ACCELERATING BENDERS DECOMPOSITION FOR NETWORK DESIGN

by

Richard Tekee Wong

S.B., Massachusetts Institute of Technology
1972

S.M., Massachusetts Institute of Technology
1972

E.E., Massachusetts Institute of Technology
1973

SUBMITTED IN PARTIAL FULFILLMENT OF THE

REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 1978

Signature of Author.....
Department of Electrical Engineering and Computer Science
February 1978

Certified by .....
Thesis Supervisor

Accepted by .....
Chairman, Departmental Committee on Graduate Students

ACCELERATING BENDERS DECOMPOSITION FOR NETWORK DESIGN

by

Richard Tekee Wong

Submitted to the Department of Electrical Engineering and Computer Science on February 3, 1978 in partial fulfillment of the requirements for the Degree of Doctor of Philosophy.

ABSTRACT

Network design problems arise in many different application areas such as air freight, highway traffic and communications systems. This thesis concerns the development, analysis and testing of new techniques for solving network design problems.

We study the application of Benders decomposition to solve mixed integer programming formulations of the network design problem. A new methodology for accelerating the convergence of Benders decomposition for network design is introduced. These acceleration techniques are also applicable to a much broader class of algorithms that includes Benders decomposition for general mixed integer programs, Dantzig-Wolfe decomposition for linear and nonlinear programs and other related procedures.

These methods are specialized to network problems by the development of very efficient algorithms that exploit the underlying structure of these models. Computational experience demonstrating the value of these techniques is given.

Another important issue in improving the computational performance of Benders decomposition for network design and other mixed integer programming models is the selection of a "good" mixed integer programming problem formulation. We give a criteria for choosing between two mathematical formulations of the same problem in the context of Benders decomposition.

Since good heuristic methods are important in generating initial starting points for Benders decomposition, we study the accuracy of heuristic solution procedures for a particular type of network design problem. Worst-case performance measures of heuristic network design procedures are presented. For a restricted version of the network design model, we describe a procedure whose maximum percentage of error is bounded by a constant. For a more general version of the problem, we give results concerning the

complexity of finding efficient (polynomially time bounded) heuristics for the network design problem.

THESIS SUPERVISOR: Thomas L. Magnanti

TITLE: Associate Professor of Operations Research and Management

ACKNOWLEDGMENTS

I would like to express my sincerest thanks to Professor Thomas Magnanti not only for his outstanding inspiration and guidance but also for his understanding and friendship which have made my work with him into a uniquely rewarding experience. Also his competition as a tennis opponent and his English lessons on active voice are appreciated.

I would like to thank my readers, Professors Sanjoy Mitter, Jeremy Shapiro and Robert Simpson for their comments and suggestions. Professor John Wozencraft (now at the Naval Postgraduate School) also provided some helpful discussions on this work.

Everyone at the M.I.T. Operations Research Center provided a friendly, helpful environment for this work. I would especially like to thank Jim Yee (now at the University of Maryland) and Bruce Golden (now at the University of Maryland) for their help and support. Also Fred Shepardson provided several productive discussions on the results of chapter III.

I would like to thank the staff at the Center for Operations Research and Econometrics (CORE), now at Louvain-la-Neuve, Belgium, for their hospitality during my year as a visitor. I am indebted to Etienne Loute of Core for his patience and invaluable advice in using the CORE computer facilities.

My thanks also to Peeter Kivestu for his help in conducting the computational experiments at M.I.T.

I wish to thank Annie Cooper for her excellent typing of his manuscript and for her help in the last final rush to complete this thesis.

I would like to thank Professor Richard Larson who encouraged my initial study of Operations Research and acted as my academic advisor for several years.

Finally I would like to express my sincere thanks to Professor Alvin Drake for both the opportunity to learn some probability through teaching his course Probabilistic Systems Analysis and for his concern and understanding in helping me through the entire doctoral program.

This work was supported in part by the Department of Transportation Advanced Research Program (TARP) contract #DOT-TSC-1058.

TABLE OF CONTENTS

TABLE OF CONTENTS

(continued)

CHAPTER I

INTRODUCTION

Federal Express currently guarantees overnight delivery of small packages between any of numerous cities in the United States. Of the fleet of 41 Falcon jets that it maintains, at least one plane leaves and returns to each of about 30 strategically located airports every night. Most planes meet in Memphis, Tennessee, where goods are reassigned according to their points of destination [18].

This system illustrates the use of Memphis as a "break-bulk" center. By aggregating shipments to and from break-bulk centers, and by reducing travel distance, this type of distribution system has the potential to reduce routing costs and improve service. These benefits must be weighed against costs for owning and operating the centers. Break-bulk centers are used in air, trucking, rail and other forms of freight distribution.

Choosing the sites and number of break-bulk centers can be modeled as a special case of the generic mixed integer program:

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

In this formulation x is an n-vector of continuous variables, y is a k-vector of discrete variables, and Y is a subset of the integer points

in k-dimensions. The matrices A and D and vectors c, d and b have dimensions compatible with those of x and y.

For the general break-bulk problem, each y_j is a binary, or indicator, variable specifying whether or not center j is opened. x_{ij}^k is the amount of product k routed from city (junction) i to city (junction) j at per unit cost c_{ij}^k. d_j is the fixed cost for owning a center at location j. The products k frequently will be distinguished only by their points of origin and destination. The model might include intermediate transshipment points in addition to the origins, destinations, and break-bulk centers, and it might permit shipments through more than one break-bulk center. The constraints of the model, which will be discussed formally in chapter III, account for demand requirements between the origin and destination points and specify that no material is routed through a center that is not opened.

The break-bulk problem also belongs to a more general class of network model known as the network design problem. In chapter 2 several variants of the network design problem are discussed. One versions (which contains the break-bulk problem as a special case) has the following description: there is a set of nodes and arcs; between every pair of nodes there is a required flow that must be routed through the network. Each arc has a capacity which can either be zero or infinity. A construction cost is incurred for setting an arc capacity to infinity. There are also arc routing costs which are linear functions of the total arc flow. The network design problem is to select a network (i.e. a set of arcs with infinite capacity) so that all the required flows are satisfied and the total construction and routing costs are minimized.

In addition to the break-bulk problem, the network design problem includes a wide variety of other applications. For example, models similar to the above network design variant have been used to locate concentrators in communications networks (Drinkwater [29]), to locate bank accounts to maximize float (Cornuejols, Fisher and Nemhauser [21]), to design production schedules (Wagner and Whitten [132] and Billheimer [13]) and to locate urban transit stations (Billheimer [13]).

Other variants of the network design problem have been used to design rail networks (Barbier [7] and Haubrich [117]), highway networks (Steenbrink [117]) and communication networks (McCallum [85]).

In the succeeding chapters of this thesis we study the solution of network design problems, which include the break-bulk problem and other applications. Our goal is to present, analyze and test new techniques for increasing the current capabilities for solving network design models.

The next chapter is a survey of network design problems and their solution methods. In this survey, we specify a general network design model and use it to analyze and relate a large number of network design papers. The purpose of this chapter is to describe the current uses and computational capabilities of network design models.

In the third chapter we consider the application of a decomposition procedure, Benders algorithm, to solve a mixed integer programming formulation of the network design problem. We present a new methodology for accelerating the convergence of Benders procedure for this type of problem. In fact, these acceleration techniques are applicable to a much broader problem setting that includes Benders decomposition for general mixed

integer programs, Dantzig-Wolfe Decomposition for linear and nonlinear programs, and related "cutting plane" type algorithms that arise in decomposition techniques.

After describing this general methodology, we next present specializations to network problems, developing very efficient algorithms that exploit the underlying structure of these models. We also describe other related techniques for improving the performance of Benders decomposition in solving various network problems such as facility location and network design. Finally, we describe computational experience demonstrating the value of our techniques in solving facility location and network design problems.

In the fourth chapter, we discuss another important issue in the application of Benders decomposition to network design and other mixed integer programming models, namely the selection of a "good" mixed integer programming problem formulation. Two different formulations of the same problem might be identical in terms of feasible solutions, but might be distinguishable in others ways. For example, they might have different linear programming or Lagrangian relaxations, one being preferred to the other when used in conjunction with algorithms like branch and bound or Benders decomposition. Geoffrion and Graves [49] have shown that proper model formulation can greatly improve the computational performance of Benders procedure.

One aspect of this discussion is a criteria for selecting between two mixed integer programming problem formulations in the context of Benders decomposition procedure. We discuss the application of this criteria for solving network optimization problems, and give examples

demonstrating its usefulness.

In the fifth chapter we consider the use of heuristic techniques to solve network design problems. In many large scale applications, approximate techniques are the only methods available for generating acceptable answers. Also, heuristic solutions are useful as good initial starting points for Benders decomposition and other procedures that guarantee optimality.

A major drawback to the use of heuristic algorithms has been the uncertainty concerning the accuracy of these approximate techniques. A recent trend in both operations research [21] and computer science [39] has been to analyze heuristics in terms of their worst case performance. That is, we evaluate a heuristic by its maximum possible error in approximating the optimal solution. Using this type of analysis, we can guarantee the performance of a procedure to within prespecified bounds.

Our analysis in this chapter provides worst-case performance measures of heuristic procedures for network design problems that are closely related to the break-bulk problem. For some versions of the network design model, we describe procedures whose maximum percentage of error is bounded by a constant. For more general versions of this problem, we derive upper and lower performance bounds for the class of all reasonable heuristic algorithms (i.e. algorithms whose computation time grows polynomially with problem size). These analyses also allow us to give worst-case bounds for several network design heuristics given in the literature.

Finally in the last chapter we present some concluding remarks about the various results presented and give suggestions for future work.

CHAPTER II

A SURVEY OF NETWORK DESIGN PROBLEMS

2.1 Introduction

The selection of an optimal configuration or design of a network occurs in many different application contexts including transportation (airline, railroad, traffic and mass transit), communication (telephone and computer network), electric power systems, and oil and gas pipelines. For example, consider a traffic network whose nodes represent both origin and destination areas for the vehicular traffic of a city and also intersections in the road network. The arcs correspond to streets in the city, and the arc flows denote the amount of traffic traversing the streets. A typical network design problem would be to select a subset of the possible road improvements subject ot a budget constraint. The design objective would be to minimize the total travel cost for all travelers in the city network.

In this survey, we will introduce a basic network design model which frequently occurs in the network literature. Although most real-world network design problems are more complicated than our general model, we believe that our basic framework embodies many of the most essential features of network design problems. Thus, any sophisticated design model will have to deal with the issues represented in our general framework.

We will discuss a number of network design papers in terms of this basic model. Although this general framework is applicable to many

different problem domains, we will concentrate mainly on transportation network problems since most of the work concerning network design has focused on these applications. Our goal is to present a coherent unified view of these papers and their contribution to the network design literature. We will review suggested solution procedures, computational experience, relations between various network models, and potential application areas. We also indicate promising areas of research for improving, solving, and extending the models reviewed in this survey.

Previous survey work in the area of network design problems includes reports by MacKinnon [76], Schwartz [113], Stairs [116,] and Steenbrink [118].

2.2 Problem Formulation

In this section we will give a general framework for the network design problems that will be discussed in this survey.

Our basic network design model has the following description: we have sets of nodes N and arcs A; between each pair of nodes $(k,\ell)\in N\times N$ there is a required flow $R_{k\ell}$ that must be routed through the network. $f_{ij}^{j\ell}$ is the amount of required flow between nodes k and ℓ on arc (i,j).

For each node $i\in N$ we can write the flow conservation equations:

$$\sum_{j\in N} f_{ji}^{k\ell} - \sum_{j\in N} f_{ij}^{k\ell} = \begin{cases} -R_{k\ell} & k=i \\ R_{k\ell} & \ell=j \\ 0 & \text{otherwise} \end{cases} \qquad (2.1)$$

$$(k,\ell) \in N\times N.$$

For each arc in the network we assume there is an initial given capacity u_{ij} and a set of possible capacity improvement levels L_{ij}. Thus we can write the following arc capacity constraints:

$$\sum_{(k,\ell)\in N\times N} f_{ij}^{k\ell} = \hat{f}_{ij} \leq u_{ij} + \ell_{ij} \qquad (i,j)\in A$$
$$\ell_{ij} \in L_{ij} \qquad (2.2)$$

For example, suppose all capacities are initially zero and an arc (i,j)'s capacity can either remain zero or be increased to a value K_{ij}. Then $L_{ij} = \{0, K_{ij}\}$ and (2.2) can be represented by:

$$\sum_{(k,\ell)\in N\times N} f_{ij}^{k\ell} = \hat{f}_{ij} \leq 0 + K_{ij}y_{ij} \qquad (i,j)\in A$$
$$y_{ij} = 0 \text{ or } 1$$

where y_{ij} is a 0-1 variable indicating the capacity level of arc (i,j).

Our general framework includes two types of costs. The first kind, denoted by $RC_{ij}(\hat{f}_{ij})$, is the routing cost for arc (i,j) associated with satisfying the required flow constraints (2.1). In various applications the routing costs may correspond to travel time, risk of accidents or any other "costs" which vary with the amount of traffic on the arc.

The second type of cost, the construction cost for the capacity improvement of arc (i,j), is denoted by $CC_{ij}(\ell_{ij})$ and includes capital construction costs, maintenance fees, and any other costs that depend solely on the arc capacity level.

In general, our objective will be to minimize the total routing and construction costs. With the above information, we can state our general network design problem as:

$$\text{Minimize} \quad \sum_{(i,j)\in A} RC_{ij}(\hat{f}_{ij}) + CC_{ij}(\ell_{ij})$$

subject to: (2.1), (2.2) and any special problem constraints

$$f_{ij}^{k\ell} \geq 0 \qquad (i,j)\in A \qquad (k,\ell)\in N\times N$$

We will further classify our network design problems according to their routing cost functions $RC_{ij}(\hat{f}_{ij})$. If, for a particular network design, all routing cost function functions are linear and every arc capacity is either zero or infinite (we usually represent an "infinite" arc capacity value as some sufficiently large number such as the total amount of required flow in the problem), then we will refer to the model as a network design problem without congestion costs. This terminology is chosen to reflect the fact that if an arc is present in the network (i.e. has non-zero capacity, then any amount of flow can be routed through it and the marginal cost for routing an additional unit of flow is always constant, independent of flow conditions in the network.

If a network design model has convex routing cost functions and/or some finite non-zero arc capacities, then we will refer to it as a network design problem with congestion costs. These congestion costs are reflected in the convex routing cost functions (i.e. increasing marginal costs) and/or the prohibition of additional flow through an arc after a certain

limit has been reached.

In the following sections we describe and analyze a number of different problems in terms of this basic network design model.

2.3 Network Design Problems without Congestion Costs

Network design problems without congestion costs often model underutilized systems such as a communications network where the amount of information transmitted is always below the capacity of a standard trunkline. In this case, the arc capacity is effectively infinite. Another use for this type of system is to gain insight into more complicated networks with congestion costs by studying this simpler network model. Also network design problems without congestion costs can be used as subproblems in a procedure for solving more complicated network design models.

The first network problem that we consider was formulated by Billheimer and Gray [14]. Initially all arcs have zero capacity. The arc routing costs are linear functions of the total arc flow. The construction cost required to build an arc with "infinite" capacity is a fixed charge. The objective is to minimize the sum of routing and construction costs. (We will refer to this network design model as the "fixed charge design problem.")

Since all arc capacities can only take on discrete values (either zero or "infinity"), we can formulate the fixed charge design problem as the following mixed integer program:

$$\text{Minimize} \quad : \quad \sum_{(i,j)\in A} d_{ij}\,\hat{f}_{ij} + c_{ij}\,y_{ij}$$

subject to: (2.1)

$$f_{ij}^{k\ell} \leq R_{k\ell} \cdot y_{ij} \qquad (i,j)\in A$$

$$f_{ij}^{k\ell} \geq 0 \qquad (k,\ell)\in N\times N$$

$$y_{ij} = 0 \text{ or } 1.$$

y_{ij} indicates whether or not arc (i,j) is present in the network. d_{ij} is the cost of routing a unit of flow through arc (i,j). c_{ij} is the cost of adding arc (i,j) to the network (i.e. setting its capacity to "infinity"). Note that the second constraint is a specialization of (2.2) to the fixed charge design model.

Note that the above network design formulation gives rise to large mixed integer programs. For example, a network design with 50 nodes and 200 possible directed arcs will be formulated with 12500 rows, 10000 continuous variables and 200 binary variables.

Magnanti and Wong [80] (also see chapter 3 of this thesis) applied Benders decomposition to the above formulation of the fixed charge design problem. They specify a technique for accelerating the convergence of Benders procedure. Their computational experience includes satisfactorily solving networks with 10 nodes and 45 arcs in about 60 seconds of IBM 370/168 computer time.

Since this problem is very complex, Billheimer and Gray propose a heuristic solution procedure. Each iteration of this procedure consists

of either deleting or adding an arc to the network so that the total cost (routing and construction) is reduced. The iterations are continued until a local optimum is reached where now further addition or deletion of a single arc reduces the cost of the network configuration.

The heuristic procedure has been tested on a problem with 68 nodes and 476 arcs. The method reached a local optimum after about 3 minutes of computation time on an IBM 360/67 computer. It is difficult to judge the quality of the heuristic's solution since no satisfactory method is known for optimally solving problems of that size.

It is interesting to see the wide range of network models that are related to the fixed charge design problem. Many combinatorial network problems are special cases of it. If all arc construction costs are set to zero, then the fixed charge design model becomes a series of shortest path problems. If all arc routing costs are set to zero, the fixed charge design model becomes a Steiner tree problem on a graph (Steiner's problem) [28, 56]. The Steiner problem occurs because the required flows will necessitate that there be a path between every pair of nodes in some subset of the nodes in the network.

Since the fixed charge design problem contains the Steiner problem as a special case, we can be confident that it is very difficult to solve. Karp [65] has shown that the Steiner three problem on a graph is NP-complete. This implies that the Steiner problem is as difficult to solve as such combinatorial problems as the traveling salesman problem [10], the maximum clique problem [57] and the 0-1 integer programming problem (see [65,66] for a full discussion of the various NP-complete problems). In view of the lack of success in solving any of these problems on a large

scale, it appears unlikely that there is an efficient algorithm for the Steiner problem or for the fixed charge design problem. In fact, the fixed charge design problem itself is NP-complete. (This result follows from the fact that the Steiner problem is a special case of fixed charge network design).

If the arc construction costs are all equal and totally dominate the routing costs (i.e., the optimal network design must be a tree), then the fixed charge design problem becomes the optimum communication spanning tree problem defined by Hu [61].

Another special case of Bilheimer and Gray's problem is the fixed charge plant location problem [30,36]. The plant location problem is normally associated with the placement of facilities on the nodes of a graph. The objective is to minimize the sum of the fixed charges for locating the various plants and the routing costs for servicing customers from the constructed plants. However, it is possible to convert the plant location problem to a network synthesis problem. This can be done in the following way: add a special node to the plant location network. This node will be the source of all the flow required by the customer nodes. Also, add a set of special arcs leading from the special node to each potential plant site (see figure 2.1). A special arc connecting the special node to a plant site has a construction cost equal to the fixed charge associated with opening the site. These special arcs will have no routing costs. Arcs connecting plant locations with customers have no construction costs. However, they will have a routing cost equal to the transportation cost from the plant location to the customer. So now the corresponding synthesis problem is to design the minimum total cost

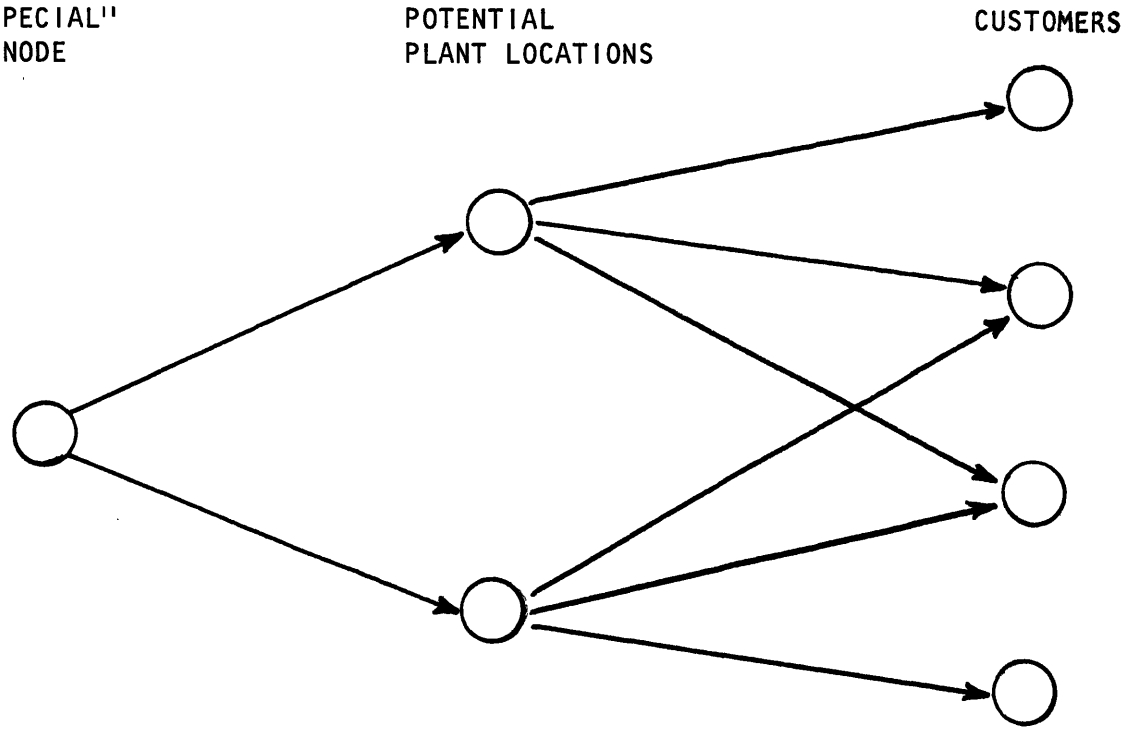


Figure 2.1 Plant Location as an Arc Design Problem

(construction plus routing cost) network so that all the flow requirements between the special node and the customers are satisfied. Thus, the fixed charge plant location problem is a special case of the fixed charge design problem.

Viewing the fixed charge plant location problem as a special case of the fixed charge design model gives us additional insight into the network synthesis problem. For instance, Billheimer and Gray describe some methods for partially characterizing the optimal network configuration. These techniques can be shown to be generalizations of procedures given by Efoymsen and Ray [30] for characterizing the optimal set of sites in the plant location problem.

By using a similar transformation we can show that many other different facility location problems are special cases of various network design problems. For example, if we have a capacitated plant location problem, the node capacity constraint can be represented by a capacity constraint on one of the "special" arcs added to the network. Since there has been so much work done in the area of facility location problems (see [21, 36]), it may be possible to generalize some of these other techniques in order to apply them to network design problems. The rules given by Billheimer and Gray and Efroymson and Ray are one example of such a generalization.

Scott [114,115] has introduced another network synthesis problem, called the "optimal network" problem, that is closely related to the fixed charge design problem. The arc routing costs in this problem are all linear functions of the total flow. Arc capacities, which are all initially zero, can be raised to infinity. The objective is to minimize total routing cost subject to the usual capacity and flow routing constraints and the added constraint that the total construction costs cannot exceed a given budget.

The optimal network problem can be formulated as the following mixed integer program:

$$\text{Minimize} \quad \sum_{(i,j)\in A} d_{ij}\, \hat{f}_{ij}$$

subject to: (2.1)

$$f_{ij}^{k\ell} \leq R_{k\ell}\, y_{ij} \qquad (i,j)\in A$$

$$\sum_{(i,j)\in A} c_{ij}\, y_{ij} \leq \text{BUDGET} \qquad (k,\ell)\in N\times N$$

$$f_{ij}^{k\ell} \geq 0$$

$$y_{ij} = 0 \text{ or } 1.$$

All variables and constants have the same interpretation as in the formulation of the fixed charge design problem.

Many researchers have considered this problem since its solution could be useful to the design of various transportation (highway, rail or air) systems. As noted by Dionne and Florian [27], since these systems usually have many more operating constraints, "the justification for studying this problem is that its solution may be used as a measuring standard for the efficiency of proposed designs."

Boyce et al. [15] utilized a branch and bound algorithm to solve the optimal network problem. They were able to solve problems with 10 nodes and 45 arcs in 3 to 400 seconds of IBM 360/75 computer time depending on the value of the construction budget. Hoang [59] presented another branch and bound procedure that has been modified and improved by Dionne and Florian [27]. Their procedure produced computation times that were comparable to the Boyce et al. results. Dionne [26] has shown that the computation time of the Dionne and Florian procedure increases exponentially with decreasing construction budget. It is believed that the algorithm of Boyce et al. should behave in a similar manner. Geoffrion [48] has presented another branch and bound procedure that is based on

Lagrangian relaxation techniques.

In order to address large-scale optimal network problems, several researchers have suggested using heuristic procedures. Scott [115] and Dionne and Florian [27] proposed heuristic algorithms that are closely related to the Billheimer and Gray procedure for the fixed charge design problem. Dionne and Florian have solved test problems containing up to 29 nodes and 54 arcs. The computational results were very promising with the average error relative to the optimal solution less than 1%. Computation times ranged from .1 to 12 seconds on the CDC Cyber 74 computer. However, our analyses in Chapter 5 indicate the maximum error for such heuristics could be very large. Further computational tests should be performed in order to resolved this issue.

It is unlikely that there exists an efficient optimal algorithm for the optimal network problem since Lenstra, Rinooy Kan and Johnson [63] have shown that the optimal network problem is NP-complete.

Another group of network design problems without congestion costs concerns network improvement where we start with an initial feasible network and then attach additional arcs. As in the case of the optimal network problem, there are the usual capacity and flow routing constraints and also a construction budget for the added arcs.

Ridley [106] suggested a network based branch and bound approach for these problems. Stairs [116] indicated that Ridley's method has been used to solve problems containing up to 12 nodes. Goldman and Nemhauser [53] consider a special case of the network improvement problem where the objective is to improve the shortest path between a single pair of nodes. They show how to transform the problem into a shortest route problem on an

expanded network. Wollmer [126] and Ridley [105] give efficient procedures for solving special cases of the shortest path improvement problem. However, these techniques are just special cases of Goldman and Nemhauser's procedure.

Stairs [116] presented a network improvement problem that is related to Billheimer and Gray's network synthesis problem. She described an interactive computer solution procedure which has been successfully applied to a test problem containing 35 nodes and 10 possible arcs that could be improved.

Note that our network improvement models are all special cases of the network synthesis models presented earlier. So it should be possible to adapt the previously described network synthesis techniques to network improvement applications.

2.4 Network Design Problems with Congestion Costs

A more complex type of network design problem incorporates congestion costs for the routing of the network flows. These congestion costs can be represented by i) convex flow routing costs that could reflect such effects as highway traffic congestion or communication network queuing delays; ii) finite arc capacities that could represent physical, environmental or political limits on the total traffic that can pass through an arc.

Some of the models described here have been used to help design traffic network, rail network and communication network systems. All of the models that we will discuss are network improvement problems. Unless

specified otherwise, we assume that the initial arc capacities constitute a feasible network design solution.

The first type of network improvement problem that we consider is similar to the uncongested design problems of the previous section. In addition to the usual capacity and flow routing constraints that must be satisfied, we must select the arc capacities from a discrete set of values. Thus, the problem is essentially a combinatorial one as was the case for the uncongested design problems. Roberts and Funk [107], Carter and Stowers [19] and Hershdorfer [58] described work in this area. Hershdorfer utilized a branch and bound procedure with networks containing up to 12 nodes.

Agarwal [1] considered a different kind of network improvement problem where the possbile capacity of an arc (i,j) ranges continuously between zero and some upper bound K_{ij}. Construction costs are linear functions of the arc capacity increase. Routing costs are convex piecewise linear functions of the flow. The objective is to minimize the total routing cost subject to all the usual constraints and a construction budget constraint.

Agarwal conducted computational tests on a network with 24 nodes and 38 arcs that was formulated as a linear program with 667 rows and 1938 variables. The results were quite discouraging since the simplex method, Dantzig-Wolfe decomposition and the Boxstep method [83] all failed to solve the problem in a reasonable amount of time. Agarwal concluded that none of the methods was effective because of the arc capacity upper bounds present in the problem.

The difficulty caused by the capacity constraints should not be surprising. Note that the problem of computing the routing cost for a particular proposed network solution requires the solution of a difficult capacitated multi-commodity flow problem [3, 68]. Since the problem of evaluating a proposed solution is so difficult, it should be expected that the problem of finding the optimal network improvement solution is also very difficult. Next we review several models that are similar to Agarwal's problem and discuss some approaches for dealing with the difficulty of the embedded routing problem.

Steenbrink [117,118] used a model similar to Agarwal's for the design of a Dutch roadway network. The capacity of an arc (i,j) is restricted to be between zero and K_{ij}. Routing costs are convex but the construction costs are nonlinear. The objective is to minimize the total routing and construction costs subject to all the usual flow routing constraints.

Steenbrink formulated his model as an optimization problem with linear constraints and a non-linear objective function. He suggests decomposing the problem into a master problem and a series of subproblems. Each subproblem concerns finding the optimal capacity for an arc given the total flow through it. The master problem is to route the required flows through the network with a modified flow cost structure. (This master problem is again a capacitated multi-commodity flow problem). Steenbrink's heuristic procedure for solving the master problem, as was noted by Nguyen [95], is closely related to the well-known incremental loading traffic assignment procedure [84]. So Steenbrink's technique for dealing with the embedded routing problem is to solve it heuristically.

Steenbrink applied this method to a Dutch roadway design problem containing 2000 nodes and 6000 arcs. The heuristic procedure required about 50 minutes of IBM 360/65 computer time. Due to the size of the problem, there is no way to evaluate the quality of Steenbrink's solution.

Dantzig et al. [22] consider a network improvement problem identical to Agarwal's except for a crucial assumption that there is no upper limit on an arc capacity. (Note that congestion costs are still present due to the convex routing costs). They dualize with respect to the budget constraint and then use Steenbrink's decomposition. The master problem is a convex cost multi-commodity flow problem which can be solved very efficiently using the Frank-Wolfe algorithm. The procedure required 10.68 seconds of IBM 370/168 computer time on a test problem with 24 nodes and 76 arcs and produced a solution 2.5% away from optimality. In contrast, the simplex method, implemented on the MPS/360 package, required 40.8 minutes to obtain an optimal solution. The authors also report experience on a problem with 394 nodes and 1042 arcs which required 5.63 minutes of computer time.

Note that the use of a convex routing cost function to "represent" a finite flow capacity constraint greatly improved the computational performance for this type of congested network improvement problem. So, slightly altering the modelling of congestion avoids a difficult embedded routing problem.

McCallum [85] described a capacitied network planning problem concerning the location of circuits in a communication (telephone) network. This capacitated network is similar to Agarwal's except that between every pair of nodes only a few paths are allowable as flow routes. Thus, the

difficult embedded routing problem is avoided. After formulating the model as a linear program, McCallum used a specialized implementation of the generalized upper bounding technique to solve problems containing up to 563 arcs and 1857 required flows between pairs of nodes. The computation time required for a problem of this size was 173 seconds on the IBM 370/165 computer.

2.5 Network Design Problems with User Equilibrium Routing

In the network design problems with congestion costs discussed in the previous sections, all flows were routed according to a "system optimal" policy which minimized the total routing cost of all flows. In this section we consider problems where the flows are routed according to Wardrop's "Principle of Equal Travel Times" [122]. That is, the traffic is assigned so that the path or paths actually used between each origin and destination will have the smallest travel costs. (Under certain circumstances [9,78] the user equilibrium routing problem can be transformed into a system optimal routing problem). The user equilibrium routing (UER) policy has been demonstrated to be a useful method of modeling behavior in transportation systems [35].

We begin by describing a major difference between network design problems (with congestion costs) that have UER and those with system optimal routing. For a network with system optimal routing, the addition of an arc to the network never increases the total flow routing costs. Since we can always choose to use the previously determined flow routing pattern, the total routing cost can never increase and will usually decrease.

Somewhat surprisingly, for a network with UER, the addition of an arc can lead to an increase in the total flow routing costs. This phenomenon, known as Braess' paradox [16,90], indicates that great care should be used in evaluating proposed improvements to a network with UER. Knödel [69,90], described an actual situation in an urban street system where such a phenomenon occurred.

Leblanc [71] considered a network design with UER where all arcs can have either zero or infinite capacity. The objective is to minimize total routing costs subject to all the usual constraints and a construction budget. The branch and bound solution procedure proposed for this model has solved a network problem containing 24 nodes, 76 arcs and 5 arcs that could be added to the network. Computation time was about 136 seconds on the CDC 6400 computer.

Morlok and LeBlanc [89] address the same network design problem but with a heuristic procedure. The technique is based on marginal analysis of the traffic flows. The heuristic procedure essentially solved the same 24 node problem in 17.8 seconds of Cyber 70 computer time.

Ochoa and Silva [98] and Chan [20] also discuss similar types of network improvement problems.

Barbier [7,116] considered a problem similar to LeBlanc's except that the objective is to minimize the total routing and construction costs without a budget constraint. His heuristic procedure for obtaining proposed solutions has been used to study additions to the Paris rail network. Computational experience includes analyzing a network with 36 nodes, over 30 arcs and over 50 candidate arcs. Steenbrink [118] reported that Haubrich used a revised version of Barbier's method to study the Dutch

rail network. Haubrich's procedure solved a design network problem with about 1250 nodes and about 8000 arcs in less than 40 minutes of IBM 360/75 computer time.

2.6 Conclusion

In this chapter we have reviewed a large number of network design problems and their proposed solution techniques. Table 2.1 summarizes this information.

There still remains a great deal of work to be done on network design problems. Most of the network design problems without congestion costs that we have considered are known to be difficult (NP-complete) combinatorial optimization problems. All of the known exact solution techniques are limited to small and medium sized networks. In order for these models to be useful in applications such as transportation planning, large-scale problems will have to be solved. Branch and bound methods appear inadequate for this task.

Recent work by several authors (see chapter III for a summary of this research) has shown that Benders decomposition could be a useful tool.

Another promising approach is to use heuristic algorithms as appro-approximate solution techniques. Further work is required in evaluating the accuracy and reliability of these procedures. For example, see [62, 39, 67] and chapter 5 of this thesis for work in analyzing heuristics for various network optimization problems.

Also, recent advances in large scale system methodology, such as list processing techniques and network flow algorithms, may have some impact on the size of problems that can be solved practically. The reader

TABLE 2.1

AUTHORS	ARC CAPACITY VARIABLES/ CONGESTED OR UNCONGESTED PROBLEM	SOLUTION ALGORITHM	(# of nodes in test network, # of arcs, Comp. time, Machine) COMPUTATIONAL EXPERIENCE
1. Billheimer and Gray [14]	Discrete/ uncongested	Heuristic	(68 nodes, 476 arcs, 180 seconds, IBM 360/67)
2. Magnanti and Wong [80]	Discrete/ uncongested	Benders Decomposition	(10 nodes, 45 arcs, 60 seconds, IBM 370/168)
3. Boyce et al [15]	Discrete/ uncongested	Branch and Bound	(10 nodes, 45 arcs, 200 seconds, IBM 360/75)
4. Hoang [59]	Discrete/ uncongested	Branch and Bound	(8 nodes, 20 arcs, ?, ?)
5. Dionne and Florian [27]	Discrete/ uncongested	1) Branch and Bound 2) Heuristic	(29 nodes, 54 arcs, 12 seconds, Cyber 74)
6. Scott [115]	Discrete/ uncongested	Heuristic	(10 nodes, 45 arcs, 60 seconds, IBM 360/65)
7. Ridley [106]	Discrete/ uncongested	Branch and Bound	(12 nodes, ?,?,?)
8. Stairs [116]	Discrete/ uncongested	Interactive Computer System	(35 nodes, ?,?,?)
9. Agarwal [1]	Continuous/ congested	1) Simplex Method 2) Dantzig-Wolfe Decomposition 3) Boxstep	(24 nodes, 38 arcs, 840 seconds, CDC 6400) for simplex method
10. Steenbrink [117,118]	Continuous/ congested	Special decomposition with a heuristic	(2000 nodes, 6000 arcs, 2880 seconds, IBM 360/65)

TABLE 2.1 (continued)

AUTHORS	ARC CAPACITY VARIABLES/ CONGESTED OR UNCONGESTED PROBLEM	SOLUTION ALGORITHM	(# of nodes in test network, # of arcs, Comp. time, Machine) COMPUTATIONAL EXPERIENCE
11. Dantzig et al [23]	Continuous/ congested	Special decomposition with Frank-Wolfe decomposition	(394 nodes, 1042 arcs, 340 seconds, IBM 370/168)
12. McCallum [85]	Continuous/ congested	Generalized Upper Bounding	(?, 563 arcs, 173 seconds, IBM 370/165)
13. LeBlanc [71]	Discrete/ congested	Branch and Bound	(24 nodes, 76 arcs, 135 seconds, CDC 6400)
14. Morlok and LeBlanc [89]	Discrete/ congested	Heuristic	(24 nodes, 76 arcs, 18 seconds, Cyber 70)
15. Barbier [7,116]	Discrete/ congested	Heuristic	(36 nodes, 8 arcs, ?,?)
16. Haubrich [118]	Discrete/ congested	Heuristic	(1250 nodes, 8000 arcs, 2400 seconds, IBM 360/65)

may consult a recent report by Magnanti [77] for a survey of these new advances.

The network design problem with congestion costs and discrete arc capacities, is even more difficult than the uncongested case and appears to be a formidable problem. For a network with congestion costs and continuous arc capacities, there have been some successful efforts. Although the embedded multi-commodity routing problem poses difficulties for some versions of this problem, Dantzig et al. and McCallum have successfully avoided this obstacle. Utilizing special problem structures in formulating their mathematical programs, they were able to apply linear and convex programming techniques to solve problems whose size is of practical interest. It would be interesting to see if these techniques could be used to solve other versions of network design models with congestion costs.

Thers are also other kinds of basic network design models that could be explored in future research. For example, Yaged [128] and Zadeh [130] considered network design problems with concave objective functions. Soukoup [119], Newell [93], Bansal and Jacobsen [6], and Rothfarb and Goldstein [131] have also explored various other network design models.

Another promising area for future research is to extend these network design problems to more dynamic situations. The basic models considered here are all static in that the network is optimized for a single time period with all changes to the network made instantaneously.

There are several types of time-varying elements that could be incorporated into network design problems. One kind of model of this

nature involves networks where the required flows between nodes can be time-varying. For example, in an urban transportation system or a communications network, the traffic demands could vary greatly according to the time of day or season of the year. Gomory and Hu [54] and Oettli and Prager [99] have investigated this kind of network problem.

Another type of time varying design problem concerns network improvements that must be sequenced over a number of time periods. In most real situations the network can only change gradually over a given time span. Ochoa-Rosso [97], Funk and Tillman [38] and Yaged [129] have considered this type of problem.

The third type of time-dependence is related to the previous two and concerns the changes in traffic demands when the network is modified. For example, the evolution of a transportation network will influence the development of the surrounding geographic region. Therefore, future traffic demands by region will be dependent on changes to the transportation network in previous time periods. See Frey and Nemhauser [37] and Los [75] for examples of this type of problem. Also MacKinnon [76] discusses these last two types of time-dependent problems in his survey.

CHAPTER III

AN ACCELERATION TECHNIQUE FOR BENDERS DECOMPOSITION

3.1 Introduction

As we noted in Chapters I and II, the break bulk problem and other network designs problems can be formulated as mixed integer programming problems. Motivated by recent successful applications of Benders decomposition to mixed integer programs† by Florian et al [33], by Richardson [104], and particularly by Geoffrion and Graves [49], who study industrial products distribution, we viewed this algorithm as potentially useful for our applications. Our early computational experience, however, indicated that straightforward adoption of Benders algorithm converged too slowly and required the solution of far too many integer programming problems. This led us to consider mechanisms for improving the algorithm. Our intention was to reduce the number of integer programs to be solved. In this chapter, we report on the results of this study.

Rather than cast our development solely in terms of the network design problems, we consider a broader minimax setting that includes Benders Decomposition for general mixed integer programs, Dantzig-Wolfe Decomposition for linear and nonlinear programs, and related "cutting plane" type algorithms that arise in resource directive and price

---

†See Florian and Nguyen [35], Noonan and Giglio [96], and Armstrong and Willis [2] for successful application to nonlinear programs.

directive decomposition. This will allow us to explore the full generality our techniques. In the next section we review the essential properties of these cutting plane, or relaxation, algorithms in this minimax setting.

In section 3.3, we describe an acceleration technique for reducing the number of iterations of the relaxation algorithm. We accomplish this by choosing judiciously from the possible cuts that could be generated at any iteration to obtain "strong" or "pareto-optimal" cuts. In Benders Decomposition this selection process involves a choice, made by solving a linear program, from the multiple optimal solutions of another linear program.

In the next section, we specialize this general methodology to facility location and network design problems, developing very efficient algorithms that exploit the underlying structure of these models. Since the linear program for generating cuts in these applications is the dual of a network optimization problem, multiple optimal solutions will be commonplace, thus providing an excellent opportunity for applying out proposed methodology.

Section 3.5 describes our computational experience with several p-median location and network design problems. Our results on p-median problems (up to 33 nodes) show that Benders algorithm equipped with our methodology finds solutions known to be within 10 per cent of optimality in ten or fewer iterations. The standard implementation usually provides no better solutions within twenty-five iterations and solutions 10 percent farther from optimality within ten iterations. We obtained similar comparisons for network design problems, though in this case the error bounds are generally not as tight.

The final section describes possibilities for using the strong cut methodology in conjunction with other techniques for solving mixed integer programs. It also points out possibilities for further investigation.

3.2 Benders Decomposition and Minimax Optimization

3.2.1 Minimax Problems

Two of the most widely-used strategies for solving large scale optimization problems are resource directive decomposition and Lagrangian relaxation. Several papers in the mathematical programming literature (see, for example, Geoffrion [43] and [44], and Magnanti [77]) point out the central importance and unifying nature of these solution techniques. The techniques are not only applied directly; their use is, at times, combined with other approaches as when Lagrangian relaxation is embedded within the framework of branch and bound for solving integer programming problems (Fisher and Shapiro [32], Geoffrion [46]).

Since Benders algorithm, the focus of our analysis, is but one manifestation of resource directive decomposition, we shall consider a broader, but somewhat more abstract, minimax setting that captures the essence of both the resource directive and Lagrangian relaxation approaches. We study the optimization problem

$$v = \underset{y\in Y}{\text{Min}}\ \underset{u\in U}{\text{Max}}\{f(u) + yg(u)\} \qquad (3.1)$$

where Y and U are given subsets of R^k and R^m, f is a real valued function

defined on U and g(u) is an m-dimensional vector for any $u \varepsilon U$. Note that we are restricting the objective function f(u) + yg(u) to be linear-affine in the outer minimizing variable y for each choice of the inner maximizing variable u.

The relation between Benders decomposition and the minimax problem can be seen by considering the general mixed integer program:

$$\begin{aligned} &\text{Minimize} && cx + dy \\ &\text{subject to:} && Ax + Dy = b \\ & && x \geq 0,\ y \varepsilon Y. \end{aligned}$$

In this formulation x is an n-vector of continuous variables, y is a k-vector of discrete variables, and Y is a subset of the integer points in k-dimensions. The matrices A and D and vectors c, d and b have dimensions compatible with those of x and y.

We can reformulate this program in the equivalent form:

$$\begin{aligned} &\underset{y \varepsilon Y}{\text{Minimize}} && \text{Minimize}\{cx + dy\}. \\ & && x \geq 0 \\ & && Ax = b - Dy \end{aligned} \tag{3.2}$$

For any fixed value of y, the inner minimization is a linear program. If it is feasible and has an optimal solution for all $y \varepsilon Y$[†], then dualizing gives the equivalent formulation

---

[†]These assumptions can be relaxed quite easily, but with added complications that cloud our main development. See Garfinkel and Nemhauser [42] or Lasdon [70] for a review of the algorithm in full generality.

$$\underset{y\in Y}{\text{Minimize}} \quad \underset{u\in U}{\text{Maximize}}\{ub - uDy + dy\}$$

which is a special case of (3.1) in which $U = \{u\in R^m: uA \leq c\}$, $f(u) = ub$ and $g(u) = d - uD$. This reformulation is typical of the resource directive philosophy of solving parametrically in terms of complicating variables, like the integer variables y of a mixed integer program.

The minimax problem (3.1) also arises when dualizing the constraints $g(u) \geq 0$ of the optimization problem

$$\begin{aligned} &\text{Maximize} && f(u) \\ &\text{subject to:} && g(u) \geq 0 \\ & && u\in U. \end{aligned} \tag{3.3}$$

The resulting optimization problem is the Lagrangian dual, a form of the minimax problem in which Y is the nonnegative orthant, or more generally the convex subset of the nonnegative orthant for which the maximization problem over U is finite valued.

3.2.2 Solving Minimax Problems by Relaxation

For any given $y\in Y$, let $v(y)$ denote the value of the maximization problem in (3.1); that is,

$$v = \underset{y\in Y}{\text{Min}}\, v(y)$$

where

$$v(y) = \underset{u\in U}{\text{Max}}\{f(u) + yg(u)\} \tag{3.4}$$

Since $v(y)$ is defined as the pointwise maximum of linear-affine functions, it is convex, though generally nondifferentiable. Consequently, whenever the set Y is convex, the minimax problem can be viewed as a convex program. There has been a great flourish of activity recently in modifying and extending algorithms of differentiable optimization to solve this class of problems (see Dem'yanov and Malazemov [25], Lemarechal [73], Mifflin [88], Wolfe [125] and the references that they cite). An alternative solution strategy that applies even when Y is not convex is a relaxation approach. Rewrite (3.1) as

$$\begin{aligned} &\text{Minimize} && z \\ &\text{subject to:} && z \geq f(u) + yg(u) \quad \text{for all } u \in U \\ & && y \in Y,\ z \in R \end{aligned} \tag{3.5}$$

and form a relaxation

$$\begin{aligned} &\text{Minimize} && z \\ &\text{subject to:} && z \geq f\left(u^j\right) + yg\left(u^j\right) \quad (j = 1,2,\ldots, K) \\ & && y \in Y,\ z \in R \end{aligned} \tag{3.6}$$

where each u^j is an element of U. The solution y^K, z^K of this "master problem" (3.6) is optimal in (3.5) if it satisfies all of the constraints of that problem; that is, if $v\left(y^K\right) \leq z^K$. If, on the other hand, $v\left(y^K\right) > z^K$ and u^{K+1} solves[†] the "subproblem" (3.4) when $y = y^K$, then we add

$$z \geq f\left(u^{K+1}\right) + yg\left(u^{K+1}\right)$$

[†]As before, to simplify our discussion we assume that this problem always has at least one optimal solution.

as a new constraint, or cut as it is usually called, to the master problem (3.6). The algorithm continues in this way, alternately solving the master problem and subproblem.

When applied to problems (3.2) and (3.3), this algorithm is known, respectively, as Benders Decomposition and Dantzig-Wolfe Decomposition or generalized programming. The master problem is an integer program with one continuous variable when Benders algorithm is applied to mixed integer programs; it is a linear program when Dantzig-Wolfe decomposition is applied to nonlinear programs. The convergence properties of the relaxation algorithm are well-known, although usually stated in the context of particular instances of the algorithm, (see, for example, Benders [11], Dantzig [22] and Magnanti et. al. [79]). If the subproblem is a linear program then the point u^j in (3.6) can be chosen as extreme points of U and the algorithm terminates after a finite number of iterations. If the set U is compact and the functions f and g are continuous, then any limit point $y^* \in Y$, if one exists, to the sequence $\{y^K\}_{K \geq 1}$ is optimal in (3.1). Neither of these convergence properties depends upon structural properties of Y. Nevertheless, the structure of Y does determine whether or not the master problem (3.6) can be solved efficiently.

3.3 Accelerating the Relaxation Algorithm

A major computational bottleneck in applying Benders Decomposition is that the master problem, which must be solved repeatedly, is an integer program. Even when the master problem is linear program as in the application of Dantzig-Wolfe Decomposition, the relaxation algorithm has not

generally performed well due to its poor convergence properties (Orchard-Hays [101], Wolfe [124]). There are several possibilities for improvement:

(i) making a good selection of initial cuts, i.e., values of the u^j, for the master problem;

(ii) modifying the master problem to alter the choice of y^K at each step;

(iii) formulating the problem "properly"; and

(iv) if there are choices, selecting good cuts to add to the master problem at each step.

In a number of studies of mixed integer programs, Mevert [87] found that the initial selection of cuts can have a profound effect upon the performance of Benders algorithm. Geoffrion and Graves [49] have reported similar experience with facility location problems.

There have been several proposals to alter the master problem for Dantzig-Wolfe Decomposition. Nemhauser and Widhelm [91] (see also O'Neill and Widhelm [100]) show that scaling the constraints of the master problem to find the "geometrically centered" value of y^K at each step, can be beneficial. Marsten, Hogan and Blankenship [83], see also Marsten [82], have had success in restricting the solution to the master problem at each step to lie within a box centered about the previous solution. Holloway [60] shows how to select among multiple optima of the master problem to obtain better convergence.

Model formulation is an important topic which can greatly effect the computational efficiency of Benders decomposition. In chapter IV we

consider the question of problem formulation in some detail.

In many instances, as when Benders decomposition is applied to network optimization problems, the selection of good cuts at each iteration becomes an issue. In network applications, multiple optimal solutions to the subproblem (3.4) are the norm; equivalently, degenerate solutions to its dual problem

$$\text{Minimize}\{dy + cx : Ax = b - Dy,\ x \geq 0\}$$

are to be expected because the shortest route, transhipment and other network optimization problems are reknowned for their degeneracy. In the remainder of this paper, we introduce methods and algorithms for choosing from the alternative optima to (3.4) at each iteration, a solution that generates a cut that is in some sense "best."

Example 3.1

To illustrate the possibility of selecting good cuts to add to the master problem, we consider the following simple example:

$$\begin{aligned} \text{Minimize} \quad & x_3 + y \\ \text{subject to:} \quad & -x_1 \qquad x_3 + 2y = 4 \\ & -x_2 + x_3 + 5y = 4 \\ & x_1 \geq 0,\ x_2 \geq 0,\ x_3 \geq 0 \\ & y \geq 0 \text{ and integer.} \end{aligned}$$

The equivalent formulation (3.5) written in terms of the linear programming dual obtained when y is fixed is:

$$
\begin{aligned}
&\text{Minimize} && z \\
&\text{subject to:} && z \geq y \\
& && z \geq 4-y \qquad (3.7) \\
& && z \geq 4-4y \\
& && y \geq 0 \text{ and integer.}
\end{aligned}
$$

The constraints correspond to the three extreme points $u^1 = (0,0)$, $u^2 = (1,0)$ and $u^3 = (0,1)$ of the dual feasible region U.

Suppose that we initiate the relaxation algorithm with the single cut $z \geq y$ in the master problem. The optimal solution is $z^1 = y^1 = 0$. As $\hat{y} = y^1 = 0$, both the extreme points u^2 and u^3 (and every convex combination of them) solves the subproblem:

$$
\begin{aligned}
&\text{Minimize} && (4-2y)u_1 + (4-5y)u_2 + y \\
&\text{subject to:} && (u_1, u_2) \in U.
\end{aligned}
$$

Stated in another way, both the second and third constraints of (8) are most violated at $z = y = 0$.

Adding the second constraint gives the optimal solution $z^2 = y^2 = 2$ to the original problem as the next solution to the master problem. Adding the third constraint gives the nonoptimal solution $z^2 = y^2 = 1$ and requires another iteration that adds the remaining constraint of (3.7).

In this instance, the second constraint of (3.7) dominates the third in the sense that

$$4-y \geq 4-4y$$

whenever $y \geq 0$ with strict equality if $y > 0$. That is, the second constraint provides a sharper lower bound on z.

To identify the dominant cut in this case, we check to see which of the second or third constraints of (3.7) has the largest righthand side value for <u>any</u> $y^0 > 0$. In terms of the subproblem this criteria becomes: from among the alternate optimal solutions to the subproblem at $\hat{y} = 0$ choose a solution that maximizes the subproblem's objective function when $\hat{y} = y^0 > 0$.

Before extending this observation to arbitrary minimax problems, we formalize some definitions.

We say that the cut (or constraint)

$$z \geq f(u^1) + yg(u^1)$$

in the minimax problem (2) <u>dominates</u> or is <u>stronger</u> than the cut

$$z \geq f(u) + yg(u)$$

if

$$f(u^1) + yg(u^1) \geq f(u) + yg(u)$$

for all $y \in Y$ with a strict inequality for at least one point $y \in Y$. We call a cut <u>pareto optimal</u> 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 previous example, 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 $Y = \{y : y \geq 0 \text{ and integer}\}$. The following theorem shows that this observation generalizes to any minimax

problem of the form (3.1). 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 Y^c is always nonempty. For notation, let us call any point y^o contained in the relative interior of Y^c, a core point of Y.

Theorem 3.1: 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

$$\underset{u \in U}{\text{Max}} \{f(u) + \hat{y}g(u)\} \tag{3.8}$$

and let u^o solve the problem:

$$\underset{u \in U(\hat{y})}{\text{Max}} \{f(u) + y^o g(u)\}. \tag{3.9}$$

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

$$f(\bar{u}) + yg(\bar{u}) \geq f(u^o) + yg(u^o) \quad \text{for all } y \in Y \tag{3.10}$$

it is true that

$$f(\bar{u}) + wg(\bar{u}) \geq f(u^o) + wg(u^o) \quad \text{for all } w \in Y^c. \tag{3.11}$$

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

$$w = \sum\{\lambda_y y : y \in Y\}$$

where $\lambda_y \geq 0$ 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 (11) with $y = \hat{y}$, that $\bar{u}$ must be an optimal solution to the optimization problem (3.8), that is, $\bar{u} \in U(\hat{y})$. But then (3.10) and (3.9) imply that

$$f(\bar{u}) + y^o g(\bar{u}) = f(u^o) + y^o g(u^o). \tag{3.12}$$

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

$$f(u^o) + \bar{y} g(u^o) < f(\bar{u}) + \bar{y} g(\bar{u}) \tag{3.13}$$

for at least one point $\bar{y} \in Y$. Also, since $y^o \in ri(Y^c)$ there exists (see [54, Theorem 6.4]) a scalar $\theta > 1$ such that

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

belongs to Y^c. Multiplying equation (3.12) by θ and multiplying inequality (3.13) by $(1-\theta)$, which is negative and reverses the inequality, and adding gives:

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

But this inequality contradicts (3.11), showing that our supposition that u^o is not pareto optimal is untenable. This completes the proof. □

When $f(u) = ub$, $g(u) = (d - uD)$ and $U = \{u \in R^k : uA \leq c\}$ as in Benders Decomposition for mixed integer programs, problem (3.8) 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 the master problem (3.6). Therefore to find a pareto optimal point among all the alternate optimal solutions to problem (3.8), we solve problem (3.9) which is the linear program:

$$\begin{aligned} \text{Maximize} \quad & \left\{dy^o + u(b-Dy^o)\right\} \\ \text{subject to:} \quad & u(b-D\hat{y}) = \hat{z} - d\hat{y} \\ & uA \leq c. \end{aligned} \tag{3.14}$$

We should note that varying the core point y^o might conceivably generate different pareto optimal cuts. Also, any implementation of a strong cut version of Benders algorithm has the option of generating pareto optimal cuts at every iteration, or possibly, of generating these cuts only periodically. The tradeoff will depend upon the computational burden of solving problem (3.9) 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, $Y = \left\{y \in R^k : y \geq 0 \text{ and integer}\right\}$ then any point $y^o > 0$ will suffice; if $Y = \left\{y \in R^k : y_j = 0 \text{ or } 1 \text{ for } j = 1,2,\ldots, k\right\}$ then any vector y^o with $0 < y_j^o < 1$ for $j = 1,2,\ldots, k$ suffices; and if

$$Y = \left\{y \in R^k : \sum_{j=1}^{k} y_j \leq p,\ y \geq 0 \text{ and integer}\right\}$$

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

is a core point.

One particular version of the preceding theorem merits special mention. Suppose that U is a product of sets $U = U^1 \times U^2 \times \ldots \times U^J$ and that f and g are additively separable over the sets U^j; that is,

$$f(u) = \sum_{j=1}^{J} f_j\left(u_{(j)}\right)$$

and

$$g(u) = \sum_{j=1}^{J} g_j\left(u_{(j)}\right)$$

where $u = u_{(1)}, u_{(2)}, \ldots u_{(j)}$ is a partition of u with $u_{(j)} \in U^j$. The notation $u_{(j)}$ distinguishes this vector from the component u_j of $u \in U$. Then for any $y \in Y$, the subproblem (3.4) separates as:

$$v(y) = \sum_{j=1}^{J} v_j(y)$$

where for each j

$$v_j(y) = \underset{u_{(j)} \in U^j}{\text{Max}} \left\{ f_j\left(u_{(j)}\right) + y g\left(u_{(j)}\right) \right\}. \qquad (3.15)$$

Since for any $u_{(j)} \in U^j$

$$f_j\left(u_{(j)}\right) + \hat{y} g_j\left(u_{(j)}\right) \leq v_j(\hat{y})$$

the vector u belongs to $U(\hat{y})$, meaning that the sum over j of the lefthand sides of these expressions equals the sum of the righthand sides, if and only if

$$f_j\left(u_{(j)}\right) + \hat{y}g_j\left(u_{(j)}\right) = v_j(\hat{y})$$

for all j. That is, choosing u to be one of the alternate optimal solutions to (3.8) is equivalent to $u_{(j)}$ being an alternate optimal solution to (3.15) when $y=\hat{y}$. Consequently, finding a pareto optimal cut decomposes into independent subproblems, as recorded formally in the following corollary stated in terms of the notation just introduced.

Corollary 3.1: Let y^o be a core point of Y, and for each j = 1,2,..., J let $U^j(\hat{y})$ denote the set of optimal solutions to the optimization problem

$$\underset{u_{(j)} \in U^j}{\text{Max}} \left\{ f_j\left(u_{(j)}\right) + \hat{y}g_j\left(u_{(j)}\right) \right\}$$

and let $u^o_{(j)}$, solve the problem:

$$\underset{u_{(j)} \in U^j(\hat{y})}{\text{Max}} \left\{ f_j\left(u_{(j)}\right) + y^o g_j\left(u_{(j)}\right) \right\}.$$

Then $u^o = \left(u^o_{(1)}, u^o_{(2)}, \ldots, u^o_{(J)}\right)$ is pareto optimal for (2).

The separability of f and g in this discussion has historically been a major motivation for considering resource directive decomposition and Lagrangian relaxation. In this case, problem (3.14) decomposes into

several linear programs, one for each subvector $u_{(j)}$ of u.

3.4 Accelerating Benders Method For Network Optimization

Although solving the linear program (3.14) always generates pareto optimal cuts whenever 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 involves network optimization, special purpose network algorithms might be preferred to the general purpose methodology.

In this section we describe special network algorithms for generating strong cuts for the facility location and network design problems. We will discuss several different algorithms, ranging from those which produce cuts that dominate the standard Benders cut, to more elaborate algorithms that actually produce pareto optimal cuts.

3.4.1 Strong Cuts for the Facility Location Problem

We begin by considering a facility location problem formulated as the following mixed integer program:

$$v = \text{Min} \quad \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij} x_{ij} + \sum_{j=1}^{m} d_j y_j$$

$$\text{subject to:} \quad \sum_{j=1}^{m} x_{ij} \geq 1 \qquad (3.16)$$

$$x_{ij} \leq y_j \qquad (1 \leq i \leq n)$$

$$x_{ij} \geq 0 \qquad (1 \leq j \leq m)$$

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

$$y \in Y$$

where

m = number of potential facilities

n = number of customers

and Y = set of feasible values for $y \subseteq (0,1)^n$

If $y_j = 1$, we construct facility j and incur a fixed cost of d_j. If $x_{ij} = 1$ customer i receives service at facility j. The first constraint requires that each customer be serviced by some facility. The second constraint states that no customer can be serviced at a facility unless that facility is constructed. In chapter IV of this thesis we suggest reasons for choosing this particular form of the problem formulation instead of an equivalent formulation with constraints $\sum_i x_{ij} \leq n y_j$ for all j in place of the constraints $x_{ij} \leq y_j$ for all i and j.

If $Y = \left\{ y \middle| \sum_{j=1}^{m} y_j = p \right\}$, $n = m$, and $c_{jj} = 0$ for all j, then (3.16) becomes the well-known p-median location problem. If $Y = \{0,1\}^m$, then (3.16) becomes the well-known uncapacitated plant location problem. The set Y might incorporate a number of additional conditions imposed upon the configuration of open (i.e., $y_j = 1$) facilities. Among these might be contingency constraints such as "location i is opened only if location j is

opened," multiple choice constraints such as "open at most two of the locations i, j and k," and other conditions of this nature.

Suppose we fix $y = \bar{y} \varepsilon Y$; then (3.16) reduces to the following pure linear programming subproblem:

$$
\begin{aligned}
v(\bar{y}) = \text{Min} \quad & \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij} x_{ij} \\
\text{subject to:} \quad & \sum_{j=1}^{m} x_{ij} \geq 1 \\
& 0 \leq x_{ij} \leq 1 \qquad j \varepsilon O \\
& 0 \leq x_{ij} \leq 0 \qquad j \varepsilon C \\
& \qquad\qquad\qquad\qquad (1 \leq i \leq n)
\end{aligned}
\tag{3.17}
$$

where

$O = \left\{ j \mid \bar{y}_j = 1 \right\}$, the set of open facilities, and

$C = \left\{ j \mid \bar{y}_j = 0 \right\}$, the set of closed facilities.

The linear program dual of this problem is:

$$
v(\bar{y}) = \text{Max} \sum_{i=1}^{n} \left[\lambda_i - \sum_{j=1}^{m} \bar{y}_j \pi_{ij} \right]
$$

subject to:

$$\lambda_i - \pi_{ij} \leq c_{ij} \qquad (1 \leq i \leq n) \qquad (3.18)$$

$$\lambda_i \geq 0 \qquad (1 \leq j \leq m)$$

$$\pi_{ij} \geq 0.$$

Any solution to this problem determines a cut of the form:

$$v \geq \sum_{i=1}^{n} \left(\lambda_i - \sum_{j=1}^{m} \pi_{ij} y_j \right) + \sum_{j=1}^{m} d_j y_j \ . \qquad (3.19)$$

(Note that we have appended the term $\Sigma d_j y_j$ as well. This term was omitted from the objective function of the subproblem because it is a constant for any given choice of the configuration variables y_j).

Both the primal and dual subproblems are solved easily by inspection. For the primal subproblem, each customer i goes to the closest facility which has been constructed, or

$$x_{ij\ i)} = 1 \text{ where } c_{ij(i)} \equiv \underset{j \in 0}{\text{Min}}\ c_{ij}.$$

The dual subproblem (19) possesses the following "natural" solution:

$$\bar{\lambda}_i = c_{ij(i)}$$

$$\bar{\pi}_{ij} = 0 \qquad \text{if } j \in 0, \quad 1 \leq j \leq n$$

$$\bar{\pi}_{ij} = \max\left(0, \bar{\lambda}_i - c_{ij}\right) \qquad \text{if } j \in C, \quad 1 \leq j \leq n.$$

The optimal dual variables have a convenient interpretation in terms of the facility location problem (see Balinski [5]). $\bar{\lambda}_i$ is the cost of servicing customer i when $y=\bar{y}$. $\bar{\pi}_{ij}$ is the reduction in the cost of servicing customer i when facility j is opened and $y_i=\bar{y}_i$ for all $i\neq j$. So for the dual subproblem solution, we can construct the following Benders cut,

$$z \geq w - \sum_{j=1}^{m} \mu_j y_j + \sum_{j=1}^{m} d_j y_j \qquad (3.20)$$

where $$w = \sum_{i=1}^{n} \bar{\lambda}_i$$

and $$\mu_j = \sum_{i=1}^{n} \bar{\pi}_{ij} .$$

Note that w is the total servicing costs when $y=\bar{y}$ and that μ_j is the total reduction in servicing costs if facility j is opened and all other facilities retain their current, open vs. closed, status.

For reference purposes, we shall refer to the cut in (3.20) as a type A cut.

Careful inspection of the linear program (3.17) reveals that for most problems it will have a degenerate optimal basis. This implies that it usually will be possible to derive more than one Benders cut. We next describe procedures for generating alternative cuts, cuts that will usually be superior to the "standard" cut (3.20).

In deriving the Benders cut (3.20), we only considered the savings from opening a new facility, i.e. increasing some y_j from 0 to 1. We did

not, however, consider the added servicing costs produced by closing a facility.

Let $$c_{ik(i)} = \min\{c_{iq} : 1 \leq q \leq n \text{ and } q \neq j(i)\}$$

and let $$\sigma_i = \max\{c_{ik(i)} - c_{ij(i)}, 0\}.$$

If facility $j(i)$ is closed, then the service cost for customer i must be at least $c_{ik(i)}$. Whenever $\sigma_i > 0$ customer i will suffer an increase in service cost of at least σ_i if facility j is closed, i.e., if y_j is decreased from 1 to 0. Therefore

$$\nu_j = \sum\{\sigma_i : 1 \leq i \leq n \text{ and } j=j(i)\} \tag{3.21}$$

is the minimum total service cost incurred from all customers by closing facility j. So we can write a new cut, which we will refer to as a type B cut, as the following:

$$v \geq w + \sum_{j \in 0} (1-y_j)\nu_j - \sum_{i \in C} \mu_i y_i + \sum_{j=1}^{m} d_j y_j . \tag{3.22}$$

Notice that as long as there is a $\nu_j \neq 0$ and there is a $\hat{y} \in Y$ such that $\hat{y}_j \neq 0$, the type B cut will dominate the type A cut.

This strengthening of the Benders cut by considering the penalty of a customer being diverted to his second nearest facility has also been described by Balinski [5].

Further improvements are also possible. In the next section we extend these observations to derive pareto-optimal cuts for the facility

location problem.

3.4.2 Pareto Optimal Cuts for the Facility Location Problem

In this section we derive an efficient special purpose algorithm for solving the linear program (3.14) for generating pareto optimal cuts for the facility location model. The algorithm invokes the decomposition property of corollary 3.1 combined with a parametric solution technique to solve each of the subproblems.

First, we note that for any choice of $\bar{y} \in Y$, the linear programs (3.17) and (3.18) decompose into separate subproblems, one for each index $i = 1,2,\ldots, n$. Also, the "natural solution"

$$\bar{\lambda}_i = c_{ij(i)} \equiv \min\{c_{ij} : j \in 0\}$$

$$\bar{\pi}_{ij} = 0 \qquad \text{if } j \in 0$$

and $$\pi_{ij} = \max(0, \bar{\lambda}_i - c_{ij}) \qquad \text{if } j \in C,$$

to the linear programming dual problem (19) has the property that the optimal value of the ith subproblem is $v_i(y) = \bar{\lambda}_i$. Consequently, corollary 1 with $u_{(i)} = (\lambda_i, \{\pi_{ij}\}_j)$ implies that solving for each i the subproblem

$$\text{Max} \quad \lambda_i - \sum_{j=1}^{m} y_j^o \pi_{ij}$$

$$\text{subject to:} \quad \lambda_i - \sum_{j=1}^{m} \bar{y}_j \pi_{ij} = \bar{\lambda}_i \qquad (3.23)$$

$$\lambda_i - \pi_{ij} \leq c_{ij}$$

$$\pi_{ij} \geq 0 \qquad (j = 1,2,\ldots, m)$$

$$\lambda_i \geq 0$$

provides a pareto-optimal vector with components λ_i and π_{ij} for $i = 1,2,\ldots, n$ and $j = 1,2,\ldots, m$. Here, as before, $\bar{y}$ denotes the current value of the integer variables and y^o belongs to the core of y, i.e., $y \in ri(Y^c)$.

Our first objective is to show that for each i, the subproblem (3.23) is piecewise linear as a function of λ_i. Note that since the equality constraint of this problem reads

$$\lambda_i - \sum_{j \in 0} \pi_{ij} = \bar{\lambda}_i = c_{ij(i)}$$

and since

$$\lambda_i - \pi_{ij(i)} \leq c_{ij(i)}$$

and

$$\pi_{ij} \geq 0 \quad \text{for all } j,$$

it must be true that

$$\pi_{ij} = 0 \quad \text{for all } j \neq j(i), \quad j \in 0$$

and

$$\pi_{ij(i)} = \lambda_i - c_{ij(i)} = \lambda_i - \bar{\lambda}_i \ .$$

Also, if we substitute for λ_i in the objective function of (3.23) from the equality constraint, the objective becomes:

$$\text{Max } \bar{\lambda}_i + \sum_{j=1}^{m} \left(\bar{y}_j - y_j^o\right)\pi_{ij} \ .$$

Since $\bar{y}_j = 0$ if $j \epsilon C$ the coefficient $\varepsilon_j = \bar{y}_j - y_j^o$ of π_{ij} is nonpositive. Thus an optimal choice of π_{ij} satisfying the two constraints

$$\lambda_i - \pi_{ij} \leq c_{ij} \quad \text{and} \quad \pi_{ij} \geq 0$$

is

$$\pi_{ij} = \max \left\{0, \lambda_i - c_{ij}\right\} \ .$$

Collecting these results, we see that the optimal value of problem (3.23) as a function of the variable λ_i is:

$$\bar{\lambda}_i + \varepsilon_{j(i)}\left(\lambda_i - \bar{\lambda}_i\right) + \sum_{j \epsilon C} \varepsilon_j \max \left\{0, \lambda_i - c_{ij}\right\} \ . \qquad (3.24)$$

As an aid to optimizing (3.24), we note the following upper and lower bounds on λ_i:

$$\bar{\lambda}_i \leq \lambda_i \leq L_i$$

where, by definition, $L_i = \min\{c_{ij}: j\varepsilon 0 \text{ and } j\neq j(i)\}$. The lower bound is simply a consequence of the equality constraint of problem (3.23), because each $\bar{y}_j \geq 0$ and each $\pi_{ij} \geq 0$. The upper bound is a consequence of our previous observation that for all $j\neq j(i)$ and $j\varepsilon 0$, $\pi_{ij}=0$ and, therefore, the constraint $\lambda_i - \pi_{ij} \leq c_{ij}$ becomes $\lambda_i \leq c_{ij}$.

Now, since the function (3.24) is piecewise linear and concave in λ_i, we can minimize it by considering the linear segments of the curve in the interval $\bar{\lambda}_i \leq \lambda_i \leq L_i$ in order from left to right until the slope of any segment becomes nonpositive. Formally,

(1) Start with $\lambda_i = \bar{\lambda}_i$.

(2) Let $T = \{j\varepsilon C: c_{ij} \leq \lambda_i\}$ and let $s = \varepsilon_{ij(i)} + \sum\{\varepsilon_j: j\varepsilon T\}$. s is the slope of the function (25) to the right of λ_i.

(3) If $s \leq 0$, then stop; λ_i is optimal. If $s > 0$ and $T=C$, then stop, $\lambda_i = L_i$ is optimal.

(4) Let $c_{ik} = \min\{c_{ij}: j\varepsilon C \text{ and } j\notin T\}$. If $L_i \leq c_{ik}$, set $\lambda_i = L_i$ and stop. Otherwise, increase λ_i to c_{ik}. Repeat steps (2)-(4).

Once the optimal value to λ_i is found using this algorithm for each i the remaining variables π_{ij} can be set using the rules given above. Then by virtue of corollary 1, the cut obtained by substituting these values in (3.19) is pareto optimal.

The above algorithm should be very efficient. For each customer i, at most m (m=# of possible facilities) steps must be executed. So in the worst case, the procedure is an 0 (# of customers) (# of possible facilities) algorithm.

We might emphasize that this algorithm determined a pareto optimal cut for any given point y^o in the core of Y. Also, the algorithm applies to any of the possible modeling variations that we might capture in Y, such as the contingency and configuration constraints mentioned in section 3.4.1.

3.4.3 Further Results Concerning Facility Location

As we noted in section 3.4.1, the standard Benders cut considers savings in servicing costs when a new facility is opened. The improved type B cut introduces additional servicing costs that must be incurred whenever an open facility is closed. In this section we show that any Benders cut generated from an optimal solution to the dual subproblem (3.18) has a similar interpretation. We also present a new type of cut for the p-median problem and discuss its interpretation.

An Interpretation:

First, we introduce some new notation. The δ-neighborhood of customer i, denoted $N_i(\delta)$, is the set of facility locations j satisfying $c_{ij} \leq \lambda_i + \delta$. The interior of the δ-neighborhood is defined as $N_i^o(\delta) \equiv \left\{j : c_{ij} < \bar{\lambda} + \delta\right\}$.

Recall that from the last section that $\lambda_i \geq \bar{\lambda}_i$ in any solution to the subproblem (3.18); that is, $\lambda_i = \bar{\lambda}_i + \delta_i$ for some $\delta_i \geq 0$. Consequently, varying δ_i, and hence the size of the δ-neighborhood, is equivalent to varying λ_i. The operation in the pareto-optimal cut algorithm of increasing λ_i until $s \leq 0$ has the following interpretation: increase the δ-neighborhood about customer i until $\sum\{\varepsilon_j : j \in N_i(\delta_i)\} \leq 0$.

Figure 1 gives a small example of a δ-neighborhood. Assume distances in the figure are drawn to scale and that the neighborhood is constructed around customer i. Nodes, 2,3,7 and 9 represent possible facility locations. Assume that only node 9 is open in the current solution $\bar{y}$. As indicated in the figure, the current δ-neighborhood contains nodes 3,7 and 9. If $\varepsilon_9 + \varepsilon_3 + \varepsilon_7 = s > 0$, then the pareto optimal cut algorithm would expand the neighborhood to the next nearest facility, which is node 2.

The cut determined by the neighborhood pictured in figure 1 has the following interpretation. For notational convenience let us assume at this point that customer i is the only customer, and that there are no fixed charges, i.e. $d_j=0$ for all j. Then we can express the cut for any optimal solution to subproblem (3.18) as:

$$v \geq \lambda_i - \sum_{j=1}^{m} \pi_{ij} y_j .$$

As we noted in deriving equation (3.24) from problem (3.23), every solution to subproblem (3.18) can be written as

$$\lambda_i = \bar{\lambda}_i + \delta_i \quad \text{for some} \quad \delta_i \geq 0$$

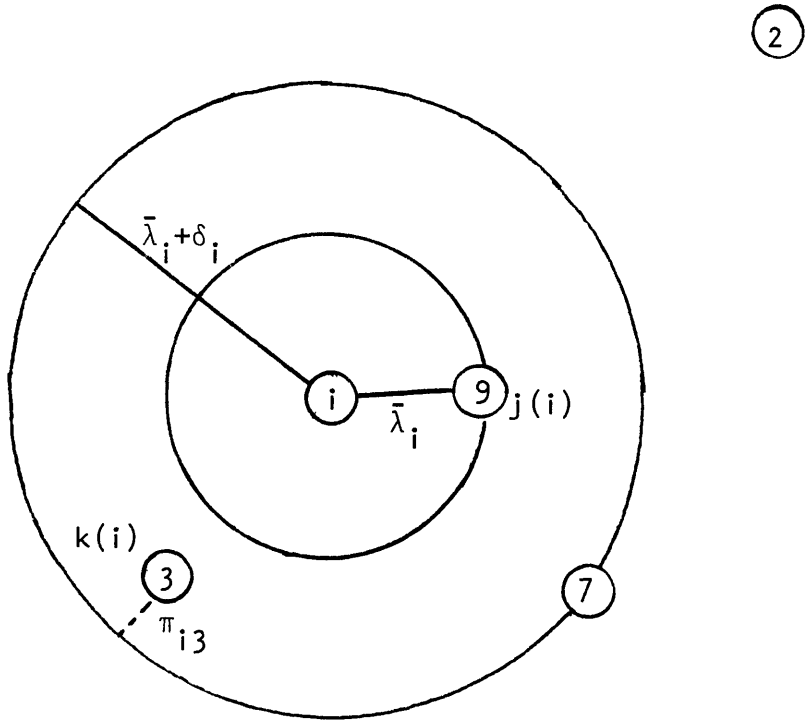


Figure 3.1 Neighborhood About A Customer

$$\bar{\lambda}_i = c_{ij(i)} \equiv \min\left\{c_{ij} : j \in 0\right\}$$

$$\pi_{ij} = \begin{cases} \lambda_i - \bar{\lambda}_i = \delta_i & \text{if } j = j(i) \\ \lambda_i - c_{ij} & \text{if } j \in C \cap N_i(\delta_i) \\ 0 & \text{otherwise.} \end{cases}$$

Consequently, the cut corresponding to figure 3.1 is:

$$v \geq \bar{\lambda}_i + \delta_i (1 - y_9) - (\bar{\lambda}_i + \delta - c_{i3}) y_3 .$$

If we set $y_9 = 0$ to close the facility at node 9, then customer i must be serviced from outside the neighborhood (or on its boundary) at a cost v of at least $\bar{\lambda}_i + \delta_i$ unless the facility at node 3 is opened. If this facility is opened, then the servicing cost for node i becomes c_{i3}. The coefficient of y_3 compensates for this reduction in service cost when $y_3 = 1$.

The general situation is much the same. Given any neighborhoods for the customers, let $w = \sum_{i=1}^{n} \bar{\lambda}_i$ be the current routing cost, let $\nu_j = \sum\left\{\delta_i : j = j(i)\right\}$, and let $\mu_j = \sum_{i=1}^{n} \pi_{ij}$. Substituting these values in the form of the cut expressed in (3.19) gives:

$$v \geq w + \sum_{j \in 0} \nu_j (1 - y_j) - \sum_{j \in C} \mu_j y_j + \sum_{j=1}^{n} d_j y_j . \qquad (3.25)$$

The coefficient ν_j accounts for the fact that the open facility j may lie interior to several neighborhoods. Closing this facility increases routing costs to the boundary of each of these neighborhoods unless some closed facility within any neighborhood is opened. The coefficient μ_j records the savings for opening facility j considering all the neighborhoods that it belongs to.

Suppose, as before, that $c_{ik(i)}$ denotes the cost to the closest facility $k(i)\neq j(i)$ to node i. Setting $\lambda_i = \bar{\lambda}_i$, and $\delta_i = \max\left(0, c_{ik(i)} - c_{ij(i)}\right)$, expression (3.25) reduces to the type B cut introduced in section 3.4.1.

A New Cut for the p-Median Problem:

When specialized, our cut-generating techniques provide a new type of Benders cut for the p-median problem, one that dominates the type B cut. To simplify our development, we temporarily assume that all servicing costs c_{ij} are nonnegative and that $c_{ii}=0$ for all i. Recall that every node in a p-median problem is both the source of a customer and a potential facility location (thus m=n). The problem involves no fixed costs, $d_j=0$ for all j.

As we have seen, the type B cut introduces penalties for customers forced to travel to their nearest closed facility. For the p-median problem, these penalties separate into two groups: (i) a customer and a facility are located at the same node i. Then the servicing cost for that customer is $c_{ii}=0$ and the penalty in servicing cost for this customer is $c_{ik(i)} \equiv \min\left\{c_{ij}: j\neq i\right\}$ if the facility is closed. (ii) a customer, but no facility, is located at node i. Then the closing of any open facility

need not insure any servicing penalty, since the customer might conceivably be serviced by a facility at node i at cost $c_{ii}=0$. Stated in terms of the neighborhood interpretation, this observation implies that the δ_i-neighborhood about customer i is of minimal size, $N_i(\delta_i) = N_i(0)$, if $y_i=0$ in the current solution y; if $y_i=1$, then $c_{ij(i)} \equiv \min\{c_{ij}: j\varepsilon 0\} = c_{ii}$ and $\delta_i=c_{ik(i)}$ is the size of the neighborhood.

Since closing a facility at node j only contributes to the penalty in the type B cut of a customer at that node, the term $\nu_j \equiv \{\delta_i: j=j(i)\}$ equals $c_{jk(j)}$, the distance to node j's second nearest neighbor and the type B cut is written in the form of expression (3.25) as:

$$v \geq w + \sum_{j\varepsilon 0} c_{jk(j)}(1-y_j) - \sum_{j\varepsilon C}\mu_j y_j \ . \qquad (3.26)$$

The terms w and μ_j are defined as before.

The algorithm presented in section 3.4.2 shows how to expand the neighborhoods about every customer from the values associated with this type B cut in order to obtain pareto-optimality. Although the new cut must be pareto-optimal, there is no guarantee that it dominates the type B cut.

To develop a cut that dominates the type B cut, we proceed as follows. We maintain the neighborhood about nodes whose facilities are closed at their minimal size $\delta_j=0$, and we increase the neighborhoods about the other nodes all by the same amount. That is, we set $\delta_j=c_{jk(j)} + \delta$ for every node j with $\bar{y}_j=1$. This procedure avoids the formal slope checking mechanism of the algorithm for generating pareto-optimal cuts. Although other options are certainly possible, choosing to expand every neighborhood equally leads to a very simple implementation.

The choice $\bar{\delta}$ of δ is governed by two restrictions. First, any value of δ determines values $\lambda_i = \bar{\lambda}_i + \delta_i = \bar{\lambda}_i + c_{ik(i)} + \delta$ for $i\varepsilon 0$ and $\lambda_i = \bar{\lambda}_i$ for $i\varepsilon C$ of the variables in the objective function (3.24) of the dual linear programming subproblems (3.23). As noted in section 3.4.1, these values will be feasible only if

$$\lambda_i \leq \min\left\{c_{ij} : j\varepsilon 0 \text{ and } j \neq j(i)\right\} .$$

That is, the interior of every neighborhood may contain at most one open facility. The second restriction is that every closed facility lie interior to at most one neighborhood about an open facility. Although this restriction is not imposed by the linear programs (3.23), later we will show by an example that the new cut need not dominate the type B cut if this condition is not fulfilled. Our choice of δ is made as large as possible, consonant with these two restrictions.

We will call the result of this procedure

$$v \geq w + \sum_{j\varepsilon 0} \left(c_{jk(j)} + \bar{\delta}\right)\left(1 - y_j\right) - \sum_{j\varepsilon C} \left(\mu_j + \bar{\delta}_j\right) y_j \ . \qquad (3.27)$$

a type C cut. Note that the coefficient of the closed facilities j must be altered from the values μ_j in the type B cut (3.26). Since out restrictions on $\bar{\delta}$ insure that every closed facility j lies interior to only one neighborhood q, if any, about an open facility, as we increase δ only the term π_{qj} in the saving expression $\mu_j = \sum_{i=1}^{n} \pi_{ij}$ changes. δ_j equals the difference between $\pi_{qj} = \max\left(\bar{\lambda}_q - c_{qj}, 0\right) = \max\left(c_{qk(q)} + \delta - c_{qj}, 0\right)$, see section

3.4.1, at $\delta=0$ and at $\delta=\bar{\delta}$. Note that this observation implies that $\bar{\delta}_j \leq \bar{\delta}$ for all $j \in C$.

In comparing cuts, we noted previously that type B cuts dominate type A cuts as long as at least one $v_j \neq 0$. The following result summarizes the relationship between type B and type C cuts.

Proposition 3.1: For a given iteration of Benders decomposition for the p-median problem, a type C cut will either dominate or be equivalent to a type B cut.

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

$$y_1 + y_2 + \cdots + y_n = p$$

$$y_j \in \{0,1\}.$$

Let $R_C(y)$ and $R_B(y)$ denote the righthand sides of the type C cut (3.27) and the type B cut (3.26). Then

$$DIFF(y) = R_C(y) - R_B(y) = \sum_{j \in 0} \bar{\delta}\left(1-y_j\right) - \sum_{j \in C} \bar{\delta}_j y_j .$$

By the p-median constraint, if K of the facilities $j \in C$ are opened, then K of the facilities $j \in 0$ must be closed. As we have noted just prior to the proposition, though, $\bar{\delta}_j \leq \bar{\delta}$ for all $j \in C$. These two facts imply that $DIFF(y) \geq 0$, so the type C cut is always at least as strong as the type

B cut.

Reviewing the definition of the type C cut and the proof of this proposition shows that our assumptions that service costs are nonnegative and that $c_{ii} = 0$ for all i are dispensable. These assumptions merely lead to more attractive interpretations and motivation.

Example 3.2

As an illustration of these strong cuts, consider a two-median problem on the network in figure 3.2 .

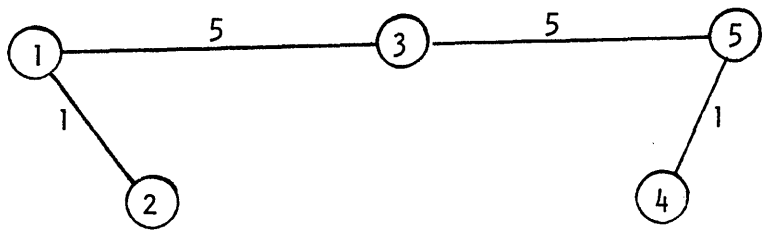


Figure 3.2 p-median example

The number on each arc is the distance between the nodes incident to that arc. The cost c_{ij} for servicing a customer at node i from a facility at node j is computed as the shortest distance between these nodes.

Assume that the current configuration $\bar{y}$ has $\bar{y}_1 = 1$, $\bar{y}_5 = 1$ and $\bar{y}_2 = \bar{y}_3 = \bar{y}_4 = 0$. Then customers at nodes 1 through 3 will be serviced at node 1 and the customers at nodes 4 and 5 will be serviced at node 5.

The usual application of Benders method gives the following type A cut:

$$v \geq 7 - y_2 - 5y_3 - y_4$$

Incorporating the penalty of closing a facility, we obtain the following type B cut:

$$v \geq 7 + 1\left(1-y_1\right) - y_2 - 5y_3 - y_4 + 1\left(1-y_5\right).$$

Using our δ-neighborhood concept, we find that δ equals 4 (for $\delta > 4$, node 3 lies in the interior of the neighborhoods about node 1 and node 5), and we obtain the following type C cut:

$$v \geq 7 + (1+4)\left(1-y_1\right) - 5y_2 - 5y_3 - 5y_4 + (1+4)\left(1-y_5\right).$$

Note that the type C cut dominates the types A and B cuts.

If we ignored the restriction prohibiting node 3 from being in the interior of the neighborhoods about both nodes 1 and 5, we could expand the neighborhoods until $\delta=9$ and the cut would become:

$$v \geq 7 + 10\left(1-y_1\right) - 10y_2 - 15y_3 - 10y_4 + 10\left(1-y_5\right)$$

Observe that this cut does not dominate the type B cut (take $y_2=y_3=1$, $y_1=y_4=y_5=0$). The difficulty is that $\bar{\delta}_3=10$ exceeds $\bar{\delta}=9$.

3.4.4 Strong Cuts For Fixed Charge Network Design

Next we discuss the generation of strong cuts for the fixed charge network design problem. This problem is formulated as:

$$\text{Minimize} \quad \sum_{(i,j)\in A} \sum_{k} \sum_{\ell} c_{ij} x_{ij}^{k\ell} + \sum_{(i,j)\in A} b_{ij} y_{ij}$$

$$\text{subject to:} \quad \sum_{j} x_{ij}^{k\ell} - \sum_{j} x_{ji}^{k\ell} = \begin{cases} 0 & i \neq k, \quad i \neq \ell \\ R_{k\ell} & i = k \quad \text{for all } k, \ell \\ -R_{k\ell} & i = \ell \end{cases}$$

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

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

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

where

i,j,k,ℓ are nodes indices;

$x_{ij}^{k\ell}$ 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 network;

R_{ij} is the amount of flow that must be routed between nodes i and j.

The break-bulk problem discussed in chapter I can be viewed as a special case of this problem. To formulate the problem in this way, we modify the network representation of the break bulk problem by splitting every node corresponding to a break-bulk center into a "receiving node" and a "sending node". Introducing a "throughput arc" connecting these nodes, we then identify the construction of this throughput arc with construction (or rental) of the break-bulk center. Since this construction cost will be positive, we can formally let the candidate arcs A consist of both throughput arcs and routing arcs, which have zero fixed costs. We can always assume, then, that all of the routing arcs are "constructed" in the optimal solution to the design problem.

In chapter II we showed that facility location models can be formulated as a network design problem, though possibly with additional side constraints on the configuration variables y_{ij}. Since this transformation does not, however, seem to provide new insight concerning strong cuts for the facility location models, we have treated the problems separately.

Finally, we noted that the constraints $x_{ij}^{k\ell} \leq R_{k\ell} y_{ij}$ for all $(i,j) \in A$ and all k,ℓ can be compressed into constraints $\sum_{k,\ell} x_{ij}^{k\ell} \leq \left(\sum R_{k\ell}\right) y_{ij}$

for all $(i,j)\varepsilon A$ without affecting the feasible solutions to the problem. The reasons for not doing so are similar to those for not compressing the constraints of the facility location model (see chapter IV for a further discussion of this issue).

Inspection of the fixed charge design problem indicates that the selection of flow variables decomposes into a series of problems of the following form:

$$\text{Minimize} \quad \sum_i \sum_j c_{ij} x_{ij}$$

$$\text{subject to:} \quad \sum_i x_{ik} - \sum_j x_{kj} = R_{k\ell}$$

$$\sum_i x_{i\ell} - \sum_j x_{\ell j} = -R_{k\ell} \qquad (3.28)$$

$$\sum_i x_{ih} - \sum_j x_{hj} = 0 \qquad \text{all } h\neq k,\ h\neq \ell$$

$$x_{ij} \leq R_{k\ell} y_{ij}$$

$$x_{ij} \geq 0, \quad y_{ij}\varepsilon\{0,1\} \qquad \text{all } i \text{ and } j$$

where $1\leq k\leq N$ and $1\leq \ell\leq N$. For notational simplicity we have subsumed the superscripts k and ℓ.

For any assignment of the y_{ij} variables, (3.28) becomes a shortest path problem. Let the configuration variables y be fixed at values $y=\bar{y}$.

The dual of (3.28) is:

$$\text{Maximize} \quad R_{k\ell}\left[\left(\pi_\ell - \pi_k\right) - \sum_i \sum_j \gamma_{ij}\bar{y}_{ij}\right] .$$

$$\text{subject to:} \quad \left(\pi_j - \pi_i\right) - \gamma_{ij} \leq c_{ij} \tag{3.29}$$

$$\gamma_{ij} \geq 0, \quad \pi_h \text{ unrestricted.}$$

Suppose that $\left\{\bar{\pi}_i^{k\ell}\right\}$ and $\left\{\bar{\gamma}_{ij}^{k\ell}\right\}$ solve (3.29); then in the context of Benders decomposition,

$$v \geq \sum_k \sum_\ell R_{k\ell}\left[\left(\bar{\pi}_\ell^{k\ell} - \bar{\pi}_k^{k\ell}\right) - \sum_i \sum_j \bar{\gamma}_{ij}^{k\ell} y_{ij}\right]$$

defines a cut. (As before, we have dropped the constant term $b_{ij}y_{ij}$ from the righthand side of all cuts.) In order to simplify the following discussion let us consider only one of the problems (3.28) (i.e. let k=t and ℓ=s) and drop the indices k and ℓ. We assume that $R_{k\ell}$=1 for the problem being considered. Further assume that $\left\{\pi_i^*\right\}$ and $\left\{\gamma_{ij}^*\right\}$ solve (3.29). So the Benders cut becomes:

$$v \geq \left(\pi_t^* - \pi_s^*\right) - \sum_i \sum_j \gamma_{ij}^* y_{ij} .$$

Now problem (3.28) 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.

Since (3.28) is usually degenerate, its dual (3.29) will generally have multiple optimal solutions.

With $y=\bar{y}$, regarding (3.28) as a shortest path problem gives us one possible interpretation of the dual variables. π_i is the shortest distance between nodes s and i on the network 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 s and t if $y_{ij}=1$ (i.e. if arc (i,j) is added to the network defined by $\bar{y}$).

Let $\{\pi_i^1, \gamma_{ij}^1\}$ be set of optimal dual values defined by the above procedure. We shall refer to the cut associated with this optimal dual solution as the standard cut.

Other values of the optimal dual variables are usually possible. We next describe a procedure that yields another optimal dual solution. This solution has the property that the Benders cut that it defines is never weaker than the standard cut; that is, is $\{\pi_i^2, \gamma_{ij}^2\}$ denotes the optimal dual values produced by this new procedure, then

$$\pi_t^2 - \pi_s^2 - \sum_i \sum_j \gamma_{ij}^2 y_{ij} \geq \pi_t^1 - \pi_s^1 - \sum_i \sum_j \gamma_{ij}^1 y_{ij}$$

for all possible values of y. Usually this new set of dual values produces a cut which dominates the standard cut.

The new procedure uses two pices of information-shortest path distances from node s to all other nodes and shortest path distances from all nodes to t. Define

D^* = optimal value of (3.28) when $y=\bar{y}$

π_i^1 = minimum distance from node s to node i on the network defined by $y=\bar{y}$.

and

D_i = minimum distance from node i to node t on the network defined by all the arcs in A [i.e. $y_{ij}=1$ for all $(i,j)\in A$].

Let

$$\Delta_i = D^* - D_i$$

$$\pi_s^2 = 0, \quad \pi_t^2 = D$$

$$\pi_i^2 = \min\left(\pi_i^1, \Delta_i\right) \qquad \text{all } i\neq s,\ i\neq t$$

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 3.2: (a) The set of dual values $\{\pi_i^2, \gamma_{ij}^2\}$ is an optimal solution to (3.29) when $y=\bar{y}$.

(b) $\gamma_{ij}^2 \leq \gamma_{ij}^1$.

That is, the Benders cut generated by the dual values $\{\pi_i^2, \gamma_{ij}^2\}$ will either dominate or be identical to the standard cut.

Proof [for part (b)]. By cases.

Case 1 Suppose that $\pi_i^2 = \pi_i^1$.

Then $\pi_j^2 - \pi_i^2 \leq \pi_j^1 - \pi_i^1$,

which implies that $\gamma_{ij}^2 \leq \gamma_{ij}^1$.

Case 2 Suppose that $\pi_i^2 = \Delta$.

By the triangle inequality we have,

$$D_i \leq D_j + c_{ij}$$

or $$D_i - c_{ij} \leq D_j .$$

Consequently,

$$\Delta_j = D^* - D_j \leq D^* - (D_i - c_{ij}) = D^* - D_i + c_{ij}$$

$$\Delta_j \leq \Delta_i + c_{ij}$$

and

$$\pi_j^2 - \pi_i^2 \leq \Delta_j - \pi_i^2 = \Delta_j - \Delta_i$$

$$\leq \Delta_i + c_{ij} - \Delta_i$$

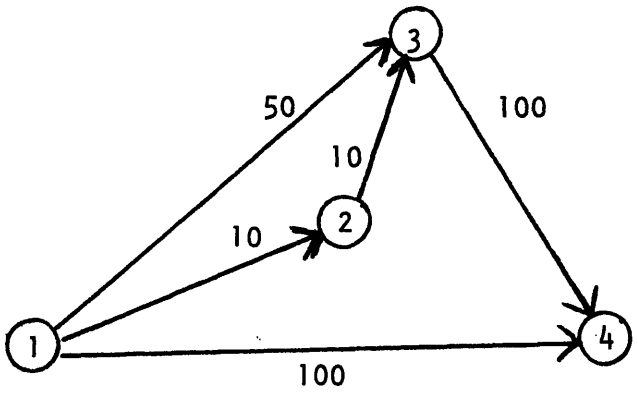
$$\leq c_{ij}$$

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 $\left\{\pi_i^2, \gamma_{ij}^2\right\}$ is a feasible solution of (3.29). By definition $\pi_s^2=0$ and $\pi_t^2=D^*$. Now $\bar{y}_{ij}=1$ implies that $\gamma_{ij}^2=0$ (See the interpretation of γ_{ij} given above). Therefore the value of (3.29) for the set $\left\{\pi_i^2, \gamma_{ij}^2\right\}$ is D^*. So the solution is optimal as well as feasible. □

Example 3.3 (N=4)



We assume s=1 and t=4.

Solid lines for arc (i,j) indicated $y_{ij}=1$.

Dotted lines for arc (i,j) indicates $y_{ij}=0$.

$$\pi_1^1 = 0 \qquad \pi_2^1 = 10 \qquad \pi_3^1 = 50 \qquad \pi_4^1 = 100$$

$$\gamma_1^1 = 0 \qquad \gamma_{13}^1 = 0 \qquad \gamma_{14}^1 = 0 \qquad \gamma_{23}^1 = 30 \qquad \gamma_{34}^1 = 0 .$$

In this instance,

$$D^* = 100 .$$

In addition,

$$\Delta_2 = -10 \qquad \Delta_3 = 0 \qquad \pi_1^2 = 0 \qquad \pi_2^2 = \min(-10,10) = -10$$

$$\gamma_{12}^2 = 0 \qquad \gamma_{13}^2 = 0 \qquad \gamma_{14}^2 = 0 \qquad \gamma_{23}^2 = 0$$

$$\pi_3^2 = \min(0,50) = 0 \qquad \pi_4^2 = 100 \qquad \gamma_{34}^2 = 0.$$

The cut defined by the first set of dual variables is: $v \geq 100 - 30y_{23}$. The cut defined by the new procedure is: $v \geq 100$. So the new cut is preferred. Notice how the new procedure has improved upon the standard cut. Solving the shortest path problem by forming a shortest path tree routed at node 1, we might believe that arc (2,3) would lead to a shorter path. Looking ahead, though, from node 3 to node 4 by computing the shortest path distances from every node to node 4, we "see" the high cost arc (3,4). In general, the new procedure looks for additional costs that must be incurred "further down the road".

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, though usually not pareto-optimal, represents a substantial strengthening of the standard cut. We also feel that any pareto-optimal cut that dominates the new cut will produce only marginal improvements. In fact, one way to improve the new cut is to proceed as in the development of the strong p-median cuts. That is, calculate the increase of the flow routing cost which results from closing an arc.

3.5 COMPUTATIONAL EXPERIENCE

In this section we present some computational results for Benders decomposition equipped with the strong cut methodology. Our tests involved two types of network design problems: the p-median location problem discussed in sections 3.4.1-3.4.3 and the fixed charge network design problem described in section 3.4.4.

3.5.1 The p-median Problem

For the p-median tests, an all-FORTRAN implementation of Benders decomposition was programmed on the PRIME minicomputer system at the Massachusetts Institute of Technology's Sloan School. To generate initial feasible integer solutions for the first iteration of Benders procedure, we applied a very effective heuristic procedure described by Cornuejols, Fisher and Nemhauser [21]. The Benders continuous subproblems were solved three different ways in order to compare the convergence properties of the type A, B and C cuts (as described in the previous section). The master

problems for these tests were solved via an exhaustive enumeration program. Although this expedient limited the size of our computational tests, it did permit us to readily test our conjectures about reducing the number of iterations.

The first p-median test problem was a 10 node network taken from Garfinkel, Neebe, and Rao [41]. Tables 3.1 and 3.2 show the computational results for locating 3 and 6 medians on the 10 node network. For the 3 median problem, Benders procedure with the type B and C cuts converged to the optimal solution in 10 iterations whereas the "standard" type cut required almost three times as many iteractions. For the 6-median problem, the results emphatically show the superiority of the strong cuts over the standard cut. The type B and type C cuts (which for this problem are identical) converged very rapidly. The standard cut exhibited a pronounced "tailing effect."

The difference in computation times for computing the three types of cuts, when compared with the time for solving the master integer programs, is believed to be negligible, although the lack of timing facilities did not allow us to make precise measurements.

Notice that in all of the p-median test problems considered, the initial solution found by the heuristic algorithm was optimal. The computational tests of Benders procedure are still meaningful, though. First, the heuristic algorithm will not always generate an optimal solution (see [21] for error bounds). Second, as a general rule, verifying whether or not a given solution to an integer program is optimal can be as hard as solving the problem from scratch.

ITERATION NUMBER	STANDARD CUT* RATIO	TYPE B CUT RATIO	TYPE C CUT RATIO
1	.39	.69	.69
2	.47	.78	.78
3	.58	.89	.89
4	.67	.92	.94
5	.69	.92	.94
6	.69	.94	.97
7	.72	.97	.97
8	.75	.97	.97
9	.81	.97	.97
10	.81	1.00	1.00

*Converges to 1.00 at iteration number 27
Optimal Solution = Initial Solution = 36
Ratio = Lower Bound/Best Upper Bound
TABLE 3.1 10 NODE, 3 MEDIAN TEST PROBLEM

ITERATION NUMBER	STANDARD CUT* RATIO	TYPE B & C CUT RATIO
1	.00	.83
2	.17	.92
3	.42	1.00
4	.42	
5	.50	
6	.50	
7	.58	
8	.58	
9	.58	
10	.67	

*Ratio = 0.92 at iteration number 26
Optimal Solution = Initial Solution = 12
Ratio = Lower Bound/Best Upper Bound
TABLE 3.2 10 NODE, 6 MEDIAN TEST PROBLEM

The computational experience for this 10 node problem conforms with our belief that the strong cuts will be clearly preferred to the standard cut whenever the fraction

$$F = \frac{\text{number of medians}}{\text{number of nodes}}$$

is relatively large, say greater than 1/4. When this ratio is low, there quite likely is insufficient interaction between the medians for the penalty considerations of the strong cuts to be meaningful.

Next we tested a network of 33 nodes taken from Karg and Thompson [64]. Table 3.3 and Table 3.4 display results for the location of 2 medians and 4 medians on the network. For the case of 2 medians there is not much difference among the three cuts although all of them performed quite well.

For the 4-median problem, the strong cuts performed somewhat better than the standard Benders cut. At the end of 10 iterations, the type C cut had performed about 10% better than the standard cut. The difference in the performance of the strong cuts is less dramatic for the 33 node tests than for the 10 node network tests. This is probably due to the relatively small ratio of medians to nodes for the 33 node network tests.

Further computer tests were suspended due to the excessive time required to solve the master problem by exhaustive enumeration. Computation times for the above experiments were on the order of .1 to 3 minutes of minicomputer time for each Benders iteration.

ITERATION NUMBER	STANDARD CUT RATIO	TYPE B CUT RATIO	TYPE C CUT RATIO
1	.72	.75	.80
2	.76	.78	.83
3	.84	.86	.86
4	.88	.90	.90
5	.90	.93	.92
6	.90	.94	.93
7	.92	.94	.97
8	.94	.96	.99
9	.96	.98	.99
10	.97	.99	.99

Optimal Solution = Initial Solution = 17,474

Ratio = Lower Bound/Best Upper Bound

TABLE 3.3 33 NODE, 2 MEDIAN TEST PROBLEM

ITERATION NUMBER	STANDARD CUT RATIO	TYPE B CUT RATIO	TYPE C CUT RATIO
1	.52	.59	.66
2	.67	.73	.79
3	.71	.79	.83
4	.76	.84	.84
5	.78	.85	.85
6	.79	.85	.87
7	.80	.87	.88
8	.81	.87	.89
9	.81	.88	.90
10	.82	.88	.91

Optimal Solution = Initial Solution = 12363

Ratio = Lower Bound/Best Upper Bound

TABLE 3.4 33 NODE, 4 MEDIAN TEST PROBLEM

3.5.2 Fixed Charge Network Design

The fixed charge network design problem required a more elaborate implementation of Benders decomposition. For some small problems we again solved the master problems by exhaustive enumeration. For the larger problems we solved the master problems as linear programs via the MPSX inear programming routine. This enabled us to compute lower bounds very quickly by rounding.

We solved the Benders subproblems two different ways in order to compare the standard cut with the strong cut described in the previous section. The choice of an initial integer solution varied freatly so we will discuss this selection for each individual test problem.

Most of our test problems were adapted from related network design problems in the literature. In every case all required flows were equal to one. Unless otherwise noted, the arc construction costs are proportional to the continuous routing costs. We used a procedure given by Billheimer and Gray [14] to determine which arcs, if any, could be open or closed.

Hoang [59] gives two network examples that we adapted as test problems. Since these were small problems (7 nodes and 8 nodes), we solved the master problem by exhaustive enumeration. Tables 3.5 and 3.6 give the results of applying Benders decomposition to these problems. The strong cuts performed significantly better than the standard cuts and were able to converge towards an optimal solution very quickly. Notice that the choice of the initial integer solution (either near-optimal or not) did not seem to effect the convergence of the strong cuts.

ITERATION NUMBER	STANDARD CUTS UPPER BOUND	RATIO	STRONG CUTS* UPPER BOUND	RATIO
1	1737	.79	1737/2112	.90/.60
2	>1737	.79	>1737/1877	.95/.87
3	>1737	.80	>1737/1738	.95/.96
4	>1737	.80	>1737/1758	.96/.97
5	>1737	.80	>1737	.98
6	>1737	.81	>1737	.98
7	>1737	.82	>1737	.98
8	>1737	.83	>1737	.98

*Second numbers are 4 iterations from a different starting solution.

TABLE 3.5 7 NODE, 15 ARC NETWORK DESIGN PROBLEM (4 ARCS FIXED OPEN)

ITERATION NUMBER	STANDARD CUTS UPPER BOUND	RATIO	STRONG CUTS UPPER BOUND	RATIO
1	3021.0	.00	3021.0	.55
2	2433.5	.74	2433.5	.90
3	2770.5	.75	2560.5	.92
4	2900.0	.75	2581.5	.92
5	2594.0	.76	2569.5	.93
6	2647.0	.76	2421.0	.96
7	2615.5	.77	2402.5	.97
8	2640.5	.77	2421.5	.97
9	2525.5	.77	2368.5	.98
10	2478.0	.77	2374.5	.99

TABLE 3.6 8 NODE, 18 ARC NETWORK DESIGN PROBLEM (5 ARCS FIXED OPEN)

ITERATION NUMBER	STANDARD CUTS UPPER BOUND	RATIO	STRONG CUTS* UPPER BOUND	RATIO
1	115063	.00	115063	.00
2	100575	.41	100575	.46
3	107910	.41	97212	.48
4	103427	.42	82746	.57
5	95198	.45	92083	.58
6	100706	.45	90813	.58
7	89486	.48	94261	.59
8	94209	.48	76341	.67
9	79096	.54	74414	.68
10	67736		76688	
1	67736	.00	74414	.223
2	100575	.60	78857	.648
3	115373	.61	74124	.710
4	107349	.61	92014	.717
5	100584	.62	63469	.841
6	88299	.63	75722	.85
7	84342	.63	86112	.85
8	76824	.63	73249	.85
9	74874	.63	73091	.85
10	70732	.63	80881	.85
11	69100		66377	

Rounding Threshold = .001

TABLE 3.7 10 NODE, 45 ARC NETWORK DESIGN PROBLEM (9 ARCS FIXED OPEN)

ITERATION NUMBER	STANDARD CUTS UPPER BOUND	RATIO	STRONG CUTS UPPER BOUND	RATIO
1	51922	.00	51922	.49
2	61281	.80	-	.49
3	-	.80	-	.50
4	-	.80	-	.50
5	-	.80	72172	.51
6	-	.80	54802	.86
7	-	.80	76270	.88
8	-	.80	50919	.89
9	-	.80	54927	.89
10	-	.80	50851	.90
11	-		55387	

Rounding Threshold = .05 - Designates an Infeasible Subproblem

TABLE 3.8 10 NODE, 45 ARC NETWORK DESIGN PROBLEM (5 ARCS FIXED OPEN)

ITERATION NUMBER	STANDARD CUTS UPPER BOUND	RATIO	STRONG CUTS UPPER BOUND	RATIO
1	76348	0	76348	.51
2	107242	0	107242	.53
3	143000	.60	143000	.65
4	212652	.60	187069	.66
5	19002	.60	198055	.67
6	193393	.60	166226	.67
7	193624	.60	159241	.67
8	174928	.60	167014	.67
9	187512	.60	137205	.67
10	167334	.60	144818	.67
11	167057		160918	

Rounding Threshold = .001 3 INITIAL CUTS ADDED TO MASTER PROBLEM

TABLE 3.9 33 NODE, 132 ARC NETWORK DESIGN PROBLEM (32 ARCS FIXED OPEN)

For the next set of tests we solved the master problems as linear programs and then rounding the continuous solution to obtain an integer feasible solution. (McDaniel and Devine [86] have also studied this solution strategy). After some initial tests, we decided to round to one all variables which were greater than some small number such as .001 or .05. This threshold varied from problem to problem. Note that solving the master problem as a linear program gives a weaker lower bound to the optimal solution.

Table 3.8 gives the results of a 10 node-45 arc test problem obtained by modifying an example reported by Boyce et al. [15]. The strong cut again performs well relative to the standard cut.

Next we modified this test problem by changing the arc construction costs so that they were inversely proportional to their respective routing costs instead of proportional to them. This was done to increase the difficulty of the problem. We started our modified Benders procedure with a deliberately poor initial solution. After 10 iterations we restarted the procedure with the best feasible solution generated. Table 3.7 gives the results of these tests. Although the bounds are not as tight as those for other problems, the strong cuts are clearly superior to the standard cuts. Also for these tests it appears that Benders algorithm with strong cuts is much more efficient when a good initial solution is provided.

For all of our 10 node-45 arc tests, Benders procedure required about 1.1 seconds of IBM 370/168 computer time for each iteration. Since our present interface with the MPSX linear programming routine (using the

MPSX READCOM facility) is known to be very inefficient, we would expect much faster computation times if a more suitable implementation were available.

For our final test problem, we modified a network given by Karg and Thompson [64] to create a 33 node-132 arc test problem. Three initial cuts were specified in the master problem. The results given in table 3.9 indicate that neither type of cut performed particularly well. However, the strong cut still outperformed (by about 10%) the standard cut.

For this test problem each Benders iteration required about 7 seconds of IBM 370/168 computer time.

In summary, our computational tests indicate that the strong cuts performed noticeably better than the standard cuts on both p-median and network design problems. The strong cuts attained a ratio of upper bound to lower bound which was 10-33% better than the standard cuts. In addition, the network design tests indicate that the strong cuts help find feasible solutions which are slightly better than those determined by the standard cuts.

In our experimentation Benders decomposition was not as powerful for the network design problem as for the p-median problem, probably due to the more complex structure of the network design problem. Further work is still required in order to utilize the full potential of the algorithm for these applications. For example, strong cuts could be generated by considering second shortest paths. Or, strong cuts might be used in conjunction with some of the other techniques described in section 3.3.

CHAPTER IV

A MODEL SELECTION CRITERIA FOR BENDERS DECOMPOSITION

4.1 Introduction

Selecting the "proper" model formulation is another important factor that effects the computational performance of Benders decomposition applied to network design and other mixed integer programming models. This chapter discusses a criteria for evaluating "different" model formulations of the same mixed integer programming problem in the context of Benders decomposition.

Many network optimization problems have several "natural" mixed integer formulations. For example, as we noted in chapter III, the facility location problem and the fixed charge network design problem can both be stated in several possible ways as mixed integer programs. We demonstrate in this chapter that some of these formulations are to be preferred to others.

Geoffrion and Graves [49] in their study of industrial distribution planning found that proper model formulation was a crucial factor in their successful use of Benders decomposition. They stated that intelligent model formulation is an aspect of Benders decomposition not properly (or fully) understood and that deserves further study.

Various authors have studied alternative mathematical formulations in other areas of combinatorial optimization. Cornuejols, Fisher and

Nemhauser [21] and Geoffrion and McBride [50] provide theoretical insight and computational experience concerning the role of model formulation in Lagrangian relaxation. Davis and Ray [24], Beale and Tomlin [8], and Williams [123], in the context of linear programming relaxation for branch and bound, show that proper model formulation can greatly improve the computational efficiency of this procedure.

The next section of this chapter gives a small example illustrating the importance of proper model selection for Benders decomposition. The third section gives some results that allow us to compare the effectiveness of various mixed integer programming formulations in the context of Benders decomposition. The fourth section discusses the use of our model selection criteria for network optimization and other more general problems.

4.2 An Example

This section presents an example concerning the role of model selection for Benders decomposition applied to the p-median facility location problem. Recall from chapter III that the p-median problem can be formulated as:

$$(P_A) \qquad \text{Minimize} : \quad \sum_{i=1}^{N} \sum_{j=1}^{N} d_{ij} x_{ij}$$

$$\text{subject to:} \quad \sum_{i=1}^{N} x_{ij} = 1 \qquad \forall_j \qquad (4.1)$$

$$x_{ij} \leq y_i \qquad \forall (i,j) \qquad (4.2)$$

$$\sum_{i=1}^{N} y_i = P$$

$$x_{ij} \geq 0 \quad \text{and } y_i \text{ integer} \quad \forall(i,j). \tag{4.3}$$

N is the number of nodes in the problem and P is the number of facilities to be located. y_i indicated whether a facility is located at node i and x_{ij} indicates whether customer j is serviced at node i.

As we noted in chapter III, an equivalent formulation is:

(P_B) Minimize $$\sum_{i=1}^{N} \sum_{j=1}^{N} d_{ij} x_{ij}$$

subject to: (4.1), (4.3) and

$$\sum_{j=1}^{N} x_{ij} \leq N y_i \tag{4.2'}$$

Note that (4.2') represents an aggregation of the constraints in (4.2). So that although P_A and P_B are equivalent mathematical descriptions, if we relax the integrality constraint on the y_i, then the feasible region for P_B will be a proper subset of the feasible region for P_A.

Let us examine the following p-median problem represented in figure 4.1:

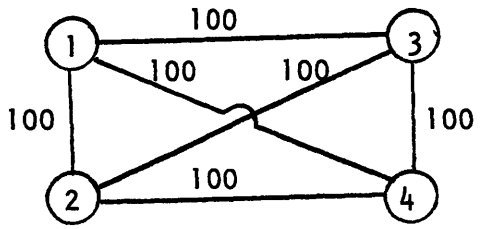


Figure 4.1 p-median Example

N is equal to 4, P is equal to 2 and all d_{ij} are 100.

The application of Benders decomposition to this example formulated as P_B yields the following set of Benders cuts:

$$z \geq 200 - 400y_1 - 400y_2 + 0y_3 + 0y_4$$
$$z \geq 200 - 400y_1 + 0y_2 - 400y_3 + 0y_4$$
$$z \geq 200 - 400y_1 + 0y_2 + 0y_3 - 400y_4$$
$$z \geq 200 + 0y_1 - 400y_2 - 400y_3 + 0y_4$$
$$z \geq 200 + 0y_1 - 400y_2 + 0y_3 - 400y_4$$
$$z \geq 200 + 0y_1 + 0y_2 - 400y_3 - 400y_4$$

This set of cuts has the property that every single one must be generated in order for Benders algorithm to converge.

Recall from chapter III that applying Benders decomposition to our example formulated as P_A yields several different sets of cuts. The first set, consisting of type A cuts, is identical to the above set except that all coefficients of -400 become -200. So all six cuts are again necessary for convergence. In contrast, generating a set of type B cuts, requires the single cut:

$$z \geq 400 - 100y_1 - 100y_2 - 100y_3 - 100y_4 .$$

We can generalize this example in the following way: let $P = \frac{N}{2}$ and let $d_{ij} = 100$ for all $i \neq j$ and $d_{ij} = 0$ for all $i=j$. With this class of examples, we have problems where the P_B formulation requires $\binom{N}{N/2}$ cuts, an exponential number of cuts for Benders algorithm to converge. For these same problems, the P_A formulation in every case requires only one Benders

cut for convergence! This dramatically illustrates the importance of intelligent model formulation for Benders decomposition.

4.3 A Criteria for Comparing Model Formulations for Benders Decomposition

First, this section presents a formal framework for comparing model formulations for Benders decomposition. This framework is then utilized to prove our main results.

Suppose we have two mixed integer programs P_1 and P_2 which are represented as:

$$(P_1) \qquad \underset{y \in Y}{\text{minimize}} \; P_1(y) \quad \text{where } P_1(y) = \begin{array}{ll} \text{minimum} & cx + dy \\ \text{subject to:} & Ax + By = b \\ & x \geq 0 \end{array}$$

and

$$(P_2) \qquad \underset{y \in Y}{\text{minimize}} \; P_2(y) \quad \text{where } P_2(y) = \begin{array}{ll} \text{minimum} & hw + dy \\ \text{subject to:} & Dw + Gy = g \\ & w \geq 0 \end{array}$$

x, w, and y are column vectors of problem variables; b and g are column vectors; c,d, and h are row vectors; A,B,D, and G are appropriately dimensioned matrices. The set Y is a set of integer valued vectors which encaptures the integer constraints of the problem. We assume that the set Y is finite.

We will say that P_1 and P_2 are equivalent mixed integer programming representations of the same problem if

$$P_1(y) = P_2(y) \qquad \text{for all } y \in Y.$$

That is, the two models have the same integer variables and may have different continuous variables and constraints, but always give the same objective function value for any feasible assignment of the integer variables.

In the context of Benders decomposition, another possible interpretation of this equivalence is that $P_1(y)$ and $P_2(y)$ represent the linear programming subproblems when Benders decomposition is applied to P_1 and P_2. So the two models are equivalent if their respective Benders subproblems always have the same optimal value.

We evaluate these two models by comparing the cuts generated from the application of Benders decomposition to these models. Following the derivation of Benders decomposition given in chapter III, we can rewrite P_1 and P_2 as[†]

$$\begin{aligned} &\text{Minimize} && z \\ &\text{subject to:} && z \geq \Pi^j(b-By) + dy \qquad j \in J \\ & && y \in Y \end{aligned}$$

where $\left\{\Pi^j\right\}_{j \in J}$ is the set of extreme points of the polyhedron $\Pi A \leq C$; and

---

[†]As we did before in chapter III, assume that the linear programming subproblems $P_1(y)$ and $P_2(y)$ are feasible and have optimal solutions for all $y \in Y$. These constraints can be relaxed, but with added complications that cloud our main development.

$$\begin{aligned} &\text{Minimize} && z \\ &\text{subject to:} && z \geq \gamma^k(g-Gy) + dy \qquad k\epsilon K \\ & && y \in Y \end{aligned}$$

where $\{\gamma^k\}_{k\epsilon K}$ is the set of extreme points of the polyhedron $\gamma D \leq h$.

The inequalities $z \geq \Pi^j(b-By) + dy$ and $z \geq \gamma^k(g-Ey) + dy$ will be referred to as the Benders cuts for P_1 and P_2 respectively. To compare equivalent model formulations, we adapt the concept of a pareto optimal cut, introduced in chapter III, by saying that a Benders cut (or constraint)

$$z \geq \Pi^j(b-By) + dy$$

for P_1 dominates a Benders cut

$$z \geq \gamma^k(g-Gy) + dy$$

for P_2 if

$$\Pi^j(b-By) + dy \geq \gamma^k(g-Gy) + dy$$

for all $y\epsilon Y$ with a strict inequality for at least one point $y\epsilon Y$.

A cut $z \geq \gamma^k(g-Gy) + dy$ will be called <u>unique</u> with respect to the formulation P_1 if there is no cut belonging for P_1 that is equal to it (in the sense that two cuts are equal if their right-hand sides are equal for all $y\epsilon Y$) or dominates it.

A formulation P_2 is <u>superior</u> to an equivalent formulation P_1 if P_2 has at least one Benders cut that is unique with respect to P_1, but P_1 does not have any cuts that are unique with respect to P_2.

In a very loose sense, P_2 is superior to P_1 if they are equivalent formulations and the set of Benders cut for P_1 is a proper subset of the Benders cuts for P_2.

With these definitions we can now prove serveral properties concerning model formulation and the strength of Benders cuts.

Lemma 4.1 Let P_1 and P_2 be equivalent formulations of a mixed integer programming problem. P_2 has a Benders cut that is unique with respect to P_1 if and only if there exists a $y^o \epsilon Y^c$ such that $P_2(y^o) > P_1(y^o)$, where Y^c denotes the convex hull of the set Y.

Proof: ($\Longrightarrow$) Let $z \geq \gamma^*(g-Gy) + dy$ be a Benders cut that is unique with respect to P_1. Since we are assuming that the set Y is finite, this implies:

$$\max_{j\epsilon J} \left[\min_{y\epsilon Y} \Pi^j(b-By) + dy - \gamma^*(g-Gy) + dy \right] < 0.$$

Now observe that the above inequality still holds if we replace the set Y by Y^c and the set $\left\{\Pi^j\right\}_{j\epsilon J}$ by $\left\{\Pi : \Pi A \leq C\right\}$. Using linear programming duality theory we can reverse the order of the max and min operation to get

$$\min_{y\epsilon Y^c} \left[\max_{\Pi A \leq C} \Pi(b-By) + dy - \gamma^*(g-Gy) + dy \right] < 0.$$

Linear programming duality theory allows us to rewrite the above expression as

$$\begin{aligned} \text{Min} \quad & cx + dy - \left[\gamma^*(g-Gy) + dy\right] < 0 \\ \text{subject:} \quad & Ax + By = b \\ & x \geq 0,\ y \in Y^c \end{aligned}$$

This implies these exists $y^o \varepsilon Y^c$ such that

$$\begin{aligned} \text{Min} \quad & cx + dy^o = P_1(y^o) < \gamma^*(g-Gy^o) + dy^o. \\ \text{subject to:} \quad & Ax = b-By^o \\ & x \geq 0 \end{aligned}$$

Another application of linear programming duality theory gives us:

$$P_1(y^o) < \gamma^*(g-Gy^o) + dy^o \leq \begin{aligned} \text{Min} \quad & hw + dy^o \\ \text{subject to:} \quad & Dw = g-Gy^o \\ & w \geq 0 \end{aligned}$$

or

$$P_1(y^o) < P_2(y^o) .$$

($\Longleftarrow$) The reverse implication has essentially the same proof with all the steps reversed. Explicit details will not be given here. □

This lemma leads to the following theorem about preferred formulations:

Theorem 4.1 Let P_1 and P_2 be equivalent formulations of a mixed integer programming problem. P_2 is superior to P_1 if and only if $P_2(y) \geq P_1(y)$ for all $y \varepsilon Y^c$ with a strict inequality for at least one $y \varepsilon Y^c$.

Proof: ($\Leftarrow$) If $P_2(y) \geq P_1(y)$ for all $y \in Y^c$, Lemma 4.1 says that P_1 does not have any Benders cuts that are unique with respect to P_2. Similarly, P_2 has a Benders cut that is unique with respect to P_1 because there is a $y^o \in Y^c$ such that $P_2(y^o) > P_1(y^o)$. So P_2 satisfies the definition of being superior to P_1.

($\Rightarrow$) If P_2 is superior to P_1, P_1, by definition of superior, does not have any cuts that are unique with respect to P_2. Lemma 4.1 then tells us that $P_2(y) \geq P_1(y)$ for all $y \in Y^c$. The definition of superior also states that P_2 has a cut that is unique with respect to P_1 and using lemma 4.1 we can say that there exists a $y^o \in Y^c$ such that $P_2(y^o) > P_1(y^o)$. □

Theorem 4.1 has an interesting interpretation. Let the relaxed primal problem for P_1 be defined as:

$$\begin{aligned} &\text{minimize} && cx + dy \\ &\text{subject to:} && Ax + By = b \\ & && x \geq 0, y \in Y^c \end{aligned}$$

Note that the only difference between P_1 and the above problem is that the set Y has been replaced by its convex hull Y^c.

Theorem 4.1 states that for a formulation of a mixed integer programming problem, the smallest possible feasible region (or the "tightest" possible constraint set) for its relaxed primal problem is preferred for generating strong Benders cuts. For any formulation P, a smaller feasible region for its relaxed primal problem will result in large values of the function P(y) which lemma 4.1 and theorem 4.1 indicates is desirable.

Consequently, for any mixed integer programming formulation, the convex hull of its feasible region will be the model formulation which is "optimal" in terms of generating Benders cuts since it has a modified primal problem whose feasible region is the smallest.

Another property of the convex hull formulation of a problem is that when Benders algorithm is applied to it, only one cut is necessary for it to converge. However, determining the correct initial starting point in order to generate this cut could be difficult.

4.4 Using the Benders Decomposition Model Selection Criteria

Although we have shown that a reduced feasible region for the modified primal problem of a formulation is desirable, there are other issues which must be considered in selecting a model for use with Benders decomposition.

First, there remains the difficulty of constructing alternative models for mixed integer programming problems. The convex hull formulation of a problem is optimal for generating strong Benders cuts but, in general, it will be very difficult to build such a model. There is no efficient procedure known for generating the constraints representing the convex hull of a set of points. Efficient methods for generating alternative models appears to be an area for future research.

For network optimization problems, the situation is more encouraging in that there are usually several evident "natural" formulations. The facility location problem and the fixed charge design problem given in chapter III, the multicommodity distribution system problem solved by Geoffrion and Graves [49], and the capacitated plant location problem

described by Guignard and Spielberg [55], are all network examples that have several easily derived formulations. For these problems, since the alternative formulations usually have the same problem variables, we can compare them by inspecting the size of the feasible region for their respective modified primal problems.

The p-median example discussed in section 4.2 has 2 formulations P_A and P_B. They differ only in that P_A has constraints of the form

$$x_{ij} \leq y_i \qquad \forall (i,j) \qquad (4.2)$$

wheras P_B has constraints of the form

$$\sum_{j=1}^{4} x_{ij} \leq 4y_i \qquad \forall i \qquad (4.2')$$

Since (4.2') is an aggregation of the constraints in (4.2), the feasible region for P_A's modified primal problem is no larger than the one for P_B. So $P_A(y) \geq P_B(y)$ for all $y \in Y^C$. A straightforward computation shows that $P_A(y^o) = 200 > P_B(y^o) = 0$ for $y^o = \left(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right)$. So the formulation P_A is superior to P_B for this example.

Due to the comparatively simple constraint sets of network problem formulations, it may also be possible to derive additional constraints from the current ones. In such a situation these new constraints could be evaluated by testing if they reduce the size of the feasible region for the modified primal problem.

Another issue that should be considered is the difficulty of solving the Benders (linear programming) subproblems. Adding constraints

to a formulation strengthens the Benders cuts that can be derived, but also complicates the solution of the linear subproblems. So there is a trade-off between the quality of Benders cuts available and the time needed to solve the Benders subproblems.

Finally, a related issue is that adding constraints to a formulation can cause the linear programming subproblems to become degenerate since we are adding constraints to a linear program while keeping the number of variables constant. Thus there may be a choice as to which cut to generate at each iteration of Benders algorithm.

So by "tightening" the formulation of a problem we can get stronger Benders cuts, but these stronger cuts may have to be distinguished from other weaker cuts. The methodology described in chapter III should be useful in such a situation.

CHAPTER V

A MATHEMATICAL ANALYSIS OF NETWORK DESIGN PROBLEM HEURISTICS

5.1 Introduction

This chapter concerns an issue related to the computation performance of Benders decomposition for network design problems, the generation of near-optimal solutions by heuristic methods. As Mevert [87] and Geoffrion and Graves [49] have noted, the selection of a good starting point for Benders decomposition is an important determinant of computational success. The tests results given in chapter III for the fixed charge network design problem also demonstrate this type of behavior for Benders decomposition. So the selection of an appropriate initial solution for Benders decomposition should greatly accelerate its convergence. Heuristic techniques are frequently the most efficient ways of generating good initial solutions for network design problems. Also these approximate techniques can be applied to large-scale network models that are too complicated to solve with exact solution techniques such as Benders decomposition.

This chapter presents results concerning the use of heuristic methods for generating solutions to some network design problems, in particular, the "optimal" network problem [114,27] and related versions of it. Recall from chapter II that the optimal network problem consists of selecting a subset of arcs, subject to a budget constraints, so that

the routing costs for required flows in the network is minimized; an equivalent mixed integer programming formulation of this problem is:

$$\text{Minimize} \quad \sum_{(i,j)\in A} \sum_{(k,\ell)\in(D\times D)} c_{ij}\, x_{ij}^{k\ell}$$

$$\text{subject to} \quad \sum_{j} x_{ij}^{k\ell} - \sum_{q} x_{qi}^{k\ell} = \begin{cases} r_{k\ell} & \text{if } i=k \\ -r_{k\ell} & \text{if } i=\ell \\ 0 & \text{otherwise} \end{cases}$$

$$x_{ij}^{k\ell} \leq r_{k\ell}\, y_{ij}$$

$$\sum_{(i,j)\in A} d_{ij}\, y_{ij} \leq B$$

$$x_{ij}^{k\ell} \geq 0 \qquad (i,j)\in A \text{ and } (k,\ell)\in(D\times D)$$

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

where D is the set of nodes, A is the set possible arcs (undirected), and $x_{ij}^{k\ell}$ is the amount of commodity (k,ℓ) routed on arc (i,j). $r_{k\ell}$ is the amount of commodity (k,ℓ) that must be routed. d_{ij} is the construction cost of arc (i,j) and c_{ij} is the per unit routing cost of arc (i,j). All data d_{ij} and c_{ij} are assumed to be nonnegative. y_{ij} is a binary variable indicating whether or not arc (i,j) is to be constructed. B is the construction budget.

The optimal network model has potential used in designing air, rail or highway transportation networks. Although such systems are usually much more complex than the above problem, this model could be useful in

screening network configurations for more detailed study.

Our literature survey in chapter II indicated that there is both theoretical and empirical evidence suggesting that optimal network design is a very difficult optimization problem. Johnson et al [63] have shown that the optimal network problem is NP-complete which means that there is probably no efficient method for solving problems of this type. Computational studies by several authors [15,27,59] using branch and bound techniques have shown that for optimal network problems with more than about 75 arcs, solution times are prohibitive. So sub-optimal heuristic methods appear to be the only methods available for generating solutions to large-scale network design models. An important question that arises in using heuristic techniques is the accuracy of the answers generated.

Although a great deal of effort has been applied to designing heuristic algorithms for various network problems, comparitively little is known about their behavior in approximating the optimal solution. The most commonly used heuristic evaluation technique is to conduct empirical tests by applying the heuristic to a set of "typical" problems and then assessing the results of the sub-optimal procedure. However, it is difficult to decide what constitutes a set of "typical" problems and frequently the set is chosen through arbitrary means.

Another method of evaluating a heuristic algorithm is to analyze its "average case" performance. For this technique a probability measure is imposed on the set of possible input problems to the algorithm and the expected value of the ratio of heuristic solution to optimal solution value is computed. Although this method has many advantages compared to the empirical testing technique, there are several drawbacks to its use. The

first disadvantage is that it may be very difficult to find a probability measure that is appropriate for the set of problems that the algorithm will be applied to. For example, in integer programming problems, researchers have found that real-world problems are generally "easier" to solve than "randomly" generated test problems.

Another drawback to the average case method is that analyzing the expected behavior of an algorithm is usually very difficult. Consequently, there has not been much work in this area. See Karp [67] for one of the few examples of average case analysis of heuristics.

Finally, another method of analyzing a heuristic is to utilize a worst-case performance measure, that is, to compute the maximum possible percentage deviation from the optimal solution when using the heuristic. This type of analysis is conservative in that only the worst possible error is computed. It has the advantage of being more analytically tractable to perform than average-case analysis. Also the results of a worst-case evaluation are applicable regardless of the underlying probability distribution of the input problems. However, there is no conclusive evidence as to whether the worst-case error bound is an appropriate method for comparing the accuracy of various heuristics. See Rosenbrantz et al [109] and Johnson [62] for some considerations of this last point.

Many researchers have analyzed heuristics for combinatorial problems in terms of their worst-case error performance. See Garey and Johnson [40] for a survey of these results.

In this chapter we present some worst-case analyses of heuristics for the optimal network problem. The next section contains a review of some past work in designing heuristics for the optimal network problem.

Also some examples are given which demonstrate the worst-case behavior for some of these procedures. The third section contains this chapter's main results which concern the computational complexity of designing heuristics with a given accuracy. These results indicate that all polynomial-time heuristics for the optimal network problem probably have poor worst-case error bounds. The fourth section describes a particular heuristic algorithm whose worst-case error ratio for a restricted version of the optimal network problem is bounded by a constant that does not depend on the characteristics of the input problem.

We should note that most of the previous work in this area (see [15,27,59]) dealt with a restricted version of the optimal network design problem where all required flows $r_{k\ell}$ were one and every arc routing cost c_{ij} was equal to its construction cost d_{ij}. For the rest of this chapter, unless otherwise noted, we assume that all required flows $r_{k\ell}$ are one but that an arc routing cost may be different from its construction cost.

Finally, for technical purposes only and without any loss of generality, we assume that all c_{ij} and d_{ij} are integer valued and that all problems under consideration have an optimal solution greater than zero.

5.2 Previous Work in Optimal Network Design Heuristics

Scott [114] and Dionne and Florian [27] have presented some optimal network design heuristics which we consider here.

The first heuristic that we review is due to Dionne and Florian and was stated as follows:

(H1) 1) Construct the minimal cost spanning tree as the initial network configuration.

2) As long as the budget constraint is not violated, add to the network configuration the arc whose construction cost is the least of all arcs not yet included in the network design.

Note that if the minimal cost spanning tree is infeasible because of the construction budget constraint then the problem is infeasible.

Dionne and Florian also presented another heuristic that is a modified version of one described by Scott. It has the following description:

(H2) 0) Let M be the set of arcs in the current network design. For $k \in M$, define $Q_k(M)$ as the increase in the total routing cost if arc k is deleted from M.

1) Initialize M so it contains all arcs in the network.

2) Find k^* such that

$$L_{k^*}(M) = \frac{Q_{k^*}(M)}{d_{k^*}} = \underset{k \in M}{\text{MIN}} \frac{Q_k(M)}{d_k} \quad ,$$

where d_k is the construction cost of arc k.

If $L_{k^*}(M) = \infty$, then the removal of any link will disconnect the network and computation should be restarted using heuristic H1. Otherwise, delete arc (k^*) from M and continue with step 3.

3) If $\sum_{k\epsilon M} d_k > B$, i.e. the current network exceeds the construction budget, go to step 2; otherwise continue with step 4.

4) If $B - \sum_{k\epsilon M} d_k \geq 0$, then introduce as many arcs as pos possible so that the routing cost decrease is maximized and the budget constraint is satisfied. END

This second heuristic is related to the "greedy" heuristic that has been studied by Nemhauser, Wolsey and others [21,92,]. The quantity $L_k(M)$ can be considered as the normalized "loss" due to deleting arc k. At each iteration we delete the arc whose loss is the minimum of all arcs; the process continues until a feasible solution is reached. Comparing this procedure to the "greedy" heuristic, we can see that they are very similar.

Dionne and Florian performed computational tests to compare both heuristics. H2 performed noticeably better than H1. In fact, for many test problems H2 was able to find the optimal solution.

We now consider the worst-case performance for these heuristics. Let us define the following terms:

$V_h(\cdot)$ = the value of the solution computed by heuristic h for problem $(\cdot)$.

$V(\cdot)$ = the optimal solution value for problem $(\cdot)$.

$p(N)$ = the set of optimal network problems containing N nodes.

$$R_h(N) = \underset{p\epsilon p(N)}{MAX} \frac{V_h(p)}{V(p)} .$$

$R_h(N)$ is the worst possible error ratio when heuristic h is applied to optimal network problems consisting of N nodes. The goal of our worst-case performance analysis is to compute $R_h(N)$.

We show that for both of the above heuristics, the worst-case error ratio essentially behaves as a linear function of N, the number of nodes in the network. Therefore the error ratio is unbounded as the size of the network increases.

Consider the following canonical example:

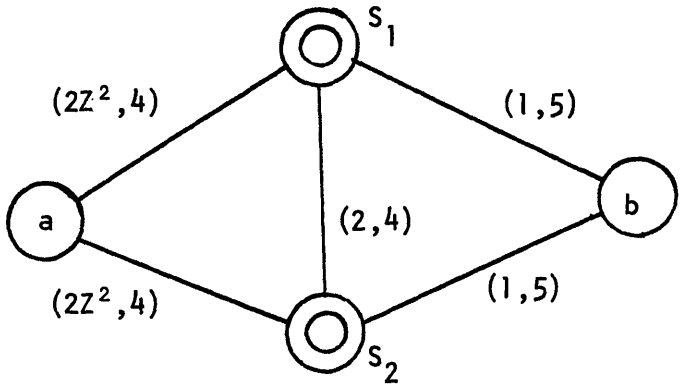


Figure 5.1 Optimal Network Problem Example for Heuristic H1.

In figure 5.1, s_1 and s_2 both represent a subnetwork consisting of k nodes. Figure 5.2 contains a diagram of this subnetwork. Any arc connected to s_1 or s_2 is considered to be connected to the center node in the corresponding subnetwork.

The label associated with each arc in Figure 5.1 denotes the arc's routing cost and the construction cost respectively. The construction budget B is 13.

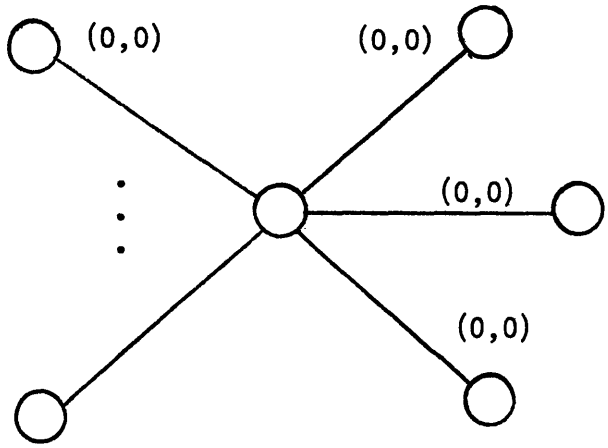


Figure 5.2 Star Network Representing a Node.

Using heuristic H2, we start with all arcs in the network. Then we drop arc (s_1,s_2). Next, we drop arc (s_1,b) or (s_2,b) the analysis is the same regardless of which are is deleted). This leaves us with the following network depicted in figure 5.3.

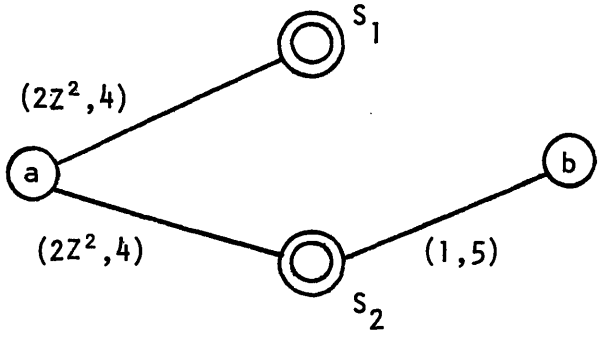


Figure 5.3 Solution Computed by Heuristic H2 for the Example.

Remembering that all required flows r_{ij} are equal to one, we compute the cost of the above solution as

$$V_{H2} = 8Z^4 + 16Z^3 + 4Z^2 + 4Z + 2$$

Figure 5.4 depicts the optimal solution to the above problem.

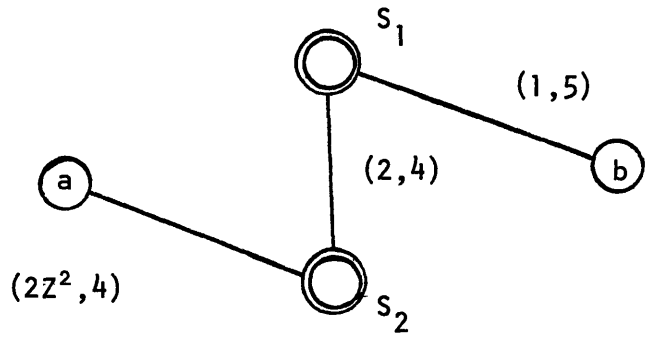


Figure 5.4 Optimal Solution for Optimal Network Example.

The optimal solution has

$$V = 8Z^3 + 8Z^2 + 12Z + 6 .$$

The total number of nodes in the network if $2Z + 2$.

$$F_{H2}(2Z+2) \geq \frac{8Z^4 + 16Z^3 + 4Z^2 + 4Z + 2}{8Z^3 + 8Z^2 + 12Z + 6}$$

$$R_{H2}(2Z+2) \geq Z \qquad \text{for } Z \geq 2$$

This implies

$$R_{H2}(N) \geq \frac{1}{2}N - 1 \qquad \text{for } N = 6,8,10,...$$

So our example shows that the worst-case error ratio for H2 must be at least essentially linear since our canonical example exhibits such behavior for an infinite number of network sizes.

Heuristic H1 behaves similarly. Consider the canonical examples represented by figure 5.5.

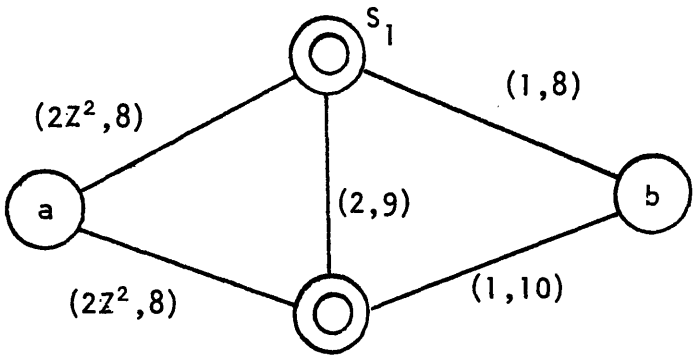


Figure 5.5 Optimal Network Problem Example for Heuristic H1.

Let the budget B be 27. An analysis that closely follows the one given above tells us that

$$R_{H1}(N) \geq \frac{1}{2} N-1 \qquad \text{for } N = 6,8,10,\ldots$$

So the worst-case error ratio for H1 must also be at least essentially linear.

The above results lead us to question if there are optimal network design heuristics whose worst case behavior is better than the ones given above. The next section gives a result which indicates that all "reasonable" heuristics must probably perform nearly as badly in terms of worst-case error margins. Also we show that the worst-error ratios for the above heuristics is no worse than a linear function of network size. So the examples given above show essentially the worst possible

behavior of heuristics H1 and H2.

5.3 Two Theorems on the Accuracy of Optimal Network Problem Heuristics

The first result that we consider concerns the class of polynomial-time heuristics for the optimal network problems, that is, the set of all optimal network design heuristics whose worst-case computation time is a polynomial function of the problem size. Usually the problem size is represented by the number of nodes N. So any heuristic whose computation time is bounded by a polynomial function of N belongs to the above class.

As we stated previously, Johnson et al [63] showed that the optimal network problem is NP-complete. Next we show that the problem of finding an optimal network design heuristic whose worst-case error ratio is less than $N^{1-\varepsilon}$, where N is the number of nodes in the network and ε is between 0 and 1, is also NP-complete. So finding a polynomial-time optimal network design heuristic that is always "close" to the optimal solution is as hard as finding a polynomial-time procedure that is always optimal. Sahni and Gonzales [111] demonstrated similar results for the traveling salesman problem (without the triangle inequality restriction), the multi-commodity network flow problem and other combinatorial problems. Garey and Johnson [40] derived a similar result for the graph-coloring problem.

Our first result can be stated in the following terms:

Definition 5.1 The approximate optimal network problem is the following: let ε be any fixed positive constant between 0 and 1, for any optimal network problem P find a solution whose value is less than or equal to $N^{1-\varepsilon}V(P)$,

where N is the number of nodes in the problem P.

Theorem 5.1 The approximate optimal network problem is NP-complete.

Proof: Since the optimal network problem belongs to NP (see [63]), the approximate optimal network problem must also belong to NP. Now we show that if the approximate problem could be solved in polynomial-time, that is, if there existed a polynomial-time heuristic h^* such that $R_{h^*}(N) \leq N^{1-\varepsilon}$, $0 < \varepsilon < 1$ and for all N, then all of the NP-complete problems could be solved in polynomial time.

Let us define a useful auxillary problem. The Steiner tree problem [65] has the following description: given a network (D,A) with node set D and arc set A and the data i) $\{d_{ij}\}_{(i,j)\in A}$, the set of arc construction costs, ii) B, the construction budget, and iii) S, a set of nodes which is a subset of D, determine if there is a subtree of the network whose construction cost is less than a given budget B and with the property that all nodes in S are connected by the subtree. Karp [65] has shown that the Steiner tree problem is NP-complete.

We next demonstrate that if the heuristic h^* defined above exists, then the Steiner tree problem could be solved in polynomial-time. It would then follow [65] that every NP-complete problem could be solved in polynomial time.

Given any Steiner tree problem, transform it into an approximate optimal network problem in the following way: replace each node in the set S by a subnetwork of the type pictured in figure 5.2. Each of these subnetworks should have N^k nodes, where N is the number of nodes in the

original Steiner tree problem and k is a constant that will be specified later. All routing and construction costs for arcs in the subnetwork should be zero.

Attach a special node T to the Steiner problem network. Every "special" arc between T and the set of nodes D has a construction cost of zero and routing cost of one. Every arc between T and a node in S, which is represented by a star network corresponding to figure 5.2, is connected to the center of the star network. All arcs originally in the Steiner problem network have zero routing cost and retain their original construction costs.

Figures 5.6-5.7 illustrate such a transformation. S' is the set (N-S). The arc labels in the original Steiner tree problem network are the arc construction costs. The arc labels in the modified optimal network problem indicate the arc routing and construction costs.

The construction budget for the optimal network problem is the same as the Steiner problem budget. As we have assumed throughout this chapter, all required flows in the optimal network problem are equal to one.

It is important to note that this transformation to create an optimal network problem from a Steiner tree problem is a polynomially time bounded procedure for any value of the parameter k.

Also note that the size M of the optimal network problem created by our transformation is at most $(N^{k+1}+1)$ nodes.

Now if one of the special arcs is utilized in the optimal network design to connect two nodes that are in S,

routing cost $\geq N^{2k}$.

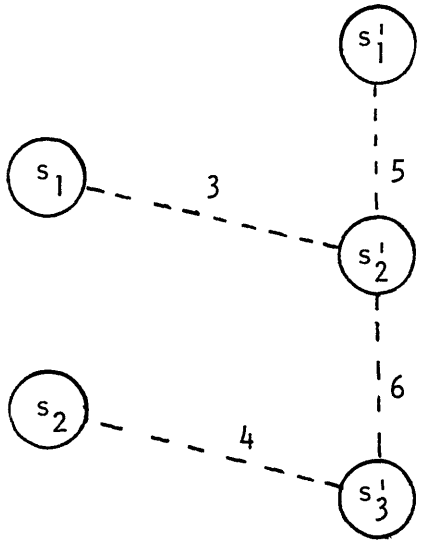


Figure 5.6 Example of a Steiner Network Problem (Before the Transformation).

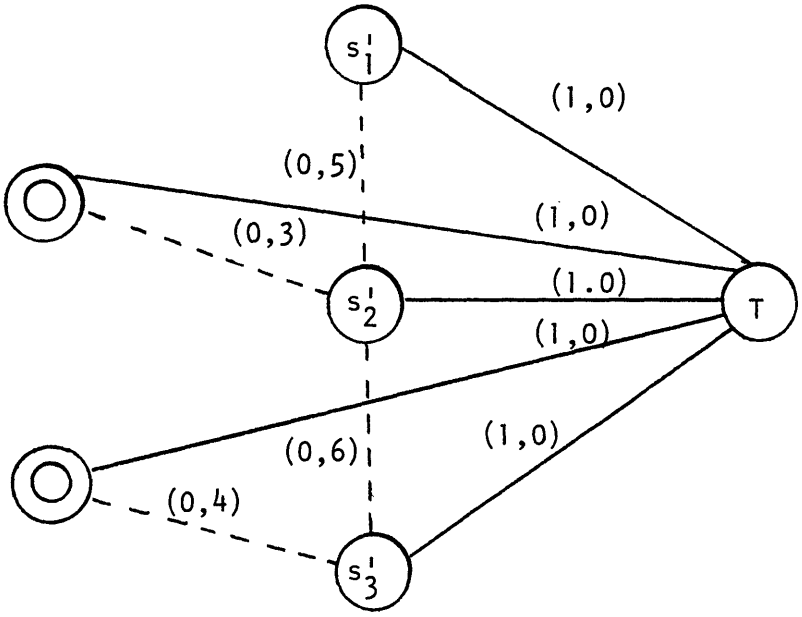


Figure 5.7 Example of a Steiner Network Problem (after the Transformation)

If all nodes in S are connected with arcs from the original Steiner tree problem,

$$\text{routing cost}^{\dagger} \leq N^{k+2}, \qquad k \geq 1 \quad \text{and} \quad N \geq 4.$$

Suppose now there is a polynomial time heuristic h^* such that for some $0 < \varepsilon < 1$

$$R_{h^*}(M) < M^{1-\varepsilon} \qquad \text{for all} \quad M \geq 0.$$

Since there exists a $k \geq 3$ such that $\frac{k-2}{k+2} \geq 1-\varepsilon$ we have

$$R_{h^*}(M) < M^{\frac{k-2}{k+2}} \leq M^{1-\varepsilon} \qquad \text{for some } k \geq 3.$$

Then for our transformed Steiner problems with $M \leq N^{k+1} + 1$, where N is the number of nodes in the original Steiner problem, we have

$$R_{h^*}(M) < M^{\frac{k-2}{k+2}} \leq (N^{k+1}+1)^{\frac{k-2}{k+2}}.$$

---

†Let $RC(N_1,N_2)$ represent the cost of routing between every pair of nodes in the set $(N_1 \times N_2)$. Then we can say total routing cost = $RC(S,S) + 2RC(S,S') + RC(S',S') + 2RC(S,\{T\}) + 2RC(S',\{T\})$, where the factors of 2 are a result of the symmetry of the required flows in the network. Since all arcs from the original Steiner tree problem have routing cost zero, $RC(S,S) = 0$. We can always utilize the special arcs connecting T to the rest of the network so we have $RC(S,S') \leq N^{k+1}$, $RC(S',S') \leq 2N^2$, $RC(S,\{T\}) \leq N^k$ and $RC(S',\{T\}) \leq N$. Therefore,

$$\text{total routing cost} \leq 2N^{k+1} + 2N^2 + 2N^k + 2N \leq 2N^{k+2}, \quad k \geq 1 \text{ and } N \geq 4.$$

So for $N \geq 4$

$$(N^{k+1} + 1)\ \frac{k-2}{k+2} \leq N^{k-2}$$

and $R_{h^*}(M) < N^{k-2}$ $\qquad N \geq 4$.

The above inequality implies that for $N \geq 4$ the Steiner tree problem could be solved in polynomial time by first using our polynomial time transformation to create an optimal network problem and then applying the heuristic h^* to it. The existence of a subtree satisfying the conditions of the Steiner problem could be verified by examining whether the heuristic gave a routing cost solution that was less than N^{2k}.

Since the finite number of cases where $N < 4$ will not effect the polynomial time bound of this procedure, the above inequality implies that the Steiner tree problem could be solved in polynomial time.

Finding a heuristic h^* as defined above is equivalent to solving an NP-complete problem, so we can say that the approximate optimal network problem is also NP-complete. □

We have seen that all polynomial-time bounded heuristics most probably have a worst-case error ratio that grows almost linearly with the size of the network, or at a faster rate. Next we see that for reasonable heuristics the error ratio grows no faster than linearly with the size of the network.

Before presenting this result we introduce some additional notation. Let T be any spanning tree of a network and arbitrarily choose a node R with degree one from T and designate it as the root node. A node f is the

father of node n if f lies on the (unique) path in T between n and R and if there is an arc in T that connects f and n. Node s is the son of node f if f is its father. Let w_i be the number of nodes which are descendants of node i (i.e. nodes other than i whose path to R in T must pass through i). Des(n) is the set of nodes which are descendants of n.

Figure 5.8 contains an example illustrating these definitions. Node 1 is the root node.

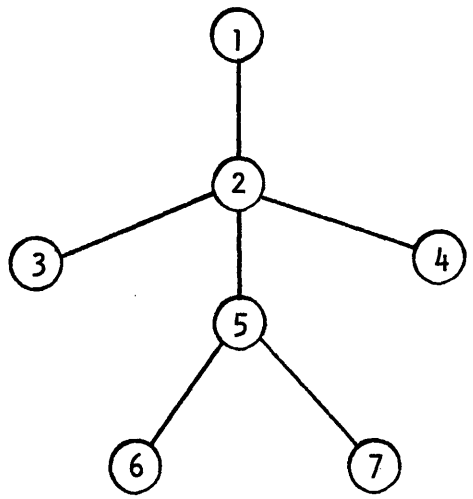


Figure 5.8 Example of a Tree with Root Node 1.

In this example node 2 is the father of node 5. $w_2=3$ and $w_6=0$.

Theorem 5.2 For optimal network problems whose routing costs satisfy the triangle inequality, any heuristic h which always produces a feasible solution will have a worst case error ratio

$$R_h(N) \leq 2N \qquad \text{for all } N,$$

where N is the number of nodes in the input network.

Proof: We will show that the ratio

$$\frac{\text{routing cost of any spanning tree network}}{\text{routing cost of the complete network}}$$

is always bounded by 2N. The theorem immediately follows from this since the above ratio is greater than or equal to $R_h(N)$.

Let T be any spanning tree for an optimal network problem P. Let C denote the complete network where every arc in P is included. Let RC(T) represent the routing cost of network design T and N be the total number of nodes in P.

Now we compute RC(T). Choose a root node R for it. Consider an arc (i,j) belonging to T (Since we are dealing with undirected arcs, let us assume that for any arc (i,j) in T, i is the father of j). Its contribution to the total routing cost is $S(i,j) = 2(w_j + 1)(N - (w_j + 1))c_{ij}$ (that is, the number of origin-destination pairs whose travel path passes through arc (i.j) multiplied by the routing cost of arc (i,j).

Therefore,

$$RC(T) = \sum_{(i,j)\varepsilon T} S(i,j).$$

Consider now the routing cost for the complete network. Since the triangle inequality holds and all required flows are one we have

$$RC(C) = 2 \sum_{(i,j)\varepsilon A} c_{ij}$$

where A is the set of arcs in the complete network.

Let us define the following quantity

$$C(i,j) = \sum_{k\epsilon Des(j)\cup\{j\}} 2(c_{jk} + c_{ki}) \geq 2(w_j + 1)c_{ij} \qquad (i,j)\epsilon T$$

where the inequality follows from the triangle inequality for the routing costs and the symmetry of the routing costs (since the arcs are undirected).

Therefore,

$$\frac{S(i,j)}{C(i,j)} \leq \frac{2(w_j+1)(N-(w_j+1))c_{ij}}{2(w_j+1)c_{ij}} \leq N \qquad (i,j)\epsilon T.$$

Combining these inequalities for all $(i,j)\epsilon T$ we have

$$\frac{\sum_{(i,j)\epsilon T} S(i,j)}{\sum_{(i,j)\epsilon T} C(i,j)} \leq N$$

and since $\sum_{(i,j)\epsilon T} S(i,j) = RC(T)$

$$\frac{RC(T)}{\sum_{(i,j)\epsilon T} C(i,j)} \leq N .$$

Next we show that $\sum_{(i,j)\epsilon T} C(i,j) \leq 2RC(C)$ and thus complete the proof.

We argue that each arc cost term c_{rt} appears in at most two expressions of the form $C(i,j)$ (without loss of generality assume that node t is a descendent of node s). c_{st} appears in the expression $C(i,j)$

only if

1) j equals s. Recall that since i must be the father of j, i must be the father of s.

2) i equal s and t belongs to $Des(j) \cup \{j\}$.

The first situation can only happen once since node s must have a unique father. The second situation can only occur once since if it happened twice, for example, with $C(i,j_1)$ and $C(i,j_2)$, $j_1 \neq j_2$, then between s and t there would be two distinct paths in the tree T.

Since $RC(C) = 2 \sum_{(i,j) \in A} c_{ij}$ and the term c_{ij} occurs in at most two terms of the form C(s,t) we have

$$\sum_{(i,j) \in T} C(i,j) \leq 2RC(C) .$$

Therefore,

$$\frac{RC(T)}{2RC(C)} \leq N$$

or

$$\frac{RC(T)}{RC(C)} \leq 2N . \qquad \square$$

Notice that optimal network problem used in the proof of Theorem 5.1 had routing costs which satisfy the triangle inequality. Therefore Theorem 5.1 also holds if we impose the triangle inequality for the routing costs of the optimal network problem.

With these two theorems we have demonstrated probable lower and upper bounds on the worst-case error ratio for all reasonable polynomial-

time heuristics for the optimal network problem with the triangle inequality for all routing costs.

The above results can also be extended to situations in which the required flows $r_{k\ell}$ are not necessarily equal to one. Suppose that all the $r_{k\ell}$ are positive integers such that

$$\max_{i,j,k,\ell} \frac{r_{k\ell}}{r_{ij}} \leq N^P \qquad \text{for some } P \geq 3. \text{ Then}$$

Theorem 5.1 is modified by changing the worst-case error ratio from $N^{1-\varepsilon}$ to $(N^{P-2} - 1)$. Theorem 5.2 is modified by changing the upper bound of 2N to $2N^{P+1}$. The proofs of such generalizations are straightforward modifications of the ones given above and will not be given here.

5.4 A Heuristic for a Special Case of the Optimal Network Problem

In this section we consider a special case of the optimal network problem where all construction costs d_{ij} are one. The budget constraint for this type of problem essentially limits the number of arcs allowed in the optimal network design. Also we will again assume that the triangle inequality holds for the routing costs. Johnson et al [63] have also shown that this restricted problem is NP-complete.

With these new restrictions on the problem, the result of Theorem 5.2 is no longer valid. We will describe a polynomial-time heuristic h whose worst-case error ratio

$$R_h(N) \leq 2 \qquad \text{for all N.}$$

Let STAR(i) be a star network consisting of all arcs connecting node i to any other node in the network. COST(I) is the sum of all the routing costs from node i to every other node in the network.

Our third heuristic can be defined as:

(H3) 1) Find i such that

$$COST(i) = \underset{j \in D}{MIN}\ COST(j).$$

2) STAR(i) is the proposed network configuration.

Theorem 5.3 For optimal network problems satifying the triangle inequality for routing costs and having all construction costs equal to one

$$R_{H3}(N) \leq 2 \qquad \text{for all } N.$$

Proof: We demonstrate this result by proving the stronger fact that

$$\frac{V_{H3}(P)}{RC(C)} \leq 2, \qquad \text{for all } P,$$

where $V_{H3}(P)$ is the value of the solution computed by heuristic H3 for optimal network problem P and RC(C) is the routing cost (and solution cost) of the complete network.

The routing cost for connecting node $j \neq i$ to all other nodes in the network using the network STAR(i) is $2(N-2)C_{ij} + COST(i)$.

So

$$V_{H3}(P) = N \cdot COST(i) + 2(N-2) \sum_{j \neq i} C_{ij} .$$

Since

$$COST(i) = \sum_{j \neq i} 2C_{ij} ,$$

$$V_{H3}(P) = (2N-2)\ COST(i) .$$

The minimum cost of servicing the traffic between node j and every other node in the network is COST(j), since the triange inequality holds for routing costs. Therefore,

$$RC(C) = \sum_{j \in D} COST(j) .$$

We chose i such that $COST(i) \leq COST(j)$ for all j. This implies

$$RC(C) \geq N \cdot COST(i),$$

and

$$\frac{V_{H3}(P)}{RC(C)} \leq \frac{2N-2}{N} \leq 2. \qquad \square$$

Note that heuristic H3 has polynomially bounded computation time so that it is possible to have a polynomial time approximation procedure for a restricted class of network design problems whose worst-case error ratio is bounded by a constant. Theorem 5.1 shows that it is unlikely that such a heuristic exists for a broader class of network design problems.

We believe that combining some local improvement heuristic (perhaps one which added arcs in a "greedy" manner) with H3 could lead to a useful optimal network problem heuristic. It would be necessary to perform

additional worst-cases analyses or some computational tests in order to verify this conjecture.

Finally, the theoretical results of section 5.3 indicate that for the general optimal network problem it is very hard to find a polynominal time heuristic whose worst-case error ratio is very small. Computational tests by Dionne and Florian [27] for some simple heuristics have indicated that their relative margins of error are quite small. So although most heuristic probably have a bad worst-case ratio, there may be some heuristic whose average behavior is quite good. Future work in this area should include some probabilistic analyses of optimal network design heuristics.

CHAPTER VI

CONCLUSIONS AND FUTURE WORK

This thesis has focused on developing and analyzing new techniques for solving network design models and other mixed integer programming problems.

Chapter II provided an overview of the current literature on network design models with particular emphasis on computational methods and results.

Chapters III and IV describe new results and methodology for enhancing the computational performance of Benders decomposition procedure.

Chapter III concerns the proper selection of cuts to use for Benders decomposition. The theoretical results of sections 3.3 and 3.4 and the computational results of section 3.5 demonstrate the possibility of accelerating Benders decomposition by using strong or pareto optimal cuts. In section 3.3, Theorem 3.1, and its corollary, provide a mechanism for accelerating Benders decomposition applied to any mixed integer program. This approach is also applicable to a broader class of relaxation algorithms for minimax problems such as Dantzig-Wolfe decomposition for the Lagrangian dual of a nonlinear problem. The adaption of these techniques in section 3.4.2 to facility location models yields an efficient special purpose algorithm.

An alternate approach to pareto optimality is to generate cuts that are insured to dominate the standard cut, but with the possible loss of

pareto optimality. Examples are the problem dependent type C cut of section 3.4.1 for facility location models and the strong cut of section 3.4.4 for the fixed charge network design model.

Our computational experience in section 3.5.1 for the type C cuts indicates that they perform well particularly as the fraction of medians to nodes increases. For p-median problems (up to 33 nodes) Benders algorithm equipped with our methodology finds solutions known to be within 10 per cent of optimality in ten or fewer iterations. The standard implementation usually provides no better solution within twenty-five iterations and solutions 10 per cent farther from optimality within ten iterations.

We might remark that Benders algorithm probably is not competitive with specialized algorithms for the simples class of facility location problems which includes p-median and uncapacitated plant location problems. Recently, several researchers, Cornuejols, Fisher and Nemhauser [21], Bilde and Krarup [12], and Erlenkotter [31], (See Marsten [81], Garfinkel et al. [41], and Schrage [112], for linear programming approaches) have reported very powerful heuristic algorithms for these problems.† However, their approaches do not easily extend to more elaborate facility location models for which Benders algorithm and our modifications are still a viable solution technique. Additional advantages of Benders decomposition are:

---

†In fact, the latter two algorithms are very similar to our algorithm for generating pareto optimal cuts for facility locations problems. These algorithms have a similar δ-neighborhood interpretation. One distinguishing feature of our algorithm is that it solves problem (24) exactly, whereas, these other procedures give approximate solutions to the dual of the linear programming relaxation of problem (17).

i) it guarantees convergence to an optimal solution, and consequently ii) it has the advantage of aiding in sensitivity analysis. Geoffrion [47] argues persuasively for this capability.

Our computational experience in section 3.5.2 for the strong cut applied to the fixed charge network design problem again demonstrates the superior performance of the strong cuts. On the average, the strong cut performs about 15% better than the standard cut with respect to the measure of best lower bound divided by best upper bound. For this class of problems, it appears that Benders algorithm is competitive with other algorithms even for "simple" uncapacitated design problems. The test problems that we are solving are comparable in size to the largest problems that other researchers have solved.

Problems apparently small in terms of number of nodes and arcs give rise to very large mixed integer programming formulations. The mixed integer programming formulation that we stated in section 3.4.4 for the 33 node-100 arc problem of our computational experiments contains approximately 100,000 continuous variables, 100 binary variables and 133,000 constraints.

Chapter IV discusses the relationship between the proper mathematical formulation of mixed integer programming models such as network design problems and the computational performance of Benders decomposition. Section 4.3 presents a criteria for selecting among alternate model formulations for use with Benders decomposition. Suggestions are also made for modifying mdoel formulations in order to improve the performance of Benders procedure.

Chapter V deals with heuristic techniques for solving network design problems, in particular, the optimal network design problem. Worst-case error bounds are derived in order to provide more insight into the behavior of these procedures. Section 5.2 gives examples illustrating the large error margins that can occur when using some network design heuristics. Theorem 5.1 indicates that most reasonable network heuristics are capable of producing very inaccurate solutions. Theorem 5.1 provides an upper bound on the maximum error that can occur when using these sub-optimal procedures. Finally, section 5.4 gives a procedure whose error margin for a particular set of optimal network problems is always bounded by a constant.

There are a number of areas for future work concerning the results described in this thesis.

For chapter III, the following areas could be fruitful:

1) Employ the methodology implied by Theorem 3.1 and its corollary for Benders decomposition applied to other mixed integer programming problems. This could include solving directly the linear program (3.14) in order to generate pareto optimal cuts. Alternatively, special algorithms such as the one described in section 3.4.2 for facility location models could be designed to generated pareto optimal cuts.

2) All techniques for generating pareto optimal cuts require a core point (relative interior point of a convex set) as an input parameter. Find effective methods for generating these relative interior points.

3) Test the strategy of varying the core point used to generate a cut. Some core points may be preferable to others in accelerating Benders decomposition.

4) The computational tests for our Benders acceleration techniques were limited by the lack of availability of an efficient code for solving the integer programming master problem. Combine our strong cut techniques with an implementation of Benders decomposition that utilizes a powerful integer programming code. More extensive computational tests should then be performed.

5) Utilize our cut generating techniques with some variations of Benders decomposition such as the ε-optimal version described by Geoffrion and Graves [49] and the branch and bound search procedure with embedded Benders cuts (see Balas [4], Bricker [17], Guignard and Spielberg [55], Lemke and Spielberg [74], Rardin and Unger [102] and Unger [12] for a discussion of this technique).

6) Apply our acceleration technique to other relaxation algorithms such as Dantzig-Wolfe decomposition for linear and non-linear programming problems.

For chapter IV the following areas could be interesting:

7) Extended the results concerning preferred model formulations by designing procedures for strengthening mixed integer models in the context of Benders decomposition.

8) For different model formulations, explore the computational tradeoffs between the improved Benders cuts available and the more complicated subproblems that must be solved.

The results of chapter V are initial efforts to analyze the behavior of network design heuristics. The mathematical analysis of these procedures should lead to a better understanding of them and to the design of more powerful heuristics. The following areas could be useful areas of future research:

9) Derive worst-case and average-case error bounds for other network design heuristics (see [14,89,117,128] for various heuristics).

10) Explore the relationship between these theoretical performance measures and computational performance on real network design problems.

11) Most of the work in heuristics analysis has been concerned with error margins relative to the problem size (e.g. the number of nodes in the problem network). Explore the effect that other parameters such as the range of network costs have on heuristic error margins. For examples, in section 5.3 we discussed how the maximum possible error increased as a function of the range of required flows in the optimal network problem.

REFERENCES

1] Agarwal, S.K., "Optimizing Techniques for the Interactive Design of Transportation Networks Under Multiple Objectives", Ph.D. Thesis, Department of Civil Egnineering, Morthwestern University, 1973.

2] Armstrong, R.D., and Willis, C.E., "Simultaneous Investment and Allocation Decisions Applied to Water Planning", Man. Sci., Vol.23, pp. 1080-1088, 1974.

3] Assad, A., "Multicommodity Network Flows - A Survey", To appear in Networks.

4] Balas, E., "Minimax and Duality for Linear and Nonlinear Mixed-Integer Programming", in (J. Abadie, ed.), Integer and Nonlinear Programming, North Holland, Amsterdam, 1970.

5] Balinski, M.L., "Integer Programming: Methods, Uses, Computation", Man. Sci., Vol. 12, pp. 253-313, 1965.

6] Bansal, P., and Jacobsen, S., "An Algorithm for Optimizing Network Flow Capacity Under Economies of Scale", J. of Opt. Th. and Applic., Vol. 15, No. 5, pp. 565-586, 1975.

7] Barbier, M., "Le Future Réseau de Transports en Région de Paris", Cahiers de l'Institute d'Aménagement et d'Urbanisme de la Région Parisienne, 4-5, No. 4, 1966.

8] Beale, E.M.L. and Tomlin, J.A., "An Integer Programming Approach to a Class of Combinatorial Problems", Math. Prog., Vol. 3, pp. 339-344, 1972.

9] Beckmann, M., McGuire, C.B. and Winsten, C., Studies in the Economics of Transportation, Yale University Press, New Haven, CT, 1956.

10] Bellmore, M. and Nemhauser, G.L., "The Travelling Salesman Problem - A Survey", Opns. Res., Vol. 16, pp. 538-558, 1968.

11] Benders, J.F., "Partitioning Procedures for Solving Mixed Variables Programming Problems", Num. Math., Vol. 4, pp. 238-252, 1962.

12] Bilde, O., and Krarup, J., "Sharp Lower Bounds and Efficient Algorithms for the Simple Plant Location Problem," Annals of Discrete Math., Vol. Vol. 1, pp. 79-92, 1977.

13] Billheimer, J.W., "Optimal Route Configurations with Fixed Link Construction Costs", Ph.D. Thesis, Industrial Engineering Department, Stanford University, 1970.

14] Billheimer, J. and Gray, P., "Network Design with Fixed and Variable Cost Elements", Trans. Sci., Vol. 7, pp. 49-74, 1973.

15] Boyce, D.E., Farhi, A., and Weischedel, R., "Optimal Network Problem: A Branch-and-Bound Algorithm", Environment and Planning, Vol. 5, pp. 519-533, 1973.

16] Braess, D., "Uber ein Paradexen der Verkehrsplanung", Unternehmensforschung, Vol. 12, pp. 258-268, 1968.

17] Bricker, D.L., "Exact Solution of the Shipment Planning Problem with Nonconvex Freight Costs: A Localized Enumeration Algorithm Employing Benders Partitioning", Ph.D. Thesis, Northwestern University, 1975.

18] Businessweek, "Why Airlines Fear the 'Federal Express Bill'," p. 116, September 13, 1976.

19] Carter, E.C. and Stowers, J.R., "Model for Funds Allocation for Urban Highways Systems Capacity Improvements", Highway Research Record, No. 20, pp. 84-102, 1963.

20] Chan, Y.P., "Optimal Travel Time Reduction in a Transport Network: An Application of Network Aggregation and Branch-and-Bound Techniques. Report No. R69-39, Department of Civil Engineering, M.I.T., Cambridge, MA, July 1969.

21] Cornuejols, G., Fisher, M.L., and Nemhauser, G.L., "An Analysis of Heuristics and Relaxations for the Uncapacitated Location Problem," Man. Sci., Vol. 23, pp. 789-810, 1977.

22] Dantzig, G.B., Linear Programming and Extensions, Princeton University Press, Princeton, N.J., 1973.

23] Dantzig, G.B., Maier, S.F., Lansdowne, Z.F. Harvey, R.P., and Robinson, D.W., "Application of Decomposition to Transportation Network Analysis", Preliminary Report, Control Analysis Corporation, January 1976.

24] Davis, P.S. and Ray, T.L., "A Branch-Bound Algorithm for Capacitated Facilities Location Problem", Nav. Res. Log. Q., Vol. 16, pp. 331-344, 1969.

25] Dem'yanov, V.F. and Malozemov, V.N., Introduction to Minimax, (Translated from Russian by D. Louvish), John Wiley & Sons, New York, 1974.

26] Dionne, R., "Une Analyse Théorique et numérique du Problème du Choix Optimal d'un Réseau de Transport sans congestion", Publication #198, Département d'informatique, Université de Montreal, Octobre 1974.

27] Dionne, R., and Florian, M., "Exact and Approximate Algorithms for Optimal Network Design", Publication #41, Centre de Recherche Sur les Transports, Universite de Montreal, 1977.

28] Dreyfus, S.E., Wagner, R.A., "The Steiner Problem in Graphs", Networks, Vol. 1, pp. 195-207, 1972.

29] Drinkwater, R.W., "The Placing of Multiplexers to Minimize Cable Distance", Op. Res. Q., Vol. 28, pp. 267-273, 1977.

30] Efroymsen, M.A. Ray, T.L., "A Branch-Bound Algorithm for Plant Location", Oper. Res., Vol. 14, pp. 361-368, 1966.

31] Erlenkotter, D., "A Dual-Based Procedure for Uncapacitated Facility Location", Working Paper No. 261, UCLA Western Man. Sci. Inst., 1976.

32] Fisher, M.L., and Shapiro, J.F., "Constructive Duality in Integer Programming", Siam J. on Applied Math, Vol. 27, pp. 31-52, 1974.

33] Florian, M.G., Guerin, G., and Bushel, G., "The Engine Scheduling Problem in a Railway Network", INFOR Journal, Vol. 14, pp. 121-138, 1976.

34] Florian, M., and Nguyen, S., "A Method for Computing Network Equilibrium with Elastic Demands", Trans. Sci., Vol. 8, pp. 321-332, 1974.

35] Florian, M. and Nguyen, S., "An Application and Validation of Equilibrium Trip Assignment Methods", Trans. Sci., Vol. 10, pp. 324-390, 1976.

36] Francis, R.L. and Goldstein, J.M., "Location Theory: A Selective Bibliography", Opns. Res., Vol. 22, pp. 400-410, 1974.

37] Frey, S.C. and Nemhauser, G.L., "Temporal Expansion of a Transportation Network-I", Trans. Sci., Vol. 6, 306-323, 1972.

38] Funk, M.L., Tillman, F.A., "A Dynamic Programming Approach to Optimal Construction Staging", American Society of Civil Engineers, Journal of the Highway Division, 94, pp. 255-265, 1968.

39] Garey, M.R. and Johnson, D.S., "Approximation Algorithms for Combinatorial Problems: An Annotated Bibliography", in Algorithms and Complexity (J.F. Traub, ed.), Academic Press, New York, 1976.

40] Garey, M.R., and Johnson, D.S., "The Complexity of Near-Optimal Graph Coloring", J. of Assoc. for Com. Mach., Vol. 23, pp. 43-49, 1976.

41] Garfinkel, R.S., Neebe, A.W., and Rao, M.R., "An Algorithm for the M-Median Plant Location Problem", Trans. Sci., Vol. 18, pp. 217-236, 1974.

42] Garfinkel, R. and Nemhauser, G., Integer Programming, Wiley, New York, 1972.

43] Geoffrion, A.M., "Elements of Large Scale Mathematical Programming, Parts I and II," Man. Sci., Vol. 16, pp. 652-691, 1970.

44] Geoffrion, A.M., "Primal Resource-Directive Approaches for Optimizing Nonlinear Decomposable Systems", Opns. Res., Vol. 18, pp. 375-403, 1970.

45] Geoffrion, A.M., "Generalized Benders Decomposition", J. of Opt. Theory and Applic., Vol. 10, pp. 237-260, 1972.

46] Geoffrion, A.M., "Lagrangian Relaxation and Its Uses in Integer Programming," Math. Prog. Study 2, pp. 82-114, 1974.

47] Geoffrion, A.M., "A Guide to Computer-Assisted Methods for Distribution Planning", Sloan Management Review, Vol. 16, pp. 17-41, Winter, 1975.

48] Geoffrion, A.M., "How can Specialized Discrete and Convex Optimization Methods be Married", UCLA Western Management Science Institute, Working Paper, No. 238, 1975.

49] Geoffrion, A.M. and Graves, G., "Multicommodity Distribution System Design by Benders Decomposition", Man. Sci., Vol. 5, pp. 822-844, 1974.

50] Geoffrion, A.M. and Bride, R., "Lagrangean Relaxation Applied to Facility Location Problems", Working Paper No. 263, UCLA Western Man. Sci. Inst., 1977.

51] Geoffrion, A.M., and Marsten, R.E., "Integer Programming Algorithms: A Framework and State-of-the-Art Survey", in Perspectives on Optimization, (A.M. Geoffrion, ed.), Addison-Wesley, Reading, MA, 1972.

52] Golden, B., "A Minimum-Cost Multi-Commodity Network Problem Concerning Imports and Exports", Networks, Vol. 5, pp. 331-356, 1975.

53] Goldman, A.J. and Nemhauser, G.L., "A Transport Improvement Problem Transformable to a Best-Path Problem", Trans. Sci., Vol. 1, pp. 295-307, 1967.

54] Gomory, R. and Hu, T., "Synthesis of a Communication Network", Journal of SIAM, Vol. 12, pp. 348-369, 1964.

55] Guignard, M., and Spielberg, K., "Search Techniques with Adaptive Features for Certain Mixed Integer Programming Problems", Proceedings IFIPS Congress, Edinburgh, 1968.

56] Hakimi, S.L., "Steiner's Problem in a Graph and Its Implications", Networks, Vol. 1, pp. 113-133, 1971.

57] Harary, F., Graph Theory, Addison-Wesley, Reading, MA 1969.

58] Hershdorfer, A.M., "Optimal Routing of Urban Traffic", Ph.D. Thesis, Department of City and Regional Planning, M.I.T., Cambridge, MA, May 1965.

59] Hoang, H.H., "A Computational Approach to the Selection of an Optimal Network", Man. Sci., Vol. 19, pp. 488-498, 1973.

60] Holloway, C., "A Generalized Approach to Dantzig-Wolfe Decomposition for Concave Programs", Opns. Res., Vol. 21, pp. 210-220, 1973.

61] Hu, T.C., "Optimum Communication Spanning Trees", SIAM Journal of Computing, Vol. 3, No. 3, pp. 188-195, Sept. 1974.

62] Johnson, D.S., "Approximation Algorithms for Combinatorial Problems", J. of Computer and Systems Sciences, Vol. 9, pp. 256-278, 1974.

63] Johnson, D.S., Lenstra, J.K., Rinnooy Kan, A.H.G., "The Complexity of the Network Design Problem", (to appear in Networks).

64] Karg, R.L., and Thompson, G.L., "A Heuristic Approach to Solving Travelling Salesman Problems", Man. Sci., Vol. 10, pp. 225-248, 1964.

65] Karp, R.M., "Reducibility Among Combinatorial Problems", Complexity of Computer Computations, (R.E. Miller and J.W. Thatcher, eds.), Plenum Press, New York, pp. 85-104, 1972.

66] Karp, R.M., "On the Computational Complexity of Combinatorial Problems", Networks, Vol. 5, pp. 45-68, 1975.

67] Karp, R.M., "The Probabilistic Analysis of Some Combinatorial Search Algorithms", in Algorithms and Complexity, (J.F. Traub, ed.), Academic Press, New York, 1976.

68] Kennington, J., "Multi-Commodity Flows: A State-of-the-Art Survey of Linear Models and Solution Techniques", Tech. Rep. IEOR 75014, Southern Methodist University, Dec. 1975.

69] Knödel, W., Graphentheoreticische Methoden und ihre Anwendungen, Springer-Verlag, Berlin, pp. 56-59, 1969.

70] Lasdon, L.S., Optimization Theory for Large Systems, The MacMillan Company, New York, 1970.

71] LeBlanc, L.J., "An Algorithm for the Discrete Network Design Problem", Trans. Sci., Vol. 9, No. 3, pp. 283-287, Aug. 1975.

72] Leblanc, L.J., and Morlok, E.K., and Pierskalla, W.P., "An Efficient Approach to Solving the Road Network Equilibrium Traffic Assignment Problem", Trans. Res., Vol. 9, pp. 309-318, 1975.

73] Lemarechal, D., "An Extension of Davidson's Methods to Non-Differ Differentiable Problems", Math. Prog. Study 3, 1975.

74] Lemke, C.E., and Spielberg, K., "Direct Search Zero-One and Mixed Integer Programming", Opns. Res, Vol. 15, pp. 892-914, 1967.

75] Los, M., "Simultaneous Optimization of Land Use and Transportation: A Synthesis of the Quadratic Assignment Problem and the Optimal Network Problem", Publication #31, Centre de Recherche Sur Les Transports, Université de Montreal, 1975.

76] MacKinnon, R.D., "Optimization Models of Transportation Network Improvement: Review and Future Prospects", Working Paper WP-76, University of Toronto, Feb. 1976.

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

78] Magnanti, T.L. and Golden, B.L., "Transportation Planning: Network Models and Their Implementation", Mass. Inst. of Tech., Cambridge, MA, 1977.

79] Magnanti, T.L., Shapiro, J.F., and Wagner, M.W., "Generalized Programming Solves the Dual", Man. Sci., Vol. 22, pp. 1194-1203, 1976.

80] Magnanti, T.L., and Wong, R.T., "Accelerating Benders Decomposition for Network Design", Discussion Paper, Center for Operations Research and Econometrics, Université Catholique de Louvain, Belgium, 1978.

81] Marsten, R.E., "An Algorithm for Finding Almost All the Medians of A Network", Paper No. 23, Center Math. Studies in Economics and Man. Sci, Northwestern University, 1972.

82] Marsten, R.E., "The Use of the BOXSTEP Method in Discrete Optimization", Math. Prog. Study 3, 1975.

83] Marsten, R.E., Hogan, W.W., and Blankenship, J.W., "The BOXSTEP Method for Large-Scale Optimization", Opns, Res., Vol. 23, pp. 389-405, 1975.

84] Martin, B.V., and Manheim, M.L., "A Research Program for Comparison of Traffic Assignment Techniques", Highway Research Record, 88, 1965.

85] McCallum, C.J., "A Generalized Upper Bounding Approach to a Communications Network Planning Problem", Networks, Vol. 7, pp. 1-23, 1976.

86] McDaniel, D., and Devine, M., "Alternative Relaxation Schemes for Benders' Partitioning Approach to Mixed Integer Programming", School of Ind. Eng., University of Oklahoma, 1974.

87] Mevert, P., "Fixed Charge Network Flow Problems: Applications and Methods of Solution", Presented at Large Scale and Hier archical Systems Workshop, Brussels, May 1977.

88] Mifflin, R., "An Algorithm for Constrained Optimization with Semismooth Functions", International Inst. for Appl. Sys. Analysis, Laxenberg, Austria, 1977.

89] Morlok, E.K. and Leblanc, L.J., "A Marginal Analysis Technique for Determining Improvements to an Urban Road Network", Paper presented at ORSA/TIMS National Meeting, Las Vegas, NV, Fall 1975.

90] Murchland, J.D., "Braess' Paradox of Traffic Flow", Trans. Res., Vol. 4, pp. 391-394, 1970.

91] Nemhauser, G.L., and Widhelm, W.B., "A Modified Linear Program for Columnar Methods in Mathematical Programming," Opns. Res., Vol. 19, pp. 1051-1060, 1971.

92] Nemhauser, G., and Wolsey, L., "Best Algorithms for Approximating the Maximum of a Submodular Function", Paper presented at ORSA/TIMS Meeting, Atlanta, GA, 1977.

93] Newell, G.F., "Optimal Network Geometry", Sixth International Symp. on Trans. and Traffic Theory, Elseview, New York, pp. 561-580, 1974.

94] Nguyen, S., "An Algorithm for the Traffic Assignment Problem", Trans. Sci., Vol. 8, pp. 203-216, 1974.

95] Nguyen, S., "Book Review of Optimization of Transport Networks", Trans. Sci., Vol. 9, pp. 283-284, 1975.

96] Noonan, F., and Giglio, R.J., "Planning Electric Power Generation: A Nonlinear Mixed Integer Model Employing Benders Decomposition", Man. Sci., Vol. 23, pp. 946-956, 1977.

97] Ochoa-Rosso, F., "Applications of Discrete Optimization Techniques to Capital Investment and Network Synthesis Problems", Research Report, R68-42, M.I.T., Department of Civil Engineering, June 1968.

98] Ochoa-Rosso, F., and Silva, A., "Optimum Project Addition in Urban Transportation Network Via Descriptive Traffic Assignment Models", Research Report, R68-44, M.I.T., Department of Civil Engineering, June 1968.

99] Oettli, W. and Prager, W., "Optimal and Suboptimal Capacity Allocation in Communication Networks, Jour. of Opt. Th. and Applic., Vol. 8, pp. 396-411, 1971.

100] O'Neill, R.P., and Widhelm, W.B., "Computational Experience with Normed and Nonnormed Column-Generation Procedures in Nonlinear Programming", Opns. Res., Vol. 23, pp. 372-382, 1972.

101] Orchard-Hays, W., Advanced Linear Programming Computing Techniques, McGraw-Hill, New York, 1968.

102] Rardin, R.L., and Unger, V.E., "Surrogate Constraints and the Strength of Bounds Derived from 0-1 Benders' Partitioning Procedures", Opns. Res., Vol. 24, pp. 1169-1175, 1976.

103] Revelle, C., Marks, D., and Liebman, J., "An Analysis of Private and Public Sector Location Models", Man. Sci., Vol. 16, pp. 692-707, 1970.

104] Richardson, R., "An Optimization Approach to Routing Aircraft", Trans. Sci., Vol. 10, pp. 52-71, 1976.

105] Ridley, T.M., "Investment in a Network to Reduce the Length of the Shortest Route", Proceedings of the Third International Symposium on the Theory of Traffic Flow, Elsevier, New York, pp. 235-236, June 1965.

106] Ridley, T.M., "An Investment Policy to Reduce the Travel Time in a Transportation Network", Transportation Research, Vol. 2, pp. 409-424, 1968.

107] Roberts, P.O. and Funk, M.L., Toward Optimum Methods of Link Addition in Transportation Networks, M.I.T., Department of Civil Engineering, Sept. 1964.

108] Rockefellar, R.T., Convex Analysis, Princeton University Press, Princeton, N.J., 1970.

109] Rosenkrantz, D.J., Stearns, R.E. and Lewis, P.M., "Approximate Algorithms for the Travelling Salesperson Problem", 15th Annual IEEE Symposium on Switching and Automata Theory, pp. 33-52, 1974.

110] Rothfarb, B., and Goldstein, M., "The One-Terminal Telpak Problem", Opns. Res., Vol. 19, pp. 156-169, 1971.

111] Sahni, S., and Gonzalez, T., "P-Complete Approximation Problems", JACM, Vol. 23, pp. 555-565, 1976.

112] Schrage, L., "Implicit Representation of Variable Upper Bounds in Linear Programming", Math. Prog. Study 4, pp. 118-132, 1975.

113] Schwartz, J.G., Network Flow and Synthesis Models for Transportation Planning: A Survey, Research Report, R68-49, M.I.T., Department of Civil Engineering, June 1968.

114] Scott, A.J., "A Programming Model of an Integrated Transportation Network", Pap. Reg. Sci. Ass., 19, pp. 215-222, 1967.

115] Scott, A.J., "The Optimal Network Problem: Some Computational Procedures", Trans. Res., Vol. 3, pp. 201-210, 1969.

116] Stairs, S., "Selecting a Traffic Network", J. Transport. Econ. and Policy, Vol. 2, pp. 218-231, 1968.

117] Steenbrink, P.A., "Transport Network Optimization in the Dutch Integral Transportation Study", Transportation Research, Vol. 8, pp. 11-27, 1974.

118] Steenbrink, P.A., Optimization of Transport Networks, John Wiley & Sons, London, 1974.

119] Soukup, J., "On Minimum Cost Networks with Nonlinear Costs", SIAM J. of Applied Mathematics, Vol. 29, pp. 571-581, 1975.

120] Tomlin, J., "Minimum-Cost Multi-Commodity Network Flow", Opns. Res., Vol. 14, No. 1, pp. 45-51, 1966.

121] Unger, V.E., "Capital Budgeting and Mixed 0-1 Integer Programming" AIIE Transactions, Vol. 2, pp. 28-36, 1970.

122] Wardrop, J.G., "Some Theoretical Aspects of Road Traffic Research", Proc. Inst. Civ. Eng., 1, Part II, pp. 325-362, 1952.

123] Williams, H.P., "Experiments in the Formulation of Integer Programming Problems", Math. Prog. Study 2, pp. 180-197, 1974.

124] Wolfe, P., "Convergence Theory in Nonlinear Programming", in (J. Abadie, ed.), Integer and Nonlinear Programming, North Holland, Amsterdam, 1970.

125] Wolfe, p., "A Method of Conjugate Subgradients for Minimizing Nondifferentiable Functions", Math. Prog. Study 3, 1975.

126] Wollmer, R., "Removing Arcs from a Network", Opns. Res., Vol. 12, pp. 934-940, 1964.

127] Wong, R., "A Survey of Network Design Problems", Working Paper ORC 053-76, M.I.T., Operations Research Center, 1976.

128] Yaged, B., Jr., "Minimum Cost Routing for Static Network Models", Networks, Vol. 1, pp. 139-172, 1971.

129] Yaged, B., Jr., "Minimum Cost Routing for Dynamic Network Models", Networks, Vol. 3, pp. 193-224, 1973.

130] Zadeh, N., "On Building Minimum Cost Communication Networks", Networks, Vol. 3, pp. 315-331, 1973.

131] Goldstein, M., and Rothfarb, B., "The One Terminal Telepack Problem", Opns. Res., Vol. 19, pp. 156-169, 1971.

132] Wagner, H.M. and Whitin, T.M., "Dynamic Version of the Economic Lot Size Model", Man. Sci., Vol. 5, pp. 89-96, 1958.

BIOGRAPHY

Richard Tekee Wong was born in New Rochelle, New York in 1949. He attended grammar school and high school in New Rochelle. During the summer of 1967 he studied Chinese at Columbia University with a National Defense Foreign Language Fellowship. In 1967 he entered M.I.T. as a freshman and after four years entered the graduate school at M.I.T. in the Department of Electrical Engineering. In 1972 he received a Bachelor of Science and Master of Science in Electrical Engineering with a concentration in computer science. He also received the degree of Electrical Engineering in 1973. He received the Supervised Investors Inc. Teaching award from the Department of Electrical Engineering and Computer Science in June, 1976. During the year 1976-77 he was a visiting graduate student at the center for Operations Research and Econometrics in Heverlee, Belgium.

His research interests include Transportation networks and mathematical programming.

His publications include:

1) "A Survey of Network Design Problems," M.I.T. Operations Research Center Working Paper (May 1976).

2) "Modelling and Optimization for Transportation Systems Planning and Operations," in Large Engineering Systems (A. Wexler ed.) Pergamon Press, Toronto (with N. Gartner and B. Golden) (1977).

3) "Accelerating Benders Decomposition for Network Design," Center for Operations Research and Econometrics, Louvain-la-Neuve, Belgium, (with T. Magnanti) (1978).