring, $\delta + \nu_j$ units away from node j, contains all nodes in the neighborhood $N_j(\delta)$. (For illustration purposes, we have pictured the neighborhood as though distances were Euclidean, an assumption that is not required.)

Now, if all of the nodes within the $N_j(\delta)$ neighborhood of node j are closed, except possibly for the node on the outer boundary (node 2 in Figure 4.5) then the demand at node j must travel at least $\nu_j + \delta$ units. If node 5 in the figure is opened, though, and all nodes closer to node j are closed, then the demand at node j must only travel at least d_{j5} units. Defining $\delta_5 = \nu_j + \delta - d_{j5}$, we may incorporate this observation into the type C cut, inequality (30), by changing the coefficient of y_5 from $-\mu_5 y_5$ to $(-\mu_5 - \delta_5)y_5$ and the coefficient of y_j from $\nu_j(1 - y_j)$ to $(\nu_j + \delta)(1 - y_j)$. Then if $y_5 = 1$ and $y_j = 0$ these two terms sum to

$$-\mu_5 - \delta_5 + \nu_j + \delta = -\mu_5 + d_{j5}$$

which is the maximum savings $-\mu_5$ from opening node 5 plus the penalty for diverting the demand at node j to node 5.

Similarly, we may associate coefficients $\delta_i = \nu_j + \delta - d_{ji}$ with the variables y_3, y_7 and y_9 for the other nodes interior to the outer ring in Figure 4.5. The expression

$$(\mu_3 + \delta_3)y_3 - (\mu_5 + \delta_5)y_5 - (\mu_7 + \delta_7)y_7 - (\mu_9 + \delta_9)y_9 + (\nu_j + \delta)(1 - y_j)$$

now replaces the corresponding terms in the type C cut. In this case, if any one of the sites 3, 5, 7, or 9 is opened, say site i, and site j is closed, then the previous expression with $y_i = 1$ reduces to :

$$-\mu_i - \delta_i + \nu_j + \delta = -\mu_i + d_{ji}$$

incorporating the penalty d_{ji} for diverting the demand from node j to node i. (We will shortly consider the situation when more than one of these sites is opened.)

We can form a new type of cut, called a type A cut, by constructing a neighborhood around each of the sites open in the current solution. The cut is :

$$v \geq w + \sum_{i \in Y_0} (-\mu_i - \delta_i) y_i + \sum_{j \in Y_i} (\nu_j + \delta)(1 - y_j)$$

where

$$\delta_i = \begin{cases} \nu_j + \delta - d_{ji}; & \text{if site } i \in Y_0 \text{ is interior to the } \delta \\ & \text{neighborhood around site } j \in Y_1, \\ & \text{i.e. } d_{ji} < \nu_j + \delta. \\ 0 & \text{if site } i \in Y_0 \text{ is not interior to the } \delta \\ & \text{neighborhood around any site } j \in Y_1, \end{cases}$$

and $\delta =$ largest integer (we assume that the data d_{ij} is integer) with the property that no site $i \in Y_1$ is contained in the interior of the δ neighborhood around more than one site $j \in Y_1$.

The restriction on δ insures that the coefficients δ_i are well defined. Note that with these definitions a site i could lie in more than one δ neighborhood, as long as it lies on the boundary (i.e., $d_{ki} = \nu_k + \delta$) of all but at most one of these neighborhoods.

All three of the cuts, type A, type B and type C, can be classified according to the scheme given in the previous section. Definition (30) implies that type C cuts dominate type B cuts as long as at least one $\nu_j \neq 0$. The following theorem summarizes the relationship between type A and type C cuts.

Theorem : *For a given iteration of Benders decomposition for the p-median problem, a type* A *cut will either dominate a type* C *cut or be equivalent to it.*

Proof : Let $y = \bar{y}$ be any values for the configuration variables satisfying the p-median constraint

$$y_1 + y_2 + \ldots + y_n = p$$
$$y_j = 0 \text{ or } 1 \text{ (all } j)$$

and consider the type A and C cuts evaluated at $y = \bar{y}$:

$$v \geq w + \sum_{i \in Y_0} (-\mu_i - \delta_i)\bar{y}_i + \sum_{j \in Y_1} (\nu_j + \delta)(1 - \bar{y}_j)$$
$$v \geq w + \sum_{i \in Y_0} -\mu_i \bar{y}_i + \sum_{j \in Y_1} \nu_j (1 - \bar{y}_j) .$$

The right-hand side of the first of these cuts, the type A cut, equals the right-hand side of the second, or type C, cut plus the term

$$\sum_{j \in Y_1} \delta (1 - \bar{y}_j) - \sum_{i \in Y_0} \delta_i \bar{y}_i . \tag{31}$$

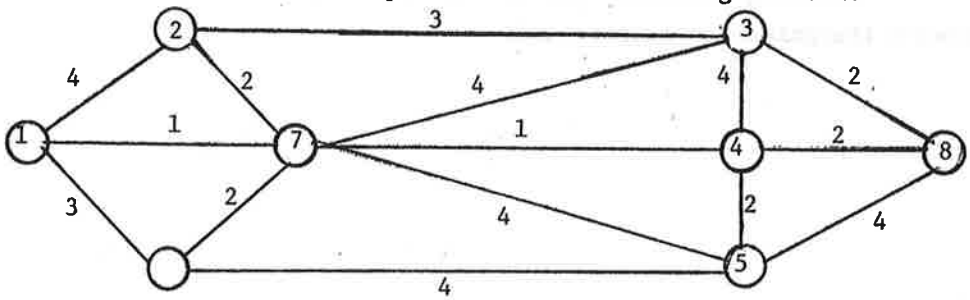
By the p-median constraint, if q of the sites $i \in Y_0$ are opened (i.e. q of the $\bar{y}_i = 1$), then q of the sites $j \in Y_1$, must be closed (i.e. q of the $\bar{y}_j = 0$). But by definition, either $\delta_i = 0$ or $\delta_i = \nu_k + \delta - d_{ki}$ for some

site k with $k \in Y_1$. In either case $\delta_i \leq \delta$ because v_k, equalling $\min_{j \neq k} d_{kj}$, is no larger than d_{ki}.

Consequently, the term (31) is always nonnegative and right-hand side of the type A cut is always greater than or equal to the right-hand side of the type C cut for every $y = \bar{y}$ satisfying the p-median constraint. This completes the proof.

Note that this proof is valid even when two sites are opened within the δ neighborhood of some node, resolving an issue raised previously. Also, in view of the previous theorem, we see that the type A cuts generally dominate the type B cuts as well.

To demonstrate the use of type A cuts, let us consider the following example which is a two-median problem on the following network:



The number next to each arc is the distance (or cost) between the nodes incident to that arc. Using this data, we can write the distance matrix d_{ij} of shortest distances between the nodes:

Distance Matrix $[d_{ij}]$

Node i \ Node j	1	2	3	4	5	6	7	8
1	0	3	5	2	4	3	1	4
2	3	0	3	3	5	4	2	5
3	5	3	0	4	6	6	4	2
4	2	3	4	0	2	3	1	2
5	4	5	6	2	0	4	3	4
6	3	4	6	3	4	0	2	5
7	1	2	4	1	3	2	0	3
8	4	5	2	2	4	5	3	0

Tables 4.3 and 4.4 summarize the interations of Benders method when applied to the above problem. Table III solves the problem using the type C cut. Table IV solves the problem using the type A cut.

Notice that for this example the type A generating technique was able to reduce the number of Benders iterations required to solve the problem from 3 iterations to 1 iteration.

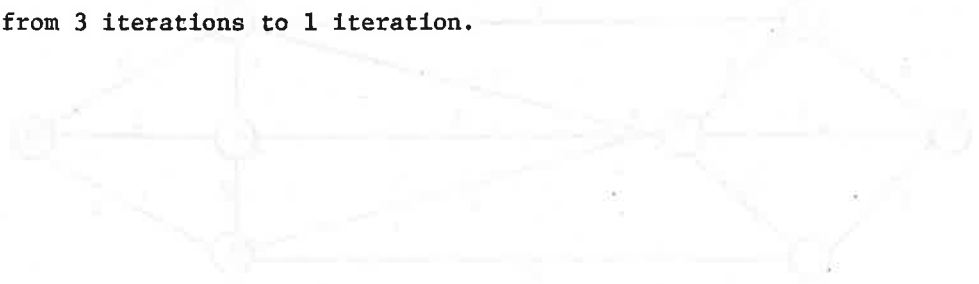


TABLE 4.3

BENDERS METHOD FOR THE EXAMPLE

USING THE TYPE C CUT

Start with $y_3 = y_7 = 1$

Iteration 1

$$V \geq 14 - 1y_1 - 2y_2 - 2y_3 - 2y_4 - 3y_5 - 2y_6 - 1y_7 - 2y_8$$

$$y_1 + y_2 + y_3 + y_4 + y_5 + y_6 + y_7 + y_8 = 2$$

set $y_2 = y_5 = 1$ $\qquad 9 \leq V \leq 11$

Iteration 2

$$V \geq 14 - 1y_1 - 2y_2 - 2y_3 - 2y_4 - 3y_5 - 2y_6 - 1y_7 - 2y_8$$

$$V \geq 22 - 4y_1 - 2y_2 - 5y_3 - 7y_4 - 2y_5 - 4y_6 - 8y_7 - 5y_8$$

$$y_1 + y_2 + y_3 + y_4 + y_5 + y_6 + y_7 + y_8 = 2$$

set $y_3 = y_4 = 1$ $\qquad 10 \leq V \leq 11$

Iteration 3

$$V \quad 14 - 1y_1 - 2y_2 - 2y_3 - 2y_4 - 3y_5 - 2y_6 - 1y_7 - 2y_8$$

$$V \quad 22 - 4y_1 - 2y_2 - 5y_3 - 7y_4 - 2y_5 - 4y_6 - 8y_7 - 5y_8$$

$$V \quad 16 - 2y_1 - 3y_2 - 2y_3 - 1y_4 - 2y_5 - 3y_6 - 4y_7 - 2y_8$$

$$y_1 + y_2 + y_3 + y_4 + y_5 + y_6 + y_7 + y_8 = 2$$

set $y_3 = y_7 = 1$ $\qquad 11 \leq V \leq 11$

OPTIMAL

TABLE 4.4

BENDERS METHOD FOR THE EXAMPLE

USING THE TYPE A CUT

- NODE ASSIGNMENT "COST" = 11

Start with $y_3 = y_7 = 1$

Iteration 1

$$V \geq 14 - 1y_1 + (-y_1) - 2y_2 - 2y_3 + 3(1-y_3) - 2y_4 + (-y_4)$$
$$-3y_5 - 2y_6 + 2(1-y_7) - 2y_8 + (-y_8)$$

or

$$V \geq 19 - 2y_1 - 2y_2 - 5y_3 - 3y_4 - 3y_5 - 2y_6 - 2y_7 - 3y_8$$

$$y_1 + y_2 + y_3 + y_4 + y_5 + y_6 + y_7 + y_8 = 2$$

set $y_3 = y_7 = 1$ $\qquad 11 \leq V \leq 11$

OPTIMAL

Many other kinds of cuts are possible for the p-median problem. For the sake of computational comparisons, we describe one additional type of cut, called a type D cut. This cut, which is also a variant of the type C cut, is defined as :

$$v \geq w + \sum_{i \in Y_0} (-\mu_i - \delta_i) y_i + \sum_{j \in Y_1} (\nu_j + \delta_j)(1 - y_j).$$

In this case, w, the μ_i and the ν_j are as given in the type C cut. For any site $j \in Y_1$ currently open, δ_j is defined so that the neighborhood $N_j(\delta_j)$ is as large as possible without containing another open site interior to its boundary, i.e.,

$$\delta_j = \max \{ \delta : \nu_j + \delta \leq d_{jk} \quad \text{for all } k \in Y_1 \} .$$

The coefficients δ_i for closed sites $i \in Y_0$ are defined as in the type A cut, except that now a closed site might lie interior to more than one neighborhood $N_j(\delta_j)$. When this occurs, δ_i is defined by adding the corresponding terms for each neighborhood, i.e.,

$$\delta_i = \Sigma \{ (\nu_k + \delta_k - d_{ki}) : k \in Y_1 \text{ and } d_{ki} < \nu_k + \delta_k \} .$$

If site i is not contained in the interior of any neighborhood around one of the open sites, i.e., $d_{ki} \geq \nu_k + \delta_k$ for all $k \in Y_1$, then δ_i is set to value zero.

The type D cuts neither dominate or are dominated by any of the other types of cuts introduced earlier. We conjecture that the type D cut is pareto-optimal, a conjecture that we are currently investigating. We discuss this cut further in section 4.4 on computational tests.

The cuts A and C introduced in this section for the p-median problem are stronger than the type B cut produced by straightforward implementation of Benders algorithm. The cuts have been derived by exploiting the network structure of the problem, rather than by using the linear programming approach of the last section for generating pareto-optimal cuts. We are currently attempting to reconcile these two approaches. We believe that the type D cut is one cut than can be generated from the linear program (21) by a particular choice for the core point y^0; by varying the core point, we hope to be able to generate a number of other pareto-optimal cuts easily using the network structure of the p-median problem.

4.3.2 Strong Cuts for Uncapacitated Network Design

Next we discuss the generation of strong cuts for the network design problem. Recall from section 2 this problem is formulated as :

$$\text{Minimize :} \quad \sum_{(i,j)\in A} \sum_{k} \sum_{l} c_{ij} x_{ij}^{kl} + \sum_{(i,j)\in A} b_{ij} y_{ij} \tag{32}$$

$$\text{subject to :} \quad \sum_{j} x_{ij}^{kl} - \sum_{j} x_{ji}^{kl} = \begin{cases} 0 & i \neq k,\ i \neq l \\ R_{kl} & i = k \\ -R_{kl} & i = l \end{cases} \quad \text{for all } k,l \tag{33}$$

$$x_{ij}^{kl} \leqslant R_{kl} y_{ij} \qquad \text{for } (i,j)\in A, \text{ for all } k,l \tag{34}$$

$$x_{ij}^{kl} \geqslant 0 \qquad \text{for } (i,j)\in A, \text{ for all } k,l \tag{35}$$

$$y_{ij} = 0 \text{ or } 1 \qquad \text{for } (i,j)\in A \tag{36}$$

where

i,j,k,l	are nodes indices.
x_{ij}^{kl}	is a variable denoting the amount of flow routed over the arc (i,j) whose origin is k and destination is l.
y_{ij}	is a 0-1 variable that will be 1 if the arc between i and j is added to the network and 0 otherwise.
A	is the set of candidate arcs for the networks.
R_{ij}	is the amount of flow that must be routed between nodes i and j.

Careful inspection of this problem indicates that it can be decomposed into a series of problems of the following form :

$$
\begin{aligned}
& \min \sum_i \sum_j c_{ij}\, x_{ij} \\
\Pi_l \quad & \sum_i x_{il} - \sum_j x_{lj} = -R_{k\ell} \\
\Pi_N \quad & \sum_i x_{iN} - \sum_j x_{Nj} = R_{k\ell} \\
\Pi_h \quad & \sum_i x_{ih} - \sum_j x_{hj} = 0 \qquad 2 \leqslant h \leqslant (N-1) \\
\gamma_{ij} \quad & x_{ij} \leqslant R_{k\ell}\, y_{ij} \\
& x_{ij} \geqslant 0,\ y_{ij} \in \{0,1\} \qquad \forall\, i,j
\end{aligned}
\tag{37}
$$

where $1 \leqslant k \leqslant N$ and $1 \leqslant \ell \leqslant N$.

For any assignment of the y_{ij} variables it is easy to see that (37) becomes a shortest path problem. Let the configuration variables y be fixed at values $y = \bar{y}$. The dual of (37) is :

$$\max R_{k\ell}[(\Pi_k - \Pi_\ell) - \sum_i \sum_j \gamma_{ij} \bar{y}_{ij}]$$

$$(\Pi_j - \Pi_i) - \gamma_{ij} \leqslant c_{ij} \qquad \forall i,j \qquad (38)$$

$$\gamma_{ij} \geqslant 0, \ \Pi_h \text{ unrestricted}$$

Suppose that $\{\Pi_i^{k\ell}\}$ and $\{\gamma_{ij}^{k\ell}\}$ solve (38); then in the context of Benders decomposition,

$$v \geqslant \sum_k \sum_\ell R_{k\ell} \ [(\Pi_N^{k\ell} - \Pi_1^{k\ell}) - \sum_i \sum_j \gamma_{ij}^{k\ell} \ y_{ij}]$$

defines a cut. In order to simplify the following discussion let us consider only one of the problems (37) and drop the indices k and ℓ. We assume that $R_{k\ell} = 1$ for the problem being considered. Further assume that $\{\Pi_i^*\}$ and $\{\gamma_{ij}^*\}$ solve (38). So the Benders cut becomes :

$$v \geqslant (\Pi_N^* - \Pi_1^*) - \sum_i \sum_j \gamma_{ij}^* \ y_{ij}.$$

Now problem (37) generally has a degenerate optimal basis. This fact has a well-known network interpretation; in a network with N nodes, the shortest path between any two nodes usually consists of fewer than the (N - 1) arcs in a basis. Basic arcs not along the shortest path are degenerate, at value 0 in (37).

Since (37) is usually degenerate, its dual (38) will generally have multiple optimal solutions.

With $y = \bar{y}$, regarding (37) as a shortest path problem gives us one possible interpretation of the dual variables. Π_i is the shortest distance between nodes 1 and i on the network is described by $y = \bar{y}$. Since

$$\gamma_{ij} = \begin{cases} 0 & \text{if } \Pi_j - \Pi_i \leq c_{ij} \\ \Pi_j - \Pi_i - c_{ij} & \text{if } \Pi_j - \Pi_i > c_{ij}, \end{cases}$$

γ_{ij} can be interpreted as the reduction in the shortest path distance between nodes 1 and N if $y_{ij} = 1$ (i.e. if are (i,j) is added to the network defined by $\bar{y}$).

Let SOPT = $\{\Pi_i^1, \gamma_{ij}^1\}$ be a set of optimal dual values defined by the above procedure. Other values of the optimal dual variables are usually possible.

We next illustrate this fact, giving a procedure that yields another set of optimal dual values. This set has the property that the Benders cut defined by it is never weaker that the cut defined by the set SOPT; That is, if $\{\Pi_i^2, \gamma_{ij}^2\}$ denotes the optimal dual values produced by this new procedure. Then,

$$\Pi_N^2 - \Pi_1^2 - \sum_i \sum_j \gamma_{ij}^2 y_{ij} \geq \Pi_N^1 - \Pi_1^1 - \sum_i \sum_j \gamma_{ij}^1 y_{ij}$$

for all possible values of y.

Usually this new set of dual values produces a cut which dominates the cut defined by the set SOPT.

Define:

D^* = optimal value of (37) when $y = \bar{y}$,

Π_i^1 = minimum distance from node 1 to node i on the network defined by $y = \bar{y}$,

and

D_i = minimum distance from node 1 to node N on the network defined by all the arcs in A [i.e. $y_{ij} = 1$ for all $(i,j) \in A$].

Let

$$\Delta_i = D^* - D_i$$
$$\Pi_i^2 = 0, \quad \Pi_N^2 = D^*$$
$$\Pi_i^2 = \min (\Pi_i^1, \Delta_i), \qquad 2 \leq i \leq N-1,$$

and

$$\gamma_{ij}^2 = \begin{cases} 0 & \text{if } \Pi_j^2 - \Pi_i^2 \leq c_{ij} \\ \Pi_j^2 - \Pi_i^2 - c_{ij} & \text{if } \Pi_j^2 - \Pi_i^2 > c_{ij} \end{cases}$$

Theorem (a) *The set of dual values* $\{\Pi_i^2, \gamma_{ij}^2\}$ *is an optimal solution to* (38) *when* $y = \bar{y}_1$.

(b) $\gamma_{ij}^2 \leq \gamma_{ij}^1$.

Proof for part (b). By cases.

Case 1 Suppose that $\Pi_i^2 = \Pi_i^1$.

Then $\Pi_j^2 - \Pi_i^2 \leq \Pi_j - \Pi_i$,

which implies that $\gamma_{ij}^2 \leq \gamma_{ij}^1$.

Case 2 Suppose that $\Pi_i^2 = \Delta_i$.

By the triangle inequality we have,

$$D_i \leq D_j + c_{ij}$$

or $$D_i - c_{ij} \leq D_j.$$

Consequently,

$$\Delta_j = D^* - D_j \leq D^* - (D_i - D_{ij}) = D^* - D_i + c_{ij}$$
$$\Delta_j \leq \Delta_i + c_{ij}$$

and

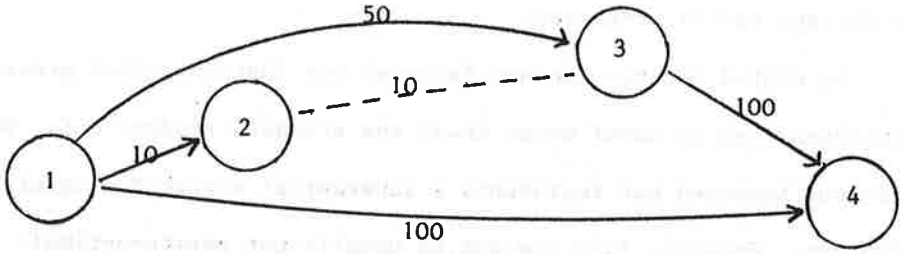
$$\begin{aligned}\Pi_j^2 - \Pi_i^2 &\leq \Delta_j - \Pi_i^2 = \Delta_j - \Delta_i \\ &\leq \Delta_i + c_{ij} - \Delta_i \\ &\leq c_{ij}\end{aligned}$$

which implies that $\gamma_{ij}^2 = 0$ and that $\gamma_{ij}^2 \leq \gamma_{ij}^1$,

For part (a).

The definition of γ_{ij}^2 guarantees that the set $\{\Pi_i^2, \gamma_{ij}^2\}$ is a feasible solution of (38). By definition $\Pi_1^2 = 0$ and $\Pi_N^2 = D^*$ now $\bar{y}_{ij} = 1$ implies that $\gamma_{ij}^2 = 0$ (See the interpretation of γ_{ij} given above.) Therefore the value of (38) for the set $\{\Pi_i^2, \gamma_{ij}^2\}$ is D^*. So the solution is optimal as well as feasible.

Example (N = 4):



Solid lines for arc (i,j) indicates $y_{ij} = 1$

Dotted " " " " " " $y_{ij} = 0$

$\Pi_1^1 = 0$ $\gamma_{12}^1 = 0$

$\Pi_2^1 = 10$ $\gamma_{13}^1 = 0$

$\Pi_3^1 = 50$ $\gamma_{14}^1 = 0$

$\Pi_4^1 = 100$ $\gamma_{23}^1 = 30$ $D^* = 100.$

$\gamma_{34}^1 = 0$

In addition,

$$\Delta_2 = -10 \qquad \gamma^2_{12} = 0$$
$$\Delta_3 = 0 \qquad \gamma^2_{13} = 0$$
$$\Pi^2_1 = 0 \qquad \gamma^2_{14} = 0$$
$$\Pi^2 = \min(-10,10) = -10 \qquad \gamma^2_{23} = 0$$
$$\Pi^2_3 = \min(0,50) = 0 \qquad \gamma^2_{34} = 0 .$$
$$\Pi^2_4 = 100$$

The cut defined by set 1 is

$$v \geq 100 - 30\, y_{23} .$$

The cut defined by the new procedure is

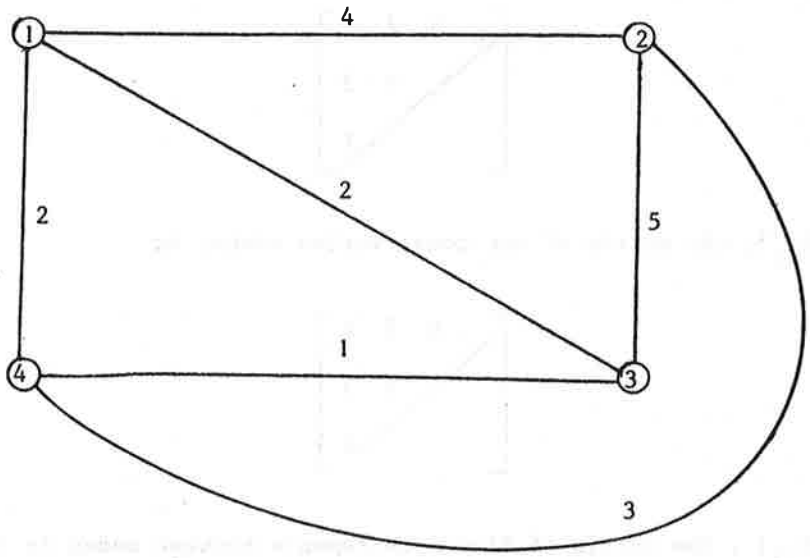
$$v \geq 100 .$$

So the new cut is preferred.

As stated before, the new (strong) cut just described generally dominates (and is never worse than) the standard Benders cut. We believe that our improved cut represents a substantial strenghtening of the standard cut. However, this new cut is usually not pareto-optimal. We believe that any pareto-optimal cut which dominates the new cut will produce only marginal improvements. In fact, one way to improve the new cut is to utilize the main idea of the strong p-median cuts. That is, calculate the increase of the flow routing cost which results from closing down an arc. Since this type of improvement for the new cut is not considered significant at this point in our investigation, we will not spell out the details here.

Now we present a small numerical example of the network design problem which further illustrates the strong cut methodology .

Consider the following network :



The arcs in the above networks are undirected. So once an arc is constructed, flow is allowed in either direction.

In order to eliminate infeasible subproblems, we will solve the master problems for this example with the following constraints added :

$$y_{12} + y_{13} + y_{14} \geq 1$$
$$y_{12} + y_{23} + y_{24} \geq 1$$
$$y_{13} + y_{23} + y_{34} \geq 1$$
$$y_{14} + y_{24} + y_{34} \geq 1$$

We let Y denote the set of 0-1 y_{ij} satisfying the above constraints.

First, we solve the network design problem using the standard Benders cuts.

$[c_{ij}]$, the matrix of arc routing costs, is

$$\begin{bmatrix} & 4 & 2 & 2 \\ & & 5 & 3 \\ & & & 1 \\ & & & \end{bmatrix}$$

$[b_{ij}]$, the matrix of arc construction costs, is

$$\begin{bmatrix} & 4 & 2 & 2 \\ & & 5 & 3 \\ & & & 1 \\ & & & \end{bmatrix}$$

and $[R_{ij}]$, the matrix of flow requirements between nodes is :

$$\begin{bmatrix} - & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & - & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & - & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & - \end{bmatrix}.$$

We start with the initial solution

$$y_{14} = y_{24} = y_{34} = 1.$$

ITERATION 1

$$v \geqslant 18 - 5y_{12} - 4y_{13} + 4y_{12} + 2y_{13} + 2y_{14} + 5y_{23} + 3y_{24} + 1y_{34}$$

or

$$v \geqslant 18 - 1y_{12} - 2y_{13} + 2y_{14} + 5y_{23} + 3y_{24} + 1y_{34}$$

$$y \in Y$$

$$\hat{y}_{12} = \hat{y}_{13} = \hat{y}_{34} = 1 \qquad 12 \leqslant v \leqslant 24.$$

ITERATION 2

$$v \geqslant 18 - 1y_{12} - 2y_{13} + 2y_{14} + 5y_{23} + 3y_{24} + 1y_{34}$$

$$v \geqslant 23 + 4y_{12} + 2y_{13} - 3y_{14} + 2y_{23} - 7y_{24} + 1_{34}$$

$$y \in Y$$

$$\hat{y}_{12} = \hat{y}_{13} = \hat{y}_{14} = \hat{y}_{24} = 1 \qquad 20 \leqslant v \leqslant 24.$$

ITERATION 3

$$v \geqslant 18 - 1y_{12} - 2y_{13} + 2y_{14} + 5y_{23} + 3y_{24} + 1y_{34}$$

$$v \geqslant 23 + 4y_{12} + 2y_{13} - 3y_{14} + 2y_{23} - 7y_{24} + 1y_{34}$$

$$v \geqslant 21 + 4y_{12} + 2y_{13} + 2y_{14} + 2y_{23} + 3y_{24} - 6y_{34}$$

$$y \in Y$$

$$\hat{y}_{13} = \hat{y}_{24} = \hat{y}_{34} = 1 \qquad 20 \leqslant v \leqslant 24.$$

ITERATION 4

$$v \geqslant 18 - 1y_{12} - 2y_{13} - 2y_{14} + 5y_{23} + 3y_{24} + 1y_{34}$$

$$v \geqslant 23 + 4y_{12} + 2y_{13} - 3y_{14} + 2y_{23} - 7y_{24} + 1y_{34}$$

$$v \geqslant 21 + 4y_{12} + 2y_{13} + 2y_{14} + 2y_{23} + 3y_{24} - 6y_{34}$$

$$v \geqslant 19 - 6y_{12} + 2y_{13} - 3y_{14} + 5y_{23} + 3y_{24} + y_{34}$$

$$y \in Y$$

$$\hat{y}_{12} = \hat{y}_{24} = \hat{y}_{34} = 1 \qquad 22 \leqslant v \leqslant 24.$$

ITERATION 5

$$v \geq 18 - 1y_{12} - 2y_{13} + 2y_{14} + 5y_{23} + 3y_{24} + 1y_{34}$$

$$v \geq 23 + 4y_{12} + 2y_{13} - 3y_{14} + 2y_{23} - 7y_{24} + 1y_{34}$$

$$v \geq 21 + 4y_{12} + 2y_{13} + 2y_{14} + 2y_{23} + 3y_{24} - 6y_{34}$$

$$v \geq 19 - 6y_{12} + 2y_{13} - 3y_{14} + 5y_{23} + 3y_{24} + 1y_{34}$$

$$v \geq 27 + 4y_{12} - 22y_{13} - 18y_{14} + 5y_{23} + 3y_{24} + 1y_{34}$$

$$y \in Y$$

$$\hat{y}_{14} = \hat{y}_{24} = \hat{y}_{34} = 1 \qquad 24 \leq v \leq 24.$$

Now we solve the same network design problem using strong cuts (the cuts are generated using the algorithm described previously.)

Again we start with the initial solution

$$\hat{y}_{14} = \hat{y}_{24} = \hat{y}_{34} = 1.$$

ITERATION 1

$$v \geq 18 - 1y_{12} - 1y_{13} + 4y_{12} + 2y_{13} + 2y_{14} + 5y_{23} + 3y_{24} + 1y_{34}$$

or

$$v \geq 18 + 3y_{12} + 1y_{13} + 2y_{14} + 5y_{23} + 3y_{24} + 1y_{34}$$

$$y \in Y$$

$$\hat{y}_{12} = \hat{y}_{14} = \hat{y}_{34} = 1 \qquad 22 \leq v \leq 24.$$

ITERATION 2

$$v \geq 18 + 3y_{12} + 1y_{13} + 2y_{14} + 5y_{23} + 3y_{24} + 1y_{34}$$

$$v \geq 23 + 4y_{12} + \quad + 2y_{14} + 3y_{23} - 3y_{24} + 1y_{34}$$

$$y \in Y$$

$$\hat{y}_{14} = \hat{y}_{24} = \hat{y}_{34} = 1 \qquad 24 \leq v \leq 24 .$$

So the use of the strong cuts has reduced the number of Benders iterations required for this problem from 5 to 2.

4.4 COMPUTATIONAL EXPERIENCE

In this section we present preliminary computational results for Benders decomposition with the strong cut methodology; all computational tests involved the p-median problem. Four different types of cuts, types A, B, C and D (as described in the previous section), were used in order to compare their respective convergence properties.

To generate initial feasible integer solutions for the first iteration of Benders procedure, we applied a heuristic procedure described by Cornuejols, Fisher and Nemhauser [4-1]. The Benders continuous subproblems were solved with the procedures described in the previous section. The master problems for these tests were solved via an exhaustive enumeration program. Naturally, the computation time for solving the master problems increased exponentially with the size of the problem when using this approach. This limited the range of p-median problems that could be tested at present.

All of the above procedures were implemented in FORTRAN on the PRIME computer system at MIT's Sloan School.

The first test problem was a 10 node network taken from [4-3]. Figure 4.6 shows the computation results for locating 3 medians on the 10 node network. The lower bound was generated by the solution of the current master problem. The upper bound was supplied by the best feasible solution generated. Since the initial heuristic solution generated for this problem was an optimal solution, the upper bound was the optimal value to the problem.

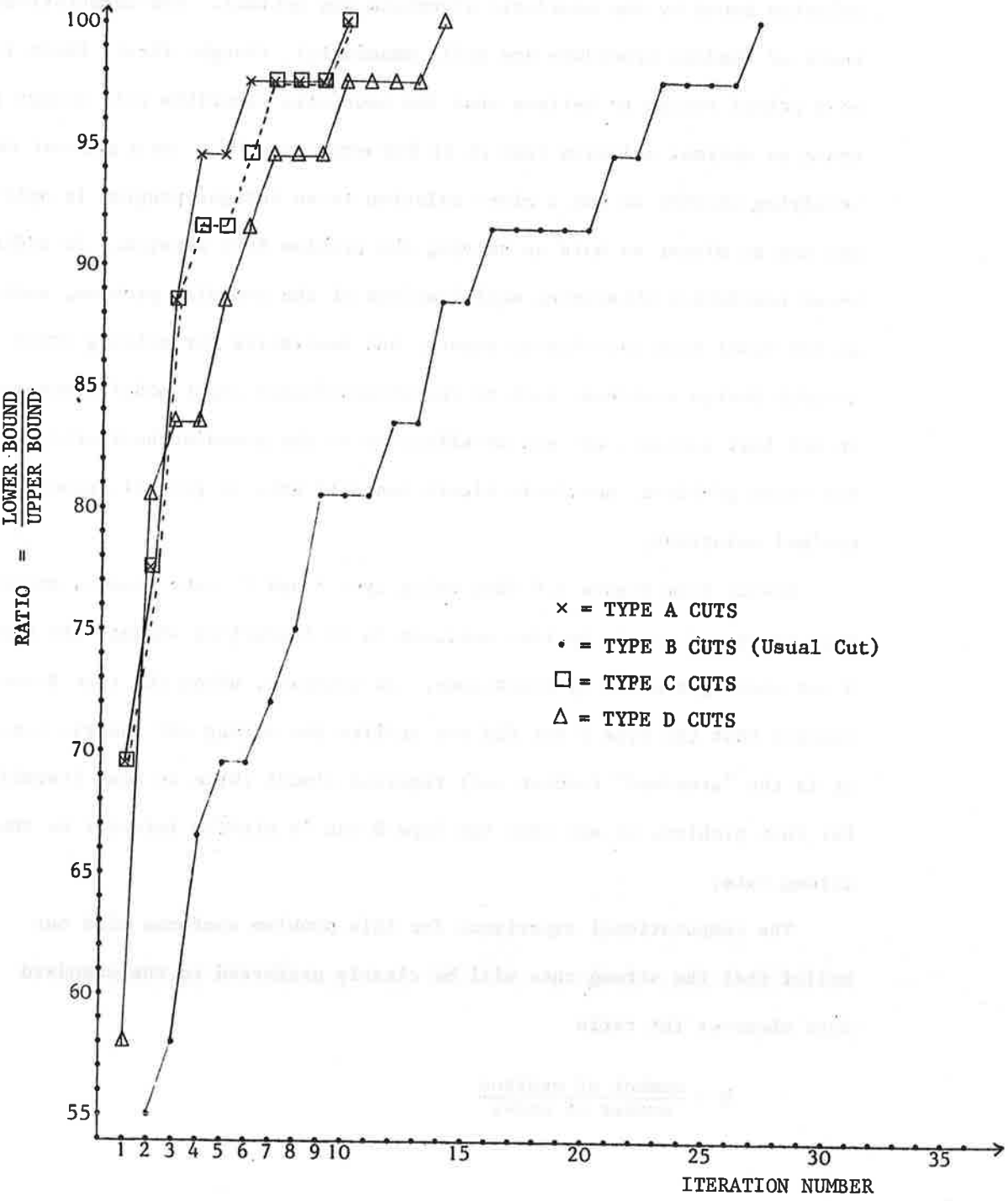


FIGURE 4.6 3-MEDIAN, 10-NODE TEST PROBLEM

In fact, in all but one of the test problems considered, the initial solution found by the heuristic algorithm was optimal. The computational tests of Benders procedure are still meaningful, though; first, there is no a priori reason to believe that the heuristic algorithm will always generate an optimal solution (see [4-1] for error bounds). As a general rule, verifying whether or not a given solution to an integer program is optimal can be almost as hard as solving the problem form scratch. In addition, known heuristics for solving modifications of the p-median problem, such as the model with capacitated depots, and heuristics for solving other network design problems, such as the uncapacitated depot model discussed in the last section, are not as effective as the p-median heuristic. For these problems, heuristic algorithms will not, in general, generate optimal solutions.

Notice from Figure 4.6 that using type A and C cuts Benders procedure converged to the optimal solution in 10 iterations whereas the type D cut converged after 14 iterations. In contrast, using the type B cut (recall that the type B cut did not utilize the strong cut theory, i.e. it is the "standard" Benders cut) required almost twice as many iterations. For this problem, we see that the type B cut is clearly inferior to the strong cuts.

The computational experience for this problem conforms with our belief that the strong cuts will be clearly preferred to the standard cuts whenever the ratio

$$R = \frac{\text{number of medians}}{\text{number of nodes}}$$

is relatively large, say greater that $\frac{1}{4}$. In this case, the ratio R equals $\frac{3}{10}$.

From Figure 4.6, we can also see that all four cuts exhibit, to some degree, a "tailing" phenomenon. That is, after a sharp initial rise in the lower bound, the convergence of the lower bound slowed considerably for some number of iterations. "Tailing", like this, occurred in most of our computational tests with p-median problems. Since this phenomenon reoccurs in many mathematical programming decomposition procedures, its appearance here is not surprising.

Figure 4.7 contains results for a 6-median location problem on the same 10 node network. All four types of cuts were tested. The results emphatically show the superiority of the strong cuts over the standard cut. The type A and type C cuts (which for this problem are identical) converged very rapidly. It is hoped that all p-median problems with such a high median to node ratio will converge this rapidly with the strong cuts.

Figure 4.8 displays results for a larger problem. The test network was a 33 node network taken from [4-6]. The problem concerns the location of 2 medians. As can be seen from the graph, there is not much difference among the four cuts. Cut B performs a little worse than the strong cuts, but the difference is negligible. Since the ratio R of medians to nodes is small, the above performance agrees with our intuition about the performance of the strong cuts. Notice that for this problem any of the four cuts performs quite well.

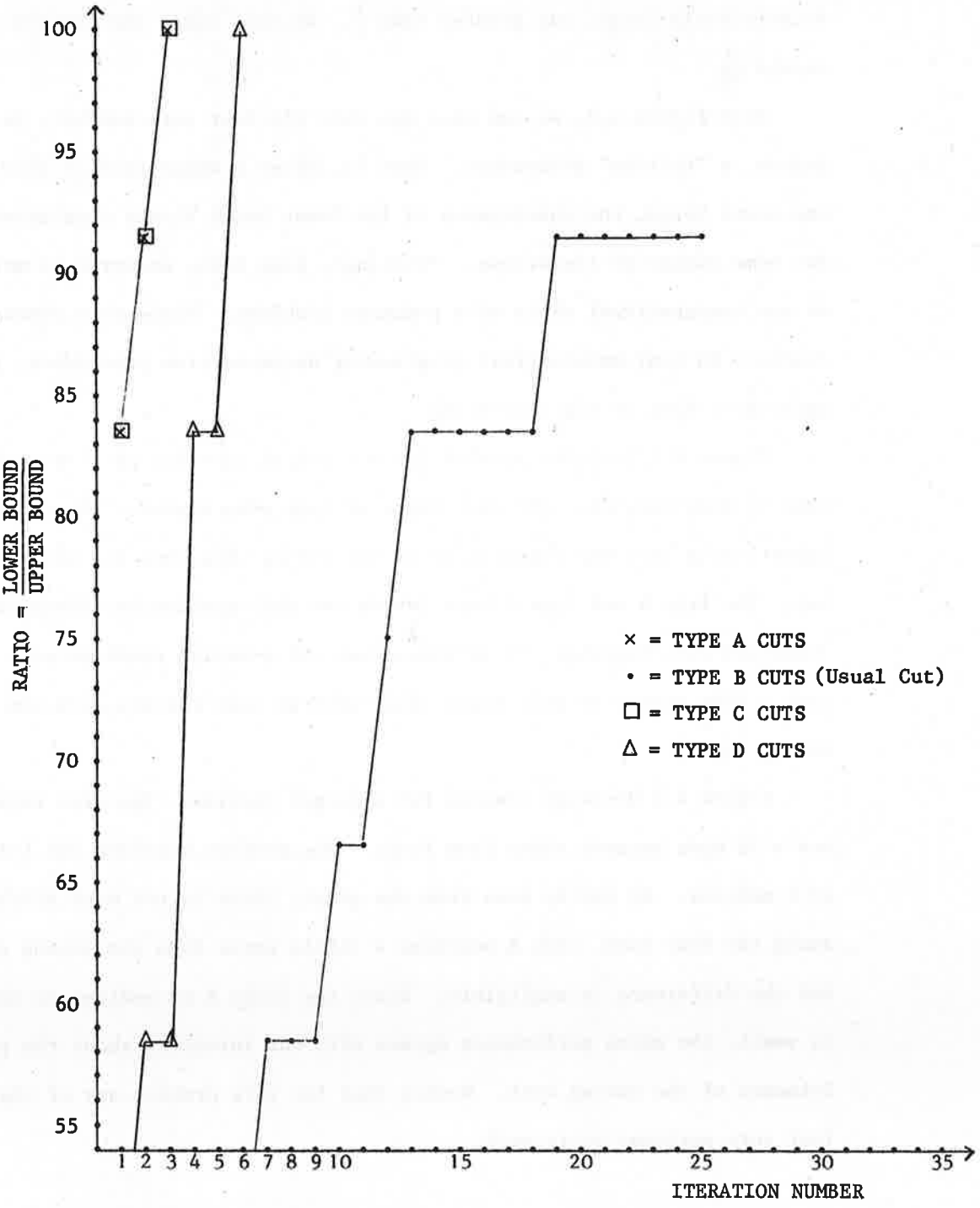


FIGURE 4.7 6-MEDIAN, 10-NODE TEST PROBLEM

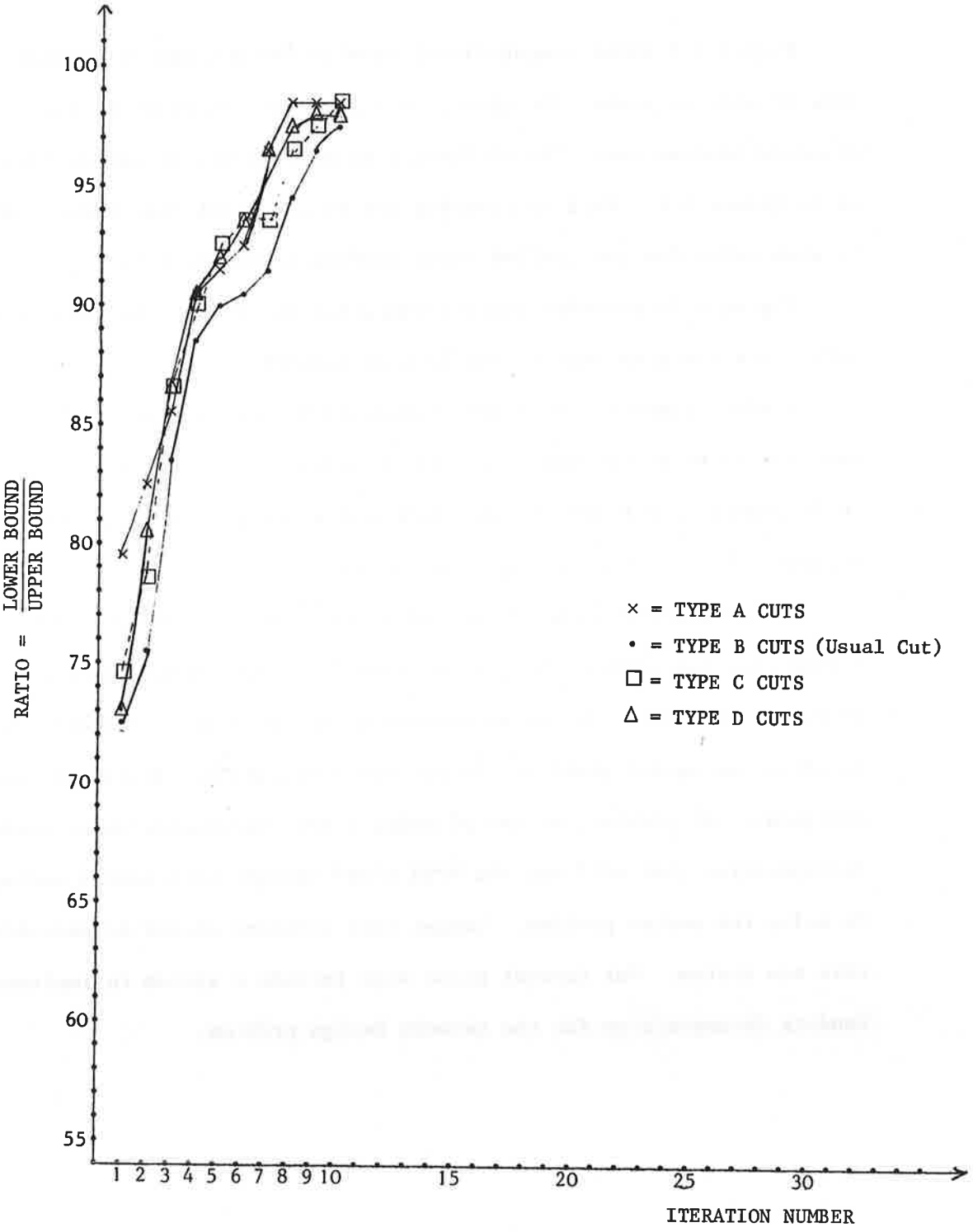


FIGURE 4.8 2-MEDIAN, 33-NODE TEST PROBLEM

Figure 4.9 shows computational results for a 4-median problem on the same 33 node network. The strong cuts performed somewhat better than the standard Benders cut. The difference in performance is not as dramatic as in Figure 4.7. This is probably due to the relatively small median to node ratio for the problem corresponding to Figure 4.9.

Figure 4.10 contains limited computational results for type A cuts only, in a 5-median test of the 33 node network.

Further computer tests were suspended due to the excessive time required to solve the master problem by exhaustive enumeration (about 15 to 20 minutes per problem). The limited results do, however, show a convergence rate similar to that seen in Figure 4.9.

In our limited series of computational tests we have seen that the strong cuts can perform much better than the standard Benders cuts. Due to time limitations, it was necessarry to use a rather crude procedure to solve the master problem. So further computational tests were not practical. At present, we are planning a new implementation of Benders decomposition that will use the MPSX mixed integer programming package to solve the master problem. Larger test problems should be solvable on this new system. Our current plans also include a system to implement Benders decomposition for the network design problem.

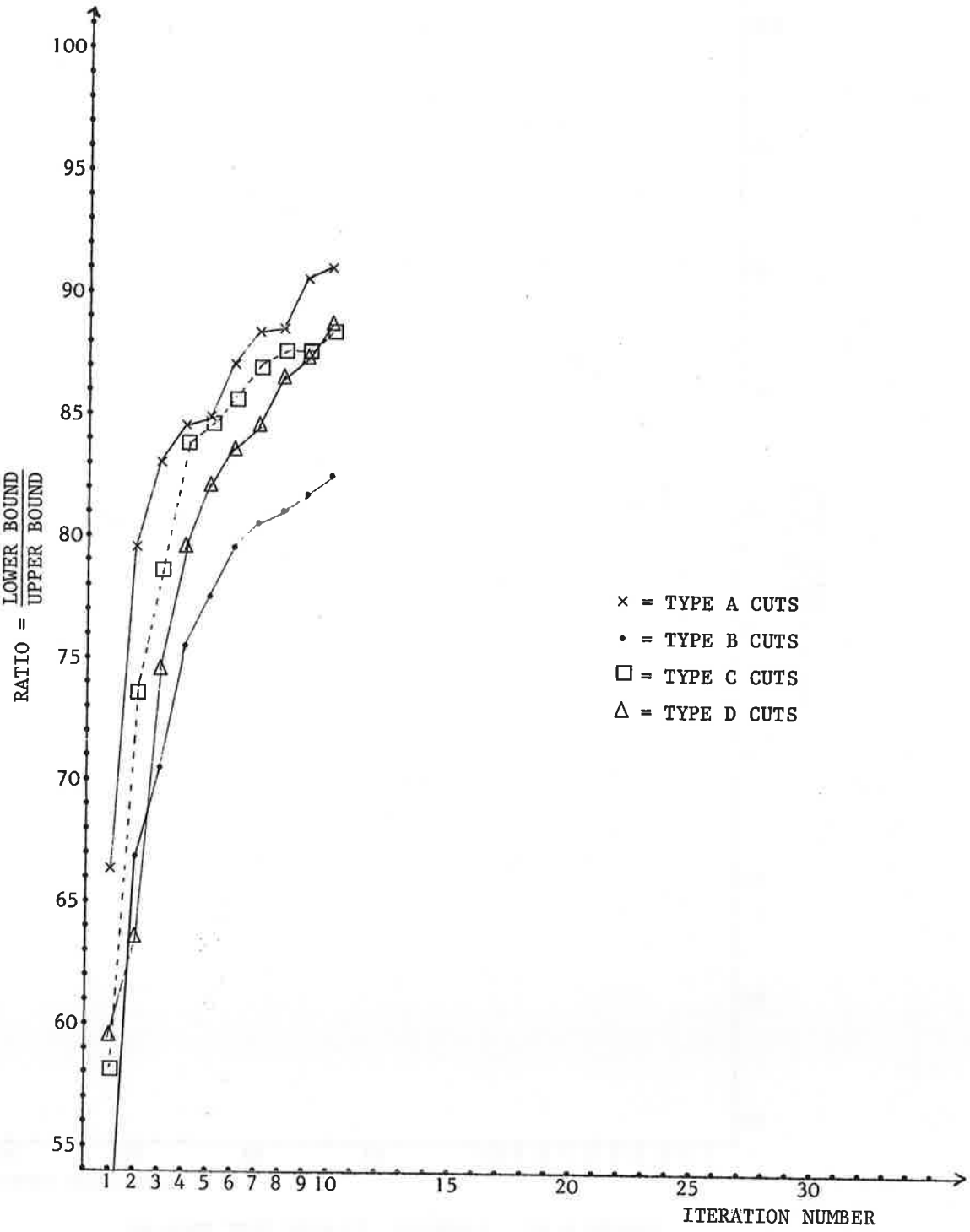


FIGURE 4.9 4-MEDIAN, 33-NODE TEST PROBLEM

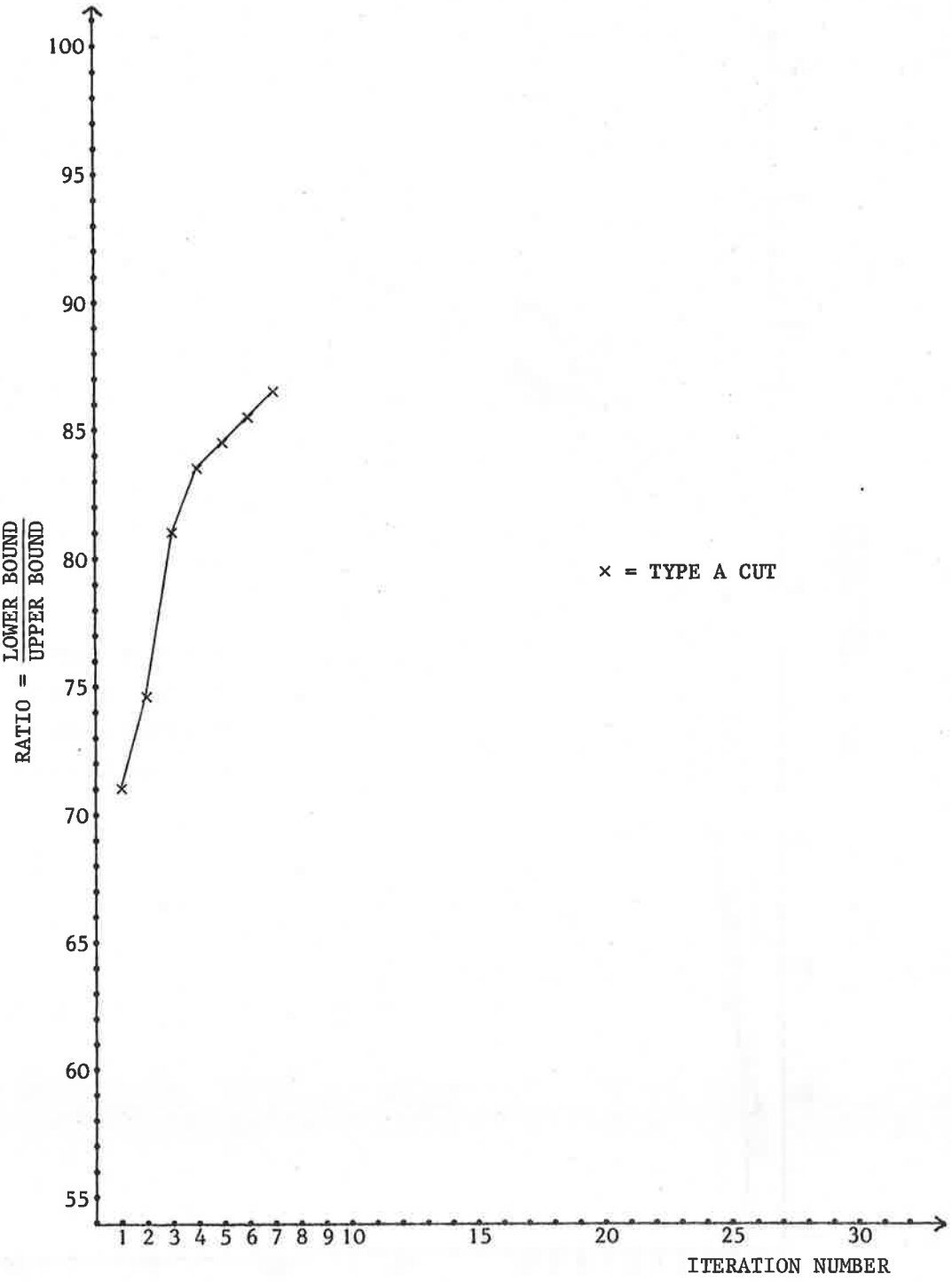


FIGURE 4.10 5-MEDIAN, 33-NODE TEST PROBLEM

REFERENCES FOR SECTION 4.

4-1 Cornuejols, G., M.L. Fisher and G.L. Nemhauser, "An Analysis of Heuristics and Relaxations for the Uncapacitated Location Problem," Discussion Paper No. 7602, Center for Operations Research and Econometrics, University of Louvain, January 1976.

4-2 Florian, M., G. Guerin, and G. Bushel, "The Engine Scheduling Problem in a Railway Network," INFOR J., 14, 121-138, 1976.

4-3 Garfinkel, R.S., A.W. Neebe and M.R. Rao, "An Algorithm for the M-Median Plant Location Problem," Trans. Sci., 18, 217-236, 1974.

4-4 Geoffrion, A.M., "Generalized Benders Decomposition," J. Opt. Theory and Applic., 10, 237-260, 1972.

4-5 _____ and G. Graves, "Multicommodity Distribution System Design by Benders Decomposition," Man. Sci., 20, 822-844, 1974.

4-6 Karp, R.L. and G.L. Thompson, "A Heuristic Approach to Solving Traveling Salesman Problems," Man. Sci., 10, 225-248, 1964.

4-7 Karp, R.M., "Reducibility Among Combinatorial Problems," in R.E. Miller and J.W. Thatcher (eds.), Complexity of Computer Computations, Plenum Press, 85-104, 1972.

4-8 _____, "On the Computational Complexity of Combinatorial Problems," Networks, 5, 45-68, 1975.

4-9 Lasdon, L.S., Optimization Theory for Large Systems, The Macmillan Company, NY, 1970.

4-10 Richardson, R., "An Optimization Approach to Routing Aircraft," Trans. Sci., 10, 52-71, 1976.

4-11 Rockafellar, R.T., Convex Analysis, Princeton University Press,

NJ, 1970.

4-12 Sahni, S., "Computationally Related Problems," SIAM J. Computing, 3, 262-279, 1974.

5. VEHICLE ROUTING PROBLEMS

5.1 HEURISTICS FOR GENERAL ROUTING PROBLEM

In this section we discuss vehicle routing and the traveling salesman problems (VRP and TSP). Proposed techniques for solving problems of this sort have fallen into two classes - those which solve the problem optimally by branch and bound techniques and heuristics. Since algorithms which are guarenteed to achieve optimality are viable only for very small problems, most authors have concentrated on the study of heuristic algorithms. Heuristic techniques for solving problems of this sort can be grouped as follows; "savings" procedures [5-6], "sweep" procedures [5-12], "nearest-neighbor" procedures [5-34], and "r-optimal" procedures [5-8].

Undoubtedly, the Clarke-Wright savings method, developed in 1964, is the most widely used and most widely cited vehicle routing algorithm. It involves first evaluating all potential savings $S_{ij} = d_{ij} + d_{lj} - d_{ij}$ from linking two nodes i and j, and then joining those nodes with the highest feasible savings at each iteration. Initially, we suppose that every two demand points i and j are supplied individually from two vehicles giving a total distance of $2d_{li} + 2d_{lj}$. Now if we used only one vehicle to service both nodes i and j, instead of two, then we would experience a savings in travel distance of $(2d_{li} + 2d_{lj}) - (d_{li} + d_{lj}) = d_{li} + d_{lj} - d_{ij}$.

For every possible pair of demand points i and j there is a corresponding savings S_{ij}. We order these savings from greatest to least and starting from the top of the list we link nodes i and j where S_{ij} represents the current maximun savings unless the problem constraints are violated. Christofides and Eilon found from ten small test problems that tours produced from the savings method averaged only 3.2 percent longer than the

optimal tours [5-4].

In 1974, Gillett and Miller [5-12] proposed a sweep algorithm for Euclidean networks which ranks and links demand points by their polar coordinate angle. We select a "seed" node randomly. With the central depot as the pivot, we start sweeping (clockwise or counterclockwise) the ray from the central depot to the seed. Demand nodes are added to a route as they are swept. If the polar coordinates (indicating angle) for the demand points are ordered from smallest to largest (with seed's angle 0), we enlarge routes as we increase the angle until capacity restricts us from enlarging a route by including an additional demand node. This demand point becomes the seed for the following route. Once we have partitioned the nodes, we can apply traveling salesman heuristics to improve tours and obtain significantly better results. In addition, we can vary the seed and select the best solution.

Tyagi [5-34], in 1968, presented a method which groups demand points into tours based on a nearest-neighbor concept. That is, points are added to a tour sequentially, each new addition being the closest point to the last point added to the tour. Having grouped the delivery points into m tours, we solve m traveling salesman problems to refine the tours.

Eilon et al. [5-8] study an r-optimal procedure for the VRP which is an outgrowth of Lin's approach to the traveling salesman problem [5-22]. The procedure presented by Eilon et al. begins with a feasible solution and tests perturbations of r arcs at a time until r-optimality is obtained. For example, if r=2 we examine each pair of arcs to see if it can be replaced by another pair such that feasibility is preserved and total distance is decreased.

Vehicle routing algorithms have recently "come of age" in the sense that they are now capable of solving some large-scale real-world problems. As we have emphasized, TSP's are often the crucial sub-problems for much larger VRP's. A 250-location problem with about 10 locations per route was solved on an IBM 360/67 in just under 10 minutes using the Gillett and Miller algorithm [5-12]. More recently, Golden, Magnanti, and Nguyen [5-15] have incorporated ideas from computer science into a modified savings procedure. In their paper, the authors emphasize data structures and list processing and present a new implementation of the Clarke-Wright algorithm which is motivated by optimality considerations, storage considerations, sorting considerations, and program running time. The Clarke-Wright algorithm is modified in the following three ways:

(1) by using a route shape parameter γ to define a modified savings $S_{ij} = d_{1i} + d_{1j} - \gamma_{ij} d_{ij}$ and finding the best route structure obtained as the parameter is varied;

(2) by considering savings only between nodes that are "close" to each other;

(3) by storing savings S_{ij} in a heap structure to reduce comparison operations and ease access.

This modified code performs from one to two orders of magnitude faster than other recently developed codes. For details concerning this approach, see the technical report [5-15] from this project.

In the next section, we show how the "savings" technique for solving the VRP, a generalization of the TSP, is used to solve large traveling salesman problems and to assess the quality of the solutions produced. A TSP

algorithm is presented which appears to be faster than many other heuristic algorithms for the TSP. The material presented is an extended version of Golden [5-13].

5.2 SPECIALIZATIONS: TRAVELING SALESMAN PROBLEM

In this section the "savings" technique for solving the vehicle routing problem, a generalization of the TSP, is used to solve large traveling salesman problems. Heuristic solutions are compared with expected values and statistical estimates of the optimal tour lengths. In particular, the Weibull distribution is used to model the distribution of heuristic solutions to the combinatorially explosive TSP. An algorithm is presented whose running time is especially encouraging. The approach is an example of the growing interface between statistics and mathematical optimization.

The traveling salesman problem arises in many different contexts; typical applications include computer wiring, vehicle routing, clustering, and job-shop scheduling [5-21]. Since TSP's with more than about 65 nodes cannot be solved optimally, in general, heuristic procedures are of special interest.

The objectives of our work are three-fold and include efficiency, accuracy, and evaluation. That is, we hope to provide an extremely fast heuristic TSP algorithm which yields solutions which are within 5 or 10 percent of the optimal solution. In addition, based on the heuristic, we want to be able to estimate the optimal tour length in order to evaluate the heuristic more suitably.

5.2.1 Algorithm Background

The Clarke-Wright algorithm is a "greedy" heuristic algorithm which has been implemented to solve large-scale vehicle routing problems. Initially, each demand node is serviced separately from a specified central depot (node 1). Nodes i and j become linked according to the magnitude of the savings

$$s(i,j) = d(1,i) + d(1,j) - d(i,j)$$

where d(i,j) is the distance from node i to node j. Figures 5.1 and 5.2 illustrate the savings formula. The basic motivation is as follows: If node i is an end-point of a tour (adjacent to the central depot) and i can be linked feasibly with nodes j and k, and $s(i,j) > s(i,k)$, then linking i and j would be preferred (in the short range) to linking i and k, although possibly not in the optimal solution. The algorithm proceeds myopically, choosing the best feasible savings at each iteration. See Clarke and Wright [5-6] and Golden, Magnanti, and Nguyen [5-15] for details. We have applied a modified version of the algorithm to a distribution system for an urban newspaper with an evening circulation exceeding 100,000. This problem contained nearly 600 drop points for newspaper bundles and was solved with 20 seconds of execution time on an IBM 370/168.

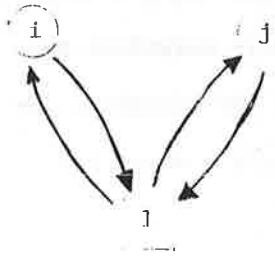


FIGURE 5.1 INITIAL SETUP

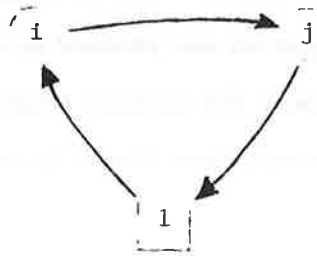


FIGURE 5.2 NODES I AND J HAVE BEEN LINKED

Golden, Magnanti, and Nguyen [5-15] have overcome some of the drawbacks of the original approach in their modified algorithm. The Clarke-Wright algorithm was designed initially to handle a matrix of real inputs, distances and savings. In dealing with a large problem, the number of required storage locations soon becomes unwieldy. Rather than consider all pairwise linkings, we can restrict our search to a small subset of the possible linkings, which we store in a list instead of a matrix. This is accomplished by superimposing an arbitrary grid over the nodes of the transportation network. Only the savings associated with arcs linking nodes in adjacent boxes are considered. Since the grid is arbitrary, care must be taken in the assignment of the parameters which determine the grid. At each step of the algorithm, we must determine the maximum savings. This comparison of savings is handled quickly and conveniently by partially ordering the data in a heap structure (see [5-15] for details) and updating the heap at each step after altering the routes.

We can use this savings approach for the solution of TSP's since a single vehicle of unlimited capacity leads to an appropriately defined VRP. In fact, the computer code can be streamlined to exploit this simplification, and extremely fast running times result. However, in some cases we sacrifice accuracy for speed to an unacceptable degree. Since we are building a hamiltonian circuit, we can consider any node to be the origin (this cannot be done in solving the VRP). If we consider several independent origins we can produce several independent heuristic solutions and simply choose the minimum length tour as our heuristic TSP solution. We gain accuracy, lose little in speed, and hopefully obtain a means for

evaluating our heuristic solution. The evaluation rests on the assumption that the independent heuristic solutions can be modeled as generated from a Weibull distribution, an issue we will take up later. The assumption of independent heuristic solutions needs, perhaps, some justification. We can generate N origins randomly from a network of n nodes with or without replacement. When we choose with replacement, the heuristic solutions will be independent. However, we prefer to select N distinct origins. When we choose without replacement we obtain more information since each new origin generates a new heuristic solution, and if the ratio N/n is small, then the heuristic solutions will be more or less independent. That is, the difference between sampling with replacement and without replacement is significant only when the population we are sampling from contains relatively few members. When the ratio N/n is small the selection and non-replacement of a particular node has a negligible effect on the probabilities of selecting additional nodes.

An underlying issue involves the evaluation of heuristic solutions to NP-complete problems, such as the TSP. Rosenkrantz, Stearns, and Lewis [5-30] and Christofides [5-5] have obtained, through elegant combinatorial arguments, some interesting (although not especially reassuring) worst-case performance bounds for approximate algorithms for the TSP when the triangle inequality holds. For the nearest-neighbor method, the worst-case ratio of the length of the obatined tour to the length of the optimal tour increases logarithmically with the number of nodes. The nearest-insertion method and the cheapest-insertion method have worst-case ratios which approach 2 as the number of nodes increases. Christofides' new heuristic has a worst-case ratio which is strictly less than 3/2. We have shown that

for a sequential version of the Clarke-Wright algorithm the worst-case ratio of the Clarke-Wright solution (from a single origin) to the optimal TSP solution can be arbitrarily as large as the number of nodes increases.

5.2.2 Discussion of the Algorithm

Every arc that we consider for linking in our modified Clarke-Wright algorithm has a corresponding savings relative to a particular origin. We order these savings from greatest to least on a heap and starting from the top of the list we link nodes i and j where s_{ij} represents the current maximum feasible savings. We continue until a tour on n nodes is formed. Linking nodes i and j is feasible so long as neither i nor j are interior to a subtour.

Now, when a number of distinct origins are generated for each problem, we can eliminate redundant calculations in the following way. First, define the arbitrary grid judiciously (set DIV). Given the x and y coordinates of the nodes in the network, the grid is divided into DIV^2 equally-sized rectangles so that each node i has box coordinates BX(i) and BY(i). Arcs with nodes in adjacent boxes are considered for linking; these arcs are stored along with their distances. In other words, if $|BX(i) - BX(j)| > 1$ or $|BY(i) - BY(j)| > 1$ then arc (i,j) is ignored. From computational experience, which will be discussed later in this paper, the number of nodes in the network suggests an appropriate range of values for the parameter DIV (see Golden, Magnanti, and Nguyen [5-15] for more details regarding DIV). As the origins are varied, only the savings values change.

The computer code reflects this observation. For each different origin o, we must redefine our savings function accordingly

$$S_{ij}(o) = d(o,i) + d(o,j) - d(i,j) \qquad (1)$$

and use the savings to construct a tour as discussed previously.

In the event that distances are not Euclidean, we can store the p nearest neighbors to each of the n nodes in a matrix. A shortest path algorithm can be used to find these nearest neighbors. Next, the data can be arranged in an n by 3p matrix T where for L = 1,2,...,p:

T(I,3L-2) = the node adjacent to node I,

T(I,3L-1) = the distance between nodes I and T(I,3L-2),

T(I,3L) = the savings obtained by linking I and T(I,3L-2).

This storage scheme is similar to the one developed by Williams [5-36] for shortest paths. For each node I, we can order the corresponding savings from largest to smallest via heap structures and proceed as before.

5.2.3 Statistical Evaluation

The Weibull family of distributions has been studied extensively in recent years, as evidenced by the journal Technometrics. It is especially useful in problems of life testing and reliability where the life length may be bounded from below (see Barlow and Proschan [5-1]. This family is, in some sense, a more generalized form of the exponential distribution since it has three parameters and can be reduced to the exponential distribution with the proper choice (c=1) of one of them. Gumbel [5-16] refers to the Weibull family as the Type III asymptotic distribution of extreme values. Fisher and Tippett [5-10], in their fundamental 1928 paper, were the first to derive the three asymptotic distributions. We will come back to this later. The probability density function for the Weibull distributions is

$$f_x(x_0) = \left(\frac{c}{b}\right)\left(\frac{x_0-a}{b}\right)^{c-1} \exp\left[-\left(\frac{x_0-a}{b}\right)^c\right] \text{for } x_0 \geq a \geq 0,\ b > 0,\ c > 0$$

where a is the location parameter, b is the scale parameter, and c is the shape parameter. It is often easier to work with the cumulative Weibull distribution given by

$$\text{Prob}\left\{x \leq x_0\right\} = 1 - \exp\left[-\left(\frac{x_0-a}{b}\right)^c\right].$$

Note that the random variable $(x-a)^c$ has an exponential distribution with expected value b^c since

$$\text{Prob}\left\{(x-a)^c \leq x_0\right\} = 1 - \exp\left[\frac{-x_0}{b^c}\right].$$

McRoberts [5-27], in dealing with combinatorially explosive plant-layout problems, introduced the idea of matching the distribution of heuristic solutions with a Weibull distribution. He further suggests that other combinatorial problems might be approached in a similar manner. Unfortunately, his arguments were substantially weakened by the fact that he treats his intermediate heuristic solutions as a set of independent observations from a parent distribution despite the fact that there exists a clear interdependence among these solutions. In part, this paper is an outgrowth of his work.

Consider N samples, each of size m, taken from a parent population which is bounded from below. In each sample there is a smallest value x_i, and the smallest value v in Nm observations is the smallest of the N smallest values x_i, i.e. $v = \min \{x_i \mid 1 \leq i \leq N\}$. All sampling is independent. Fisher and Tippett [5-10] demonstrated that when m is very large the distribution of x_i approaches what we now call the Weibull distribution. Gumbel [5-16] points out that in studying one extreme, no assumption need be made regarding the behavior of the initial distribution at the other extreme. The Weibull distribution is independent of the parent distribution, so we do not need to know the distribution from which the samples are generated. Weibull was an engineer who later derived, in an empirical way, the same distribution and applied it to the analysis of dynamic breaking strength [5-35].

Intuitively, it seems reasonable that the distribution of heuristic solutions could be Weibull. Suppose there are n nodes. Then, the parent populaton consists of (n-1)! /2 tours, with lengths bounded from below by the length of the optimal tour. Each independent heuristic solution implicitly is a local minimum from a large number m of possible tours. With this

in mind, we can perform the proposed algorithm from various origins to obtain heuristic solutions (tour lengths) $x_1, x_2, \ldots, x_N$. Next, using maximum likelihood estimation we might find best estimates for a, b, and c.

The likelihood function is given by

$$L(x_1,x_2,\ldots,x_N;\ a,b,c) = \left(\frac{c}{b}\right)\left(\frac{x_1-a}{b}\right)^{c-1} \exp\left[-\left(\frac{x_1-a}{b}\right)^c\right]$$

$$\cdots \left(\frac{c}{b}\right)\left(\frac{x_N-a}{b}\right)^{c-1} \exp\left[-\left(\frac{x_N-a}{b}\right)^c\right]$$

$$= \left(\frac{c}{b}\right)^N \left(\frac{1}{b}\right)^{Nc-N} \left\{(x_1-a)\cdots(x_N-a)\right\}^{c-1} \exp\left[-\sum_{i=1}^{N}\left(\frac{x_i-a}{b}\right)^c\right].$$

Since $L(\theta)$ and $\ln(L(\theta))$ have their maximum at the same value of θ, we maximize the natural logarithm of the likelihood

$$\ln L = N \ln c - Nc \ln b + (c-1) \sum_{i=1}^{N} \ln(x_i-a) - \sum_{i=1}^{N} \left(\frac{x_i-a}{b}\right)^c .$$

If we now equate $\frac{\partial \ln L}{\partial a} = \frac{\partial \ln L}{\partial b} = \frac{\partial \ln L}{\partial c} = 0$, we obtain the following maximum likelihood equations:

$$\frac{\partial \ln L}{\partial a} = -(c-1) \sum_{i=1}^{N} \left(\frac{1}{x_i-a}\right) + \left(\frac{c}{b^c}\right) \sum_{i=1}^{N} (x_i-a)^{c-1} = 0 ; \qquad (2)$$

$$\frac{\partial \ln L}{\partial b} = \frac{-Nc}{b} + \frac{c}{b^{c+1}} \sum_{i=1}^{N} (x_i-a)^c = 0 ; \qquad (3)$$

$$\frac{\partial \ln L}{\partial c} = \frac{N}{c} - N \ln b + \sum_{i=1}^{N} \ln(x_i-a) - \sum_{i=1}^{N} \left(\frac{x_i-a}{b}\right)^c \ln\left(\frac{x_i-a}{b}\right) = 0. \qquad (4)$$

ally been applied in situations where the theoretical cumulative distribution function is completely specified, i.e., all parameters are known. In many practical situations, however, some or all of the parameters are unknown in which case they must be estimated from the sample data; an approximate test results. Recent research in mathematical statistics has begun to focus on the exact nature of this approximation (see Stephens [5-32].

5.2.4 Expected Length of the Optimal Tour

Beardwood, Halton, and Hammersley [5-2] derive the asymptotic expected length of an optimal traveling salesman tour for a special class of networks. In addition, they prove that the variance of the length goes to zero as n becomes large. For an n node problem (n large) where the nodes are distributed randomly and uniformly over some arbitrary area of S units, the expected length of the optimal TSP tour, L(n,S), is given by

$$L(n,S) = K\sqrt{n}\,\sqrt{S}\ . \qquad (5)$$

In their excellent book, Eilon, Watson-Gandy, and Christofides [5-8] perform simulations which indicate that $K = 0.75$ approximately; the expected length formula is reasonably accurate with $K = 0.75$ although one could argue that it consistently underestimates the optimal tour length (see Figure 8.18 [5-8]). This formula is important since for large n we cannot solve TSP's exactly.

We now have two means for estimating the optimal TSP solution. Our goal is to test the Weibull estimate against the expected value formula (5)

for moderate-sized problems (70 to 130 nodes) where the nodes are distributed randomly over an area in Euclidean space. Of course, it can be argued that for very large networks when the conditions are not satisfied the expected length formula still provides an excellent approximation. This, however, is difficult to substantiate. Moreover, the statistical estimate is problem-dependent, and so it takes into account, rather than ignores, the possible nonrandomness of nodes.

5.3 COMPUTATIONAL RESULTS

We have performed extensive computational tests essentially to address the following three questions:

(i) What is the efficiency and accuracy of the proposed heuristic solution technique?

(ii) Can the Weibull hypothesis be justified statistically?

(iii) How do the estimated solutions and the expected solutions compare?

The results obtained tend to confirm our intuition: Tables 5.1 and 5.2 display the numerical findings.

Table 5.1 focuses on questions (i) and (ii). Networks of from 70 to 130 nodes were generated randomly in a square of area 10,000. In Table 5.1, for each value of n(n = 70,80,...,130) a single network was generated. The savings heuristic was applied from N distinct origins with a grid of DIV^2 boxes to obtain a final heuristic solution. Running times are remarkably fast, ranging from 16 to 40 seconds in total execution time on an IBM 370/168. These run times include all input and output operations (time spent generating networks is also included). The final heuristic solutions are within 4-10% percent of the expected solution, except in one instance. The parameters a, b, and c have been estimated and the location parameter a seems to be slightly above the expected solution in general. Finally, in all cases, the observed Kolmogorov-Smirnov statistics fall far below the critical value at the .05 level of significance.

Table 5.2 deals with the third question. Here, for each value of n, five networks have been generated as before, in order to study average behavior. The average estimated solution is compared with the expected solu-

TABLE 5.1 COMPUTATIONAL RESULTS

n	N	DIV	Running Time (seconds)	Expected Solution (1)	Heuristic Solution (2)	Approximate Deviation $\frac{(2)-(1)}{(1)}$	Parameter Estimation			Observed K-S Statistic	Critical Value
							a	b	c		
70	25	4	16	627.75	659.39	5.0%	650.	37.78	1.95	.083	.27
80	25	4	21	670.50	700.46	4.5%	695.	37.77	1.56	.097	.27
90	25	5	19	711.75	747.24	5.0%	740.	35.74	2.01	.111	.27
100	25	5	24	750.00	785.73	4.8%	720.	117.42	4.83	.114	.27
110	30	6	33	786.75	832.44	5.8%	770.	111.65	6.18	.106	.24
120	30	6	40	821.25	915.36	11.4%	870.	82.30	4.90	.086	.24
130	30	7	37	855.00	917.56	7.3%	900.	91.72	2.00	.084	.24

Each entry refers to one network.

n = the number of nodes

N = the number of origins

DIV is the parameter which determines the grid.

TABLE 5.2 COMPUTATIONAL RESULTS

n	N	DIV	Expected Solution (1)	Average Estimated Solution (2)	Average Heuristic Solution (3)	Approximate Deviation $\frac{(2)-(1)}{(1)}$	Approximate Deviation $\frac{(3)-(1)}{(1)}$
70	25	4	627.75	643.	690.	2.4%	9.9%
80	25	4	670.50	695.	727.	3.6%	8.4%
90	25	5	711.75	740.	763.	3.9%	7.2%
100	25	5	750.00	774.	804.	3.2%	7.2%
110	30	6	786.75	805.	836.	2.3%	6.2%
120	30	6	821.25	837.	879.	1.9%	7.0%
130	30	7	855.00	899.	917.	5.1%	7.2%

Each entry is an average over five networks.

TABLE 5.3 CORRELATION COEFFICIENT AS A FUNCTION OF LOCATION PARAMETER

Location Parameter	Correlation Coefficient
850	.970
855	.971
860	.972
865	.973
870	.974
875	.975
880	.976
885	.977
890	.979
895	.980
900	.982
905	.981
910	.977
915	.949

tion. In all cases the average estimated solution is above and within about five percent of the expected solution. This represents a fairly close fit. Furthermore, if K is increased to 0.76, as perhaps it should be (although 0.75 is certainly more convenient), the fit becomes still better.

For all problems in Table 5.1, estimating the location parameter was especially easy. Plotting correlation coefficient as a function of location parameter we find a unimodal function as indicated in Table 5.3 where the entries correspond to the case n = 130 from Table 5.1.

Next, we decided to construct a test network where the expected length formula was clearly not applicable. In a square area of 64 units, we let the 81 node locations in our network be at the integer lattice points (i,j) contained in the region. The proposed heuristic was applied and the observed Kolmogorov-Smirnov statistic was found to fall far below the critical level, as in the Table 5.1 experiments. The expected solution and the heuristic solution differed by more than 50 percent here, emphasizing the restrictions associated with the expected length formula. In addition, we solved the 25-city problem posed by Held and Karp [5-11] who conjectured up an optimal solution of length 1711 units. Little et al [5-24] later verified this conjecture. The expected length formula does not hold in this case either. In a second of computer time, tours from ten distinct origins were generated. Our algorithm's solution of 1750 units is about 2.2 percent above the optimal solution. The estimated optimal solution under the Weibull assumption was found to be 1725 units, less than 1 percent away from optimality. The observed Kolmogorov-Smirnov statistic was 0.071 against a critical value of 0.410 at the 0.05 level of significance. These experiments have signaled that indeed, the Weibull assumption can be exploited in more general situations than the expected value result.

Finally, we sought to determine the discriminating power of the Kolmogorov-Smirnov test. The normal distribution provides a fairly accurate approximation to the Weibull in this problem, especially in terms of central tendency. We experimented with problem #24 from Krolak et al. [5-20] (some of our results will be mentioned in the next section) first assuming an underlying Weibull distribution, then an underlying normal distribution. The observed Kolmogorov-Smirnov statistic is 0.056 under the Weibull hypothesis, and 0.130 under the normal hypothesis - quite a difference. Although neither assumption would be rejected at the 0.05 level of significance, the Weibull assumption yields a much closer fit.

5.3.1 Other TSP Heuristics

The interchange algorithm of Lin and Kernighan [5-23] is still the most effective procedure available for generating optimum and near-optimum solutions to the symmetric TSP. The heart of their procedure involves a transformation whereby k edges in the current tour are replaced by k other edges, yielding a better tour. They claim to have solved 100 node problems exactly with 99 percent confidence in 3-4 minutes running time on a GE 635. In contrast, our procedure solves 130 node problems to within approximately 7 percent of the optimal solution in about 37 seconds.

Since Lin and Kernighan and Krolak et al. have each obtained the solution 21282. for problem #24 of reference [5-20], it is believed to be the optimal solution. In applying our TSP heuristic, we obtained a solution of 21978. in 22 seconds of IBM 370/168 execution time which is only 3.3 percent away from

optimality. In addition, our estimate for the location parameter was 21650, about 1.7 percent above the optimal solution.

Our heuristic is faster than the Lin-Kernighan heuristic although not as accurate, and much faster than the Krolak et al. man-machine approach. For problems involving more than 110 nodes the computer storage requirements become too excessive for the present version of the Lin-Kernighan heuristic. Modifications are indicated (although not implemented) in their paper. Storage is not a problem with the proposed algorithm since we are selective in the arcs which we choose to consider for linking, and larger problems can be handled without difficulty. It is also conceivable that a hybrid approach in which a solution from our heuristic becomes an initial feasible solution for the interchange algorithm might be very successful.

We should point out that Steiglitz and Weiner [5-31] have presented improved implementations for Lin's 2-opt and 3-opt methods [5-32] (precursors to the sophisticated Lin-Kernighan heuristic). A tour is called K-opt if it is not possible to obtain a tour with smaller cost by exchanging K arcs in the tour for any other set of K arcs. For example, if K = 2 each pair of arcs is examined to see if it can be replaced by another pair such that feasibility is preserved and total distance is decreased. Steiglitz and Weiner performed experiments on a CDC 6600 computer which indicate that Lin's 2-opt method, in particular, can be made to run just as fast as our proposed heuristic with a similar average percentage error. In a future paper, we hope to compare these two heuristics.

5.3.2 Conclusions

If we examine the five 100 node problems in Krolak et al. [5-20] (we presume all nodes have been generated randomly in a rectangle of 2000 by 4000 units for each problem) and if we conjecture that the Lin-Kernighan solutions are optimal, then we can compare the average optimal solution 21508. with the expected optimal tour length. Setting K to .76 rather than .75 we obtain an expected length of about 21500. The limited evidence suggests that .75 is probably too conservative (low) for K. On the other hand, there may be a complex underlying bias mechanism which causes our location parameter estimates to consistently overestimate by 1 or 2 percent. Notice from Table 5.2 that there are no trends towards greater disparity as n increases.

Recent results in complexity theory indicate that many network optimization problems such as the TSP are inherently difficult to solve. In fact, it seems unlikely that polynomial algorithms can be obtained for exact solution to these problems. With this in mind, heuristic algorithms have become increasingly important. In this paper, we have presented an efficient and accurate heuristic algorithm for approximate solution of the TSP. Of course, if an improved solution is desired, we can determine some or all of the n-N remaining heuristic solutions in a reasonable amount of computer time. In addition, we have provided a statistical approach for estimating the optimal solution, which will help in assessing deviations from optimality. On the basis of our computational results we cannot reject the null hypothesis that, indeed, an underlying Weibull distribution is at work. On the other hand, as with all statistical arguments, we cannot arrive at an absolutely firm conclusion. The suggested statistical approach, along with sharp lower bounds from lagrangean relaxation [5-18], [5-19], and

combinatorial worst-case ratio analysis [5-5], [5-30], is yet another means for evaluating heuristics for hard combinatorial optimization problems.

We anticipate that heuristic solutions from the 2-opt procedure could also be successfully modeled by a Weibull distribution. In addition, we feel this statistical evaluation procedure can be used for many other combinatorially intractable problems.

REFERENCES FOR SECTION 5.

5-1 Barlow, R., and F. Proschan, Mathematical Theory of Reliability, John Wiley & Sons, NY, 1965.

5-2 Beardwood, J., J. Halton, and J. Hammersley, "The Shortest Path Through Many Points," Proc. Camb. Phil. Soc., 55, 299, 1959.

5-3 Beltrami, E., and L. Bodin, "Networks and Vehicle Routing for Municipal Waste Collection," Networks, 4 (1), 65-94, 1974.

5-4 Christofides, N., and S. Eilon, "An Algorithm for the Vehicle Dispatching Problem," Opnl. Res. Q., 20, 309, 1969.

5-5 Christofides, N., "Worst-Case Analysis of a New Heuristic for the Traveling Salesman Problem," Presented at the Symposium on New Directions and Recent Results in Algorithms and Complexity, Pittsburgh, Pa, April 1976.

5-6 Clarke, G., and J. Wright, "Scheduling of Vehicles from a Central Depot to a Number of Delivery Points," Opers. Res., 12, 568, 1964.

5-7 Dantzig, G. and J. Ramser, "The Truck Dispatching Problem," Man. Sci., 6, 81-91, 1959.

5-8 Eilon, S., C. Watson-Gandy, and N. Christofides, Distribution Management, Griffin, London, 1971.

5-9 Fiacco, A., and G. McCormick, Nonlinear Programming: Sequential Unconstrained Minimization Techniques, John Wiley & Sons, NY, 1968.

5-10 Fisher, R., and L. Tippett, "Limiting Forms of the Frequency Distribution of the Largest or Smallest Member of a Sample," Proc. Camb. Phil. Soc., 24, 180, 1928.

5-11 Garvin, W., H. Crandall, J. John, and R. Spellman, "Applications of

Linear Programming in the Oil Industry," Man. Sci., 3(4), 407-430, 1957.

5-12 Gillett, B., and L. Miller, "A Heuristic Algorithm for the Vehicle Dispatch Problem," Opers. Res., 22, 340, 1974.

5-13 Golden, B., "A Statistical Approach to the TSP," MIT Operations Research Center Working Paper OR 052-76, April 1976.

5-14 _____, "Large-Scale Vehicle Routing and Related Combinatorial Problems," Ph.D. Dissertation, MIT Operations Research Center, June 1976.

5-15 Golden, B., T. Magnanti, and H. Nguyen, "Implementing Vehicle Routing Algorithms," MIT Operations Research Center Technical Report No. 115, September 1975 (to be published in Networks).

5-16 Gumbel, E., Statistics of Extremes, Columbia University Press, NY, 1958.

5-17 Held, M., and R. Karp, "A Dynamic-Programming Approach to Sequencing Problems," J. SIAM, 10, 196-210, 1962.

5-18 "The Traveling Salesman Problem and Minimum Spanning Trees," Opers. Res., 18 (6), 1138-1162, 1970.

5-19 Held, M., and R. Karp, "The Traveling Salesman Problem and Minimum Spanning Trees: Part II," Math. Prog., 1, 6-25, 1971.

5-20 Krolak, P., W. Felts, and G. Marble, "A Man-Machine Approach Toward Solving the Traveling Salesman Problem," CACM, 14, 327-334, 1971.

5-21 Lenstra, J., and A. Rinnony Kan, "Some Simple Applications of the Traveling Salesman Problem," Opnl. Res. Q., 26 (4), 717-733, 1975.

5-22 Lin, S., "Computer Solutions of the TSP," Bell Systems Tech. J., 44, 2245, 1965.

5-23 Lin, S., and B. Kernighan, "An Effective Heuristic Algorithm for the Traveling Salesman Problem," Opers. Res., 21, 498-516, 1973.

5-24 Little, J., K. Murty, D. Sweeney, and C. Karel, "An Algorithm for the Traveling Salesman Problem," Opers. Res., 11 (5), 972-989, 1963.

5-25 Markham, R., "Some Aspects of the Weibull Distribution," Masters Thesis, Department of Mathematical Statistics, University of South Africa, 1974.

5-26 McRoberts, K., "Optimization of Facility Layout," Ph.D. Thesis, Iowa State University of Science and Technology, Ames IA, 1966.

5-27 _____, "A Search for Evaluating Combinatorially Explosive Problems," Opers. Res., 19, 1331, 1971.

5-28 Newton, R., and W. Thomas, "Bus Routing in a Multi-School System," Computer and Operations Res., 1 (2), 213-222, 1974.

5-29 Ravenis, J., "Life Testing: Estimating the Parameters of the Weibull Distribution," IEEE International Convention Record, 18-33, 1963.

5-30 Rosenkrantz, D., R. Stearns, and P. Lewis, "Approximate Algorithms for the Traveling Salesperson Problem," Proc. of the 15th IEEE Symposium on Switching and Automata Theory, 33-42, 1974.

5-31 Steiglitz, K., and P. Weiner, "Some Improved Algorithms for Computer Solution of the Traveling Salesman Problem," Proc. 6th Annual Allerton Conf. on lircuit and Systems Theory, 814-821, 1968.

5-32 Stephens, M., "Asymptotic Results for Goodness-of-fit Statistics with Unknown Parameters," Ann. Statist., 4 (2), 357-369, 1976.

5-33 Turner, W., P. Ghare, and L. Fourds, "Transportation Routing Problem - A Survey," AIIE Transactions, 6 (4), 288-301, 1974.

5-34 Tyagi, M., "A Practical Method for the Truck Dispatching Problem,"

J. Opers. Res. Soc. Japan, 10, 76-92, 1968.

5-35 Weibull, W., "A Statistical Distribution Function of Wide Applicability," J. Appl. Mech., 18, 293, 1951.

5-36 Williams, T., "The Shortest Path Problem: An Intermediate Stage Results Approach," presented at ORSA Meeting, Las Vegas, NV, November 1975.

5-37 Winkler, R., and W. Hays, Statistics: Probability, Interference, and Decision, Holt, Rinehart, and Winston, NY, 1975.

6. SUMMARY

6.1 FREIGHT FLOW PROBLEMS

The major result of the first year's work on these problems has been the development of a generalized method of generating improved "cuts" for the application of Benders decomposition. As applied to several p-median and uncapacitated network design models, the new stronger cuts showed accelerated convergence, particularly for the p-median problem when the ratio of locations to nodes was relatively high.

Since Benders decomposition requires the solution of an integer program at each iteration, this reduction in the number of iterations required for solution is significant in moving to applications for large scale transportation problems. However, we still require a successful method for solving these integer master programs, and this has been a barrier in our first year's work. In the second year, we are implementing Benders decomposition using the IBM MPSX coding to solve this master integer program, and will investigate other special integer codings.

6.2 MULTIFLEET ROUTING PROBLEMS

Our work on this problem consisted of implementing and testing codings which applied both price-directive and resource-directive methods to a series of multicommodity and multifleet test problems. The major result is the suprising success of the Dantzig-Wolfe price-directive methods on the multifleet problem. The Dantzig-Wolfe algorithm was found to reach an optimum in a small number of steps rather than approaching the optimum in an asymptotic fashion over a large number of iterations.

Thus, we seem to have found a successful decomposition technique for large scale multifleet routing problems. We are now implementing a Dantzig-Wolfe/OKF coding which links MPSX and an Out-of-Kilter flow coding at MIT/FTL for further large scale testing.

6.3 VEHICLE ROUTING PROBLEMS

For this problem, we have made substantial improvements in the performance of a heuristic technique for solving multivehicle routing problems, and the traveling salesman problem. By implementing a grid technique, and using a heap structure to store data, we have been able to use the original Clarke-Wright algorithm to solve problems from one to two orders of magnitude faster than other recent codes. This converts to an ability to solve much larger vehicle routing problems at a scale now compatible with real world applications.

For example, where the largest optimal solution for a single depot vehicle routing problem seems to be one with 23 locations, the above technique obtains answers for problems with 50 points in 1 second, and problems with 600 points in 20 seconds on an IBM 370/168. It has also obtained answers from 50 points with 4 depots in 3 seconds, and 100 points with 4 depots in 9 seconds. Applied to the traveling salesman problem, it has obtained answers for a 130 point problem in 37 seconds. In all cases, we expect the heuristic values to be within 10 percent of the optimal solution value which we feel is compatible with the accuracy of input data from typical applications.

APPENDIX

REPORT OF INVENTIONS

This research project produced advances in several areas of transportation systems analysis. Three generic transportation problems were considered: the multi-fleet scheduling problem, the freight flow network problem, and the vehicle routing problem. This research produced revised formulations of these problems and several solution techniques were proposed for analysis. Special use was made of the concept of decomposition in developing solution techniques. The project produced computational experience in problem solutions using network decomposition that will improve understanding of the capabilities and performance characteristics of the decomposition approach to solving transportation network problems.

110 Copies

U. S. DEPARTMENT OF TRANSPORTATION
TRANSPORTATION SYSTEMS CENTER
KENDALL SQUARE, CAMBRIDGE, MA. 02142

OFFICIAL BUSINESS
PENALTY FOR PRIVATE USE, $300

POSTAGE AND FEES PAID

U. S. DEPARTMENT OF TRANSPORTATION

513