VALID INEQUALITIES AND ALGORITHMS FOR THE NETWORK DESIGN PROBLEM WITH AN APPLICATION TO LTL CONSOLIDATION

by

ANANTARAM BALAKRISHNAN

Bachelor of Technology,
Indian Institute of Technology, Madras, India
(1976)

Post-graduate Diploma in Management,
Indian Institute of Management, Ahmedabad, India
(1978)

Submitted to the
Sloan School of Management
in Partial Fulfillment of the
Requirements of the Degree of

DOCTOR OF PHILOSOPHY

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

December 1984

Signature of Author: _____
Sloan School of Management, 27 December 1984

Certified by: _____
Thomas L. Magnanti, Thesis Supervisor

Accepted by: _____
Richard Schmalensee, Chairman, PhD. program

VALID INEQUALITIES AND ALGORITHMS FOR THE NETWORK DESIGN PROBLEM WITH AN APPLICATION TO LTL CONSOLIDATION

by

ANANTARAM BALAKRISHNAN

Submitted to the Sloan School of Management
on December 27, 1984 in partial fulfillment of the
requirements for the degree of Doctor of Philosophy.

ABSTRACT

The Network Design problem has numerous practical applications in contexts as diverse as transportation planning, communications system design, and production planning. In addition, it generalizes several well-known optimization models including the Plant Location and Steiner Network problems. The general problem is to select a subset of arcs from a given network and to route required multicommodity flows over these arcs, subject to arc capacity and other design constraints. The objective is to minimize the sum of the fixed costs for the selected arcs and the routing costs.

In this thesis, we examine both modeling and algorithmic aspects of the Network Design problem. We begin by studying the linear programming relaxation of the uncapacitated problem. We characterize the extreme points of this relaxation and use this result to construct two classes of valid inequalities that exclude fractional extreme points. We then specialize several families of Set packing inequalities to the Uncapacitated Network Design problem and demonstrate how some of these cuts can also be generated by constraint aggregation. Next, we consider algorithms and computational testing. Using a general framework, we discuss several alternative dual ascent schemes for generating upper and lower bounds for the problem, and report on computational experience with one particular implementation of this algorithm. Finally, we consider an application to transportation planning, namely the Less-than-Truckload (LTL) Consolidation problem. We devise a Lagrangian-based algorithm that incorporates heuristic and lower bounding procedures as well as a method for problem reduction. We discuss the computational performance of this algorithm for solving randomly generated networks with general and special configurations.

Thesis Supervisor: Thomas L. Magnanti
Title: Chairman and Professor of Management Science

To my parents

and

my wife, Revathi

ACKNOWLEDGEMENTS

Over the past several years, I have had the pleasure and good fortune to interact and work with several outstanding individuals who stimulated my interest in Operations Research and continue to inspire me. I must first thank Professor Nitin Patel of the Indian Institute of Management, Ahmedabad, for introducing me to this field and encouraging me to pursue doctoral studies. My advisor, Professor Thomas Magnanti has been a constant source of support and encouragement through all the phases of the PhD program at the Sloan School. His academic philosophy and research orientation have influenced me a great deal. He, along with the other members of my committee, Professor Stephen Graves and Professor Richard Wong, provided me valuable insights, perspective and direction. I am very grateful to them for their constructive suggestions and their time. I also wish to thank Professor John Little for initiating me into the research process at M.I.T., and Professor James Orlin for his interest and support. To these and several other people at the Sloan School who helped me both professionally and personally, I express my heartfelt thanks. In particular, I thank my colleagues, Hamid Zeghmi and Rita Vachani, for their assistance and counsel on numerous occasions. I am grateful to my parents for wholeheartedly supporting all my endeavours even if it meant a personal sacrifice. My wife, Revathi, cheerfully endured all my moods and habits. Her boundless patience, understanding and encouragement were, in no small measure, responsible for my completing this thesis. Finally, I gratefully acknowledge the financial support that I received under the auspices of the Center for Transportation Studies Affiliates Program at M.I.T., and under contract numbers ECS-792-6225 and ECS-831-664 from the Division of Systems Theory and Operations Research of the National Science Foundation.

TABLE OF CONTENTS

Page

Abstract..... 2

Dedication..... 3

Acknowledgements..... 4

Table of Contents..... 5

List of Tables..... 8

List of Figures..... 9

CHAPTER 1: INTRODUCTION..... 10

1.1 Problem motivation..... 10

1.2 Thesis objective..... 13

1.3 Formulations of the Network Design problem..... 17

1.4 Variants of the Network Design problem..... 25

1.5 Literature review..... 31

CHAPTER 2: FRACTIONAL EXTREME POINTS OF THE UNCAPACITATED NETWORK DESIGN PROBLEM..... 36

2.1 A characterization of fractional LP extreme points.... 38

2.2 Constructing violated, valid inequalities for the Uncapacitated Network Design problem..... 56

2.3 Concluding remarks..... 79

CHAPTER 3: VALID INEQUALITIES FOR THE NETWORK DESIGN PROBLEM..... 83

3.1 The Uncapacitated Network Design problem in Set-packing form..... 87

3.2 Valid inequalities induced by Cliques of the Intersection graph..... 99

3.3 Valid inequalities induced by Odd holes of the Intersection graph..... 101

3.4 Webs of the Intersection graph..... 141

Page

3.5 Inequalities induced by other subgraphs of the Intersection graph..... 144

3.6 Cutset inequalities for the Capacitated Network Design problem..... 155

3.7 Concluding remarks..... 161

CHAPTER 4: DUAL ASCENT ALGORITHMS FOR THE NETWORK DESIGN PROBLEM.. 163

4.1 Lagrangian relaxation for the Network Design problem.. 165

4.2 Benders' decomposition for the Network Design problem. 169

4.3 Dual Ascent procedures for the Network Design problem. 173

4.4 Dual Ascent procedures for the Capacitated case..... 222

4.5 Problem Reduction procedures for the Network Design problem..... 241

4.6 Computational results for the Uncapacitated Network Design problem..... 254

4.7 Concluding remarks..... 269

CHAPTER 5: A LAGRANGIAN-BASED ALGORITHM FOR THE PIECEWISE LINEAR, CONCAVE COST MULTICOMMODITY NETWORK FLOW PROBLEM..... 271

5.1 Problem definition and an application..... 272

5.2 Integer programming formulation for the Concave-cost Network Flow problem..... 278

5.3 The Lagrangian relaxation..... 282

5.4 Problem Reduction methods..... 306

5.5 Lagrangian-based heuristic procedures..... 312

5.6 Structure of the Composite Concave-cost Network Flow algorithm..... 314

5.7 Computational results..... 316

5.8 Variants of the Composite algorithm..... 335

5.9 Concluding remarks..... 343

Page

CHAPTER 6: CONCLUSIONS AND DIRECTIONS FOR FURTHER RESEARCH..... 345

APPENDIX A: A general cut-generation procedure for the Uncapacitated Network Design problem using the Auxiliary graph..... 354

APPENDIX B: Detailed computational results for General Concave-cost Network Flow problems..... 359

APPENDIX C: Detailed computational results for Three-layer Concave-cost Network Flow problems..... 363

REFERENCES..... 367

LIST OF TABLES

Page

TABLE 1: Values of the ratios r_t and r_w in different cases..... 28

TABLE 2: Uncapacitated Network Design test problem parameters..... 257

TABLE 3: Performance of various UNDP heuristic procedures..... 261

TABLE 4: Dual Ascent results for the Uncapacitated Network Design problem..... 266

TABLE 5: Problem Reduction results for the Uncapacitated Network Design problem..... 267

TABLE 6: General network problem size parameters..... 319

TABLE 7: Range widths for General networks..... 320

TABLE 8: Summary statistics for test runs on General network problems..... 323

TABLE 9: Three-layer network problem size parameters..... 330

TABLE 10: Summary statistics for test runs on Three-layer network problems..... 332

LIST OF FIGURES

Page

FIGURE 1: The Plant Location problem as a Network Design problem.. 30

FIGURE 2: Example for LP extreme point characterization..... 47

FIGURE 2A: Auxiliary graph corresponding to LP solution..... 48

FIGURE 2B: Auxiliary graph corresponding to revised LP solution.... 49

FIGURE 3: Structure of the Intersection graph..... 91

FIGURE 4: Examples of Odd holes in the Intersection graph..... 105

FIGURE 5: Structure of general Odd holes..... 106

FIGURE 6: Lifting Odd hole inequalities..... 117

FIGURE 7: Example of a 1-hole..... 122

FIGURE 8: Path from $[p(k_1)]$ to $[p'(k_1)]$ in the Auxiliary graph.... 123

FIGURE 9: Odd hole corresponding to the path in G_a..... 124

FIGURE 10: Illustration of maximum independent sets in 1-holes..... 129

FIGURE 11: Example of a 2-hole..... 131

FIGURE 12: Star configuration and associated 2-hole..... 133

FIGURE 13: Illustration for Type A nodes in 2-holes..... 136

FIGURE 14: Illustration for Type B nodes in 2-holes..... 138

FIGURE 15: Example of a Web in the Intersection graph..... 142

FIGURE 16: The subgraph $G(E^{4,3})$ of the Intersection graph..... 147

FIGURE 17: Example of an equivalent network..... 198

FIGURE 18: Total cost of flow on arc (i,j)..... 274

FIGURE 19: $z_{ij}(v)$ as a function of the total flow on arc (i,j)..... 289

FIGURE 20: Example of a Three-layer network..... 327

CHAPTER 1

INTRODUCTION

1.1 Problem Motivation

One of the central issues in many long-term planning exercises is to tradeoff the level of investment in facilities and services with the cost of operating the system using the available resources. For instance, while planning the distribution strategy of a firm, decision-makers typically face a wide spectrum of investment-distribution options for meeting demand. On the one hand, the firm can build regional warehouses to exploit economies of scale in transporting the goods; at the other extreme, by shipping directly from the plants to individual customers, the firm can minimize its investment but will incur higher distribution costs. In such situations, the decision-makers must systematically enumerate and evaluate all possible options before selecting the most appropriate strategy. Optimization problems of this type, that involve planning several interdependent activities, some or all of which have set-up or fixed charges, fall under the general category of FIXED-CHARGE PROBLEMS.

The special structure of network flow problems and the predominance of early applications of the fixed-charge problem in network-related contexts, most notably for transportation planning, communication systems design and facilities location, led researchers to focus on network specializations of this problem. When the fixed-charge problem is posed in this context, we refer to it as the NETWORK DESIGN PROBLEM. In this model, activities are

represented as commodities that must be transported from various origins to destinations (that are commodity specific). The arcs of the network represent the resources that can be used for this purpose. Each arc has a FIXED-CHARGE that is incurred if the arc is used, as well as a per unit FLOW or ROUTING cost for transporting commodities along that arc. The Network Design problem, then, involves

(1) selecting a subset of arcs from the given list of candidate arcs, and

(2) routing the different commodities over the selected arcs,

so that all demand requirements are satisfied and some function of the fixed and routing cost is optimized. The problem context might impose other constraints as well, such as capacities on the arcs or restrictions on the design topology.

The Network Design problem is a natural choice for modeling numerous complex systems. This model is distinctive because it explictly accounts for the dependence of the routing decisions on the chosen network configuration. Hence, the Network Design model is most appropriate for decision-making contexts in which

(1) the decision-makers have control over both network selection and routing, the former being a strategic issue and the latter a tactical decision,

(2) both the fixed charges and the routing costs are significant and are of comparable magnitudes, and

(3) the choice of the network configuration has a substantial impact on the system-wide flow profiles and costs.

These characteristics are especially evident in problem domains such as communications system design, transportation planning and water resources management. There is an extensive body of literature devoted entirely to the design of networks in each of these three areas. As a representative

list, we cite the following references:

- Boesch [1976], Boorstyn and Frank [1977], Chu [1977], Gerla and Kleinrock [1977], Schwartz [1979], and Tannenbaum [1982] for computer/communication networks,
- Boyce [1979], Magnanti [1981a], Magnanti and Wong [1984a], Manheim et al. [1968], and Steenbrink [1974] for transportation planning, and
- Jarvis et al. [1978], and Mandl [1981] for water resource management.

These references provide comprehensive discussions of the design issues that arise in each of the areas and outline solution approaches, many of them heuristic, that have been used in these contexts.

Network Design is also applicable to situations that do not involve the design of physical networks. For example, the problem of planning transportation services, such as airline routes and schedules, can be modeled as a Network Design problem in which nodes represent arrival and departure times at different locations and flows along arcs are interpreted as services to be scheduled. Similarly, the Production lot-sizing problem, which addresses the tradeoff between frequent set-ups and carrying inventory, can be formulated as a Network Design problem defined over an acyclic network with special structure.

In addition to these direct applications, the Network Design problem encompasses a wide range of simpler network optimization problems which, by themselves, are of great interest and relevance. Magnanti and Wong [1984a] demonstrate how various well-known optimization problems, such as the Minimal Spanning tree problem, the Shortest path problem, the Steiner Network problem, the Traveling Salesman problem, the Minimum-cost Multicommodity flow problem, the Traffic Equilibrium model and the Facility location problem, can be viewed as special cases of the Network Design

problem. These special cases arise when the network topology, the cost structure and/or the demand pattern of the general Network Design model are restricted in different ways. For instance, consider the Network Design problem with a single, linear objective function, without any capacity restrictions on arcs and having a single source node with positive demand at all other nodes. If the routing costs in this model are all zero and if all arcs are undirected, then the problem reduces to a Minimal Spanning tree problem; on the other hand, if all the fixed costs are zero, the problem is one of finding all shortest paths from the source node.

Thus, the Network Design problem is of interest not only because it is frequently encountered in its general form in practice but also because it serves as an integrating framework for numerous well-known optimization problems that have been extensively studied in the literature.

1.2 Thesis Objective

In this thesis, we focus on the class of Network Design problems in which the objective is to minimize the sum of the fixed costs of arcs that are included in the design and the flow costs incurred for routing the commodities over the selected arcs (rather than some nonlinear function of fixed and routing costs). Most of the applications that we cited above belong to this category of problems. Henceforth, whenever we refer to the Network Design problem (abbreviated as NDP), we will implicitly assume that it has this single, linear objective function.

The NDP can be formulated as a mixed-integer program in a variety of ways. In Section 1.3, we present two alternative formulations - an Arc-flow formulation and a Path-flow formulation - of this problem. We also discuss several variants of these two formulations. In Section 1.4, we briefly review the different methodologies that have been proposed in the literature for solving generic network design and closely related problems.

Characteristics of the linear programming relaxation:

As is typical of most large-scale mixed-integer programming models of this type, the linear programming relaxation of the problem plays a key role in most successful solution techniques. Understanding the structure of linear programming solutions is, therefore, desirable for both analytical and algorithmic purposes. In Chapter 2, we characterize the fractional extreme points of the linear programming relaxation of the Path-flow formulation for the Uncapacitated Network Design problem. This result generalizes the characterization for Plant Location problems developed by Cornuejols, Fisher and Nemhauser [1977b]. Based on this characterization, we derive two classes of inequalities, which when added to the linear programming relaxation of the NDP, exclude the current fractional extreme point solution.

Valid inequalities for the NDP:

The generation of facial and other valid inequalities for various network design related problems has been the subject of several recent studies. Adding such inequalities to the formulation strengthens the linear programming relaxation and significantly improves the computational

performance of algorithms that rely heavily on linear programming-based lower bound generation techniques. Formulating such inequalities for the NDP is the focus of Chapter 3. We transform the Path-flow formulation of the Uncapacitated NDP into a Set packing problem and draw upon results from the literature on facets for the Set packing polytope to construct three classes of valid inequalities for the NDP. We outline properties of "lifted" inequalities that are obtained in this manner and translate some of these lifted inequalities into valid constraints for the Arc-flow formulation of the NDP. We show that two of these classes of inequalities subsume the cuts derived earlier in Chapter 2. We also demonstrate how some of these inequalities can be obtained through a process of constraint aggregation in the original formulation. Finally, we discuss other types of valid inequalities that can be constructed for both the capacitated and uncapacitated versions of the NDP.

Dual Ascent algorithms for the NDP:

One of the most successful algorithms to date for solving general Uncapacitated Network Design problems is the composite Dual Ascent - Benders' decomposition procedure developed by Magnanti, Wong, and Mireault [1984]. Using a dual ascent scheme, they obtained extremely tight lower and upper bounds with relatively little computational effort; as a result, they were able to solve several large problems to optimality without extensive enumeration. Erlenkotter [1978] and Guignard and Spielberg [1979] have earlier reported a similar experience with dual ascent procedures for the Plant Location problem without and with side constraints, respectively. In Chapter 4, we develop a general framework for studying dual ascent procedures for the NDP. Based on this framework,

we discuss alternative schemes for the uncapacitated problem and extend the algorithms to handle arc capacity restrictions and other side constraints. We also describe several dual-based heuristic procedures and methods for 'reducing' the problem (i.e., fixing some binary variables to zero or one). We report computational experience for two types of randomly generated directed, uncapacitated network design problems using one particular implementation of the dual ascent scheme.

The Concave-cost Network flow problem:

An important problem that arises in the context of transportation planning is the LTL (Less-than-Truckload) Consolidation problem. This problem involves determining the optimal policy for consolidating Less-than-Truckload shipments in order to exploit transportation economies of scale. In Chapter 5, we cast this problem as a minimum cost network flow problem with piecewise linear, concave costs and show how it can be viewed as a special case of the NDP. We describe a Lagrangian-based algorithm that incorporates multiplier adjustment, subgradient optimization, heuristic improvement, and problem reduction procedures. This algorithm is related in many respects to the techniques developed in Chapter 4. We report computational experience using this composite algorithm for solving fairly large problems that have both general and special network topologies.

In summary, in this thesis, we address both modeling and algorithmic aspects of the Network Design problem. Much of the material that is outlined in Chapters 2, 3 and 4 can be viewed as generalizations of concepts that have been developed earlier for the Plant Location problem.

Indeed, the extensive literature on the Plant Location problem forms the basis for exposition and comparison throughout. In Chapter 6, we outline some possible extensions of the work reported in this thesis and the scope for further research, both theoretical and computational, in this area.

1.3 Formulations of the Network Design Problem

Most integer programming problems can usually be expressed mathematically in a variety of ways. Furthermore, even within the framework of the same basic formulation, numerous modeling variations are possible. Often, these variations are obtained by redefining variables, aggregating or disaggregating constraints and adding valid inequalities. In this section, we formally define the Network Design problem and describe the two alternative, but closely related, mixed-integer programming formulations of the problem that we will use subsequently - the ARC-FLOW formulation and the PATH-FLOW formulation. We begin by introducing some terminology and notation.

1.3.1 Problem definition:

Let G:(N,A) be the directed graph over which the NDP is defined. N is the set of nodes that serve as origins, destinations and transshipment points; A is the set of candidate arcs, i.e., arcs that may or may not be included in the network's design. We let K denote the set of commodities that must be transported over the network; these might be either distinct physical goods or the same good with different points of origin and

destination. In all our subsequent discussions, we will use the indices i and j to denote nodes of the given network, and the index k to denote commodities; the directed arc from node i to node j will be represented as (i,j).

For each commodity $k \epsilon K$, R_k units must be shipped from its origin node O(k) to its destination node D(k). In addition, the following problem parameters are given:

B_{ij} : the CAPACITY of arc (i,j), for all $(i,j) \epsilon A$.
Thus, if arc (i,j) is included in the network design, then the total flow of all commodities on this arc must not exceed B_{ij} units,

c^k_{ij} : the COST PER UNIT OF FLOW of commodity k on arc (i,j), for all $(i,j) \epsilon A$ and $k \epsilon K$, and

F_{ij} : the FIXED COST for using arc (i,j), for all $(i,j) \epsilon A$.

We will use the terms COST and LENGTH to mean any measure that has to be minimized. Also, we will assume throughout our discussions that the parameters c^k_{ij} and F_{ij} are non-negative.

The Network Design problem involves selecting a subset of arcs from the set of candidate arcs A and routing R_k units of each commodity $k \epsilon K$ from O(k) to D(k) along the selected arcs, subject to arc capacity and other side constraints, in order to minimize the sum of the fixed and routing costs. It is convenient to visualize the NDP as a two-stage decision process consisting of

(1) the NETWORK DESIGN decision:

This phase selects the subset of arcs to be used for transporting the commodities. This choice determines the total fixed costs that are incurred; it also restricts the number of alternative ways in which

each commodity can be routed. We will refer to the chosen subset of arcs as the Network Design.

(2) the ROUTING decision:

This stage determines the quantity of each commodity that must flow on each arc (or route) of the chosen design, thus determining the total routing costs incurred.

In order to represent this problem mathematically, we must define binary DESIGN variables, one for each arc, that indicate whether or not the arc is included in the network design. In addition, we need continuous FLOW variables for each commodity. These flow variables can be defined in several alternative ways. We next discuss two formulations - the ARC-FLOW formulation and the PATH-FLOW formulation - that define the flow variables differently. The first formulation treats the flow of each commodity on every arc as a decision variable, while the second formulation represents, for each commodity k, the flow on every path from O(k) to D(k) as a continuous variable.

1.3.2 The ARC-FLOW formulation:

Let $\{y_{ij}\}$ be a set of binary variables that model the discrete choice design decision, with

$$y_{ij} = \begin{cases} 1 \text{ if arc } (i,j) \text{ is included in the network design} \\ 0 \text{ otherwise} \end{cases}$$

for all arcs $(i,j) \epsilon A$. Let x^k_{ij} be a continuous variable denoting the flow of commodity k on arc (i,j), for each arc $(i,j) \epsilon A$ and every commodity $k \epsilon K$. Then, the Network Design problem, in Arc-Flow form, can be expressed as

[NDAF]

$$\text{Minimize} \quad \sum_k \sum_{(i,j)} c^k_{ij} x^k_{ij} + \sum_{(i,j)} F_{ij} y_{ij} \tag{1.1}$$

subject to

$$\sum_j x^k_{ij} - \sum_j x^k_{ji} = \begin{cases} +R_k & \text{if } i = O(k) \\ -R_k & \text{if } i = D(k) \\ 0 & \text{otherwise} \end{cases} \qquad \text{all } k \epsilon K \tag{1.2}$$

$$x^k_{ij} \leq d^k_{ij} y_{ij} \qquad \text{all } (i,j) \epsilon A,\ k \epsilon K \tag{1.3}$$

$$\sum_k x^k_{ij} \leq B_{ij} y_{ij} \qquad \text{all } (i,j) \epsilon A \tag{1.4}$$

$$x^k_{ij} \geq 0 \qquad \text{all } (i,j) \epsilon A,\ k \epsilon K \tag{1.5a}$$

$$y_{ij} = 0 \text{ or } 1 \qquad \text{all } (i,j) \epsilon A \tag{1.5b}$$

where $d^k_{ij} = \text{Min } \{R_k, B_{ij}\}$ for all $(i,j) \epsilon A$ and $k \epsilon K$.

For every commodity, constraints (1.2) are the flow conservation equations at each node. In these constraints, the summations in the left-hand side are (implicitly) restricted to indices j such that arc (i,j) and (j,i), respectively, belong to the set A. The 'forcing' constraints (1.3) ensure that no commodity is routed through arc (i,j) if this arc is not chosen as part of the network design, i.e., $x^k_{ij} = 0$ for all $k \epsilon K$ if $y_{ij} = 0$; on the other hand, if arc (i,j) is included in the design, there is an optimal solution of the problem in which no more than d^k_{ij} (= Min $\{R_k, B_{ij}\}$) units of commodity k flow on this arc (since the cost coefficients c^k_{ij} and F_{ij} are non-negative). Constraints (1.4) model the capacity restriction of arcs. Notice that these constraints, by themselves, force the flow to be zero on arcs that are not included in the design. Thus, the 'disaggregated' forcing constraints (1.3) are redundant in the integer programming formulation [NDAF]; furthermore, they increase the problem size significantly. However, these constraints are not

redundant in the linear programming relaxation of the problem; consequently, tneir addition generally yields tighter relaxations and hence sharper linear programming-based lower bounds for the problem. Davis and Ray [1969], Williams [1974], and Rardin and Choe [1979], among others, have advocated the use of such tighter disaggregated formulations for algorithmic development. As Magnanti and Wong [1981] have noted, an added advantage of using this type of formulation is that the enhanced set of linear programmming dual variables for the disaggregated constraints provides greater flexibility for ascent and cut-generation, thereby improving algorithmic performance. This advantage will become apparent when we discuss specific solution strategies for the NDP. We refer to the formulation [NDAF] with and without the disaggregated forcing constraints (1.3) as the STRONG and WEAK Arc-flow formulations, respectively.

1.3.3 The PATH-FLOW formulation:

In order to express the Network Design problem in terms of the flows along different paths of the network, we need the following additional notation.

For each commodity $k \epsilon K$, let $P(k)$ be the set of all possible paths from origin $O(k)$ to destination $D(k)$ in the given graph $G:(N,A)$. We will denote any arbitrary path belonging to this set as $p(k)$. Let

$P_{ij}(k) = \{p(k) \epsilon P(k) : (i,j) \epsilon p(k)\}$ for all $(i,j) \epsilon A$, and $k \epsilon K$.

Thus, $P_{ij}(k)$ is the subset of paths for commodity k that contain arc (i,j). Define $c_{p(k)}$ to be the COST PER UNIT FLOW of commodity k on path $p(k)$, i.e.,

$$c_{p(k)} = \sum_{(i,j)\epsilon p(k)} c^k_{ij} \qquad \text{for all } k\epsilon K \text{ and } p(k)\epsilon P(k).$$

Then, the PATH-FLOW formulation [NDPF] of the Network Design problem is

[NDPF]

$$\text{Minimize} \quad \sum_k \sum_{p(k)} c_{p(k)} f_{p(k)} + \sum_{(i,j)} F_{ij} y_{ij} \tag{2.1}$$

subject to

$$\sum_{p(k)} f_{p(k)} = R_k \qquad \text{all } k\epsilon K \tag{2.2}$$

$$f_{p(k)} \leq d^k_{ij} y_{ij} \qquad \text{all } k\epsilon K,\ (i,j)\epsilon A, \text{ and } p(k)\epsilon P_{ij}(k) \tag{2.3}$$

$$\sum_k \sum_{p(k)\epsilon P_{ij}(k)} f_{p(k)} \leq B_{ij} y_{ij} \qquad \text{all } (i,j)\epsilon A \tag{2.4}$$

$$f_{p(k)} \geq 0 \qquad \text{all } k\epsilon K,\ p(k)\epsilon P(k) \tag{2.5a}$$

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

where the decision variables $\{y_{ij}\}$ and $\{f_{p(k)}\}$ have the following interpretation:

$$y_{ij} = \begin{cases} 1 \text{ if arc}(i,j) \text{ is included in the network design} \\ 0 \text{ otherwise , and} \end{cases}$$

$f_{p(k)}$ = flow of commodity k on path p(k),

and the parameter $d^k_{ij} = \text{Min } \{R_k, B_{ij}\}$ for all $(i,j)\epsilon A$ and $k\epsilon K$.

As before, constraints (2.2) ensure that all demand requirements are met, inequalities (2.3) are the disaggregated forcing constraints and constraints (2.4) model the arc capacities. Notice, however, that the forcing constraints (2.3) are not as tight as the corresponding inequalities (1.3) of the formulation [NDAF]. We next discuss this issue and other relationships between the Arc-flow and Path-flow formulations. Finally, note that, when all the cost coefficients $c_{p(k)}$ and F_{ij} are non-negative, we need to consider only <u>simple</u> paths p(k) from O(k) to D(k)

in each of the sets P(k).

1.3.4 Similarities and differences between the two formulations:

The integer programming formulations [NDAF] and [NDPF] are equivalent in the sense that both formulations have the same optimal objective function value. This result follows from the fact that given any feasible path-flow vector $f = \{f_{p(k)}\}$, there is a unique corresponding arc-flow vector $x = \{x^k_{ij}\}$ that is obtained by setting

$$x^k_{ij} = \sum_{p(k) \in P_{ij}(k)} f_{p(k)} \quad \text{for all } (i,j) \in A \text{ and } k \in K.$$

Similarly, given any feasible arc-flow vector x, we can construct a corresponding (not necessarily unique) feasible path-flow vector f using flow decomposition. (See, for example, Ford and Fulkerson [1962].) Both the arc-flow solution and the path-flow solution have the same routing cost. Thus, given any mixed-integer solution (x,y) of [NDAF], we can construct a corresponding solution (f,y) of [NDPF] with the same objective function value. However, the linear programming relaxations (obtained by relaxing the integer restrictions on the y-variables) of the two formulations are not equivalent.

PROPOSITION 1:

Let z_{AF} and z_{PF} be the optimal objective function values of the linear programming relaxations of [NDAF] and [NDPF], respectively. Then,

$$z_{AF} \geq z_{PF}$$

i.e., the linear programming relaxation of [NDAF] is 'tighter' than that of [NDPF].

Proof:

Let (x,y) be an optimal solution of the linear programming relaxation of [NDAF] and let $f = \{f_{p(k)}\}$ be a path-flow vector (obtained by flow decomposition) corresponding to the given arc-flow vector $x = \{x^k_{ij}\}$. Then, the solution (f,y'), with

$$y'_{ij} = \text{Max} \left\{ \left(\max_k \max_{p(k) \in P_{ij}(k)} f_{p(k)} \right) / d^k_{ij}, \sum_k \sum_{p(k) \in P_{ij}(k)} f_{p(k)} / B_{ij} \right\}$$

is feasible in the linear programming relaxation of [NDPF]. Furthermore,

(a) $$\sum_k \sum_{(i,j)} c^k_{ij} x^k_{ij} = \sum_k \sum_{p(k)} c_{p(k)} f_{p(k)}$$

Now, for all arcs $(i,j) \in A$,

$$\begin{aligned} y_{ij} &\geq \text{Max} \left\{ \max_k x^k_{ij}/d^k_{ij}, \sum_k x^k_{ij}/B_{ij} \right\} \qquad \text{[from constraints (1.3) and (1.4)]} \\ &= \text{Max} \left\{ \max_k \sum_{p(k) \in P_{ij}(k)} f_{p(k)}/d^k_{ij}, \sum_k \sum_{p(k) \in P_{ij}(k)} f_{p(k)}/B_{ij} \right\} \\ &\geq \text{Max} \left\{ \max_k \max_{p(k) \in P_{ij}(k)} f_{p(k)}/d^k_{ij}, \sum_k \sum_{p(k) \in P_{ij}(k)} f_{p(k)}/B_{ij} \right\} \\ &= y'_{ij} \end{aligned}$$

implying that

(b) $$\sum_{(i,j)} F_{ij} y_{ij} \geq \sum_{(i,j)} F_{ij} y'_{ij} \qquad \text{[since } F_{ij} \geq 0 \text{ for all } (i,j) \in A\text{]}.$$

(a) and (b) together imply that

$$\begin{aligned} z_{AF} &= \sum_{(i,j)} F_{ij} y_{ij} + \sum_k \sum_{(i,j)} c^k_{ij} x^k_{ij} \\ &\geq \sum_{(i,j)} F_{ij} y'_{ij} + \sum_k \sum_{p(k)} c_{p(k)} f_{p(k)} \\ &\geq z_{PF}. \end{aligned}$$

Proposition 1 is a consequence of the fact that the forcing constraints (2.3) of [NDPF] are not as tight as the corresponding disaggregated constraints (1.3) of [NDAF]. Indeed, it is possible to strengthen formulation [NDPF] further by replacing constraints (2.3) by the more stringent, but valid, inequalities

$$\sum_{p(k)\in P_{ij}(k)} f_{p(k)} \leq d^k_{ij} y^k_{ij} \quad \text{for all } (i,j)\in A \text{ and } k\in K. \qquad (2.3a)$$

We will refer to the Path-flow formulation with constranits (2.3a) instead of (2,3) as the STRONG disaggregated Path-flow formulation of the Network Design problem. It is easy to show that the linear programming relaxation of this latter formulation has the same optimal value as the linear programming relaxation of [NDAF].

The Path-flow fomulation of the NDP is similar in structure to the Set packing problem. However, solving the NDP in Path-flow form would require the use of a column-generation technique, since even for small problems the number of possible paths for each commodity is likely to be very large. In the subsequent chapters, we will use the Path-flow formulation mainly for studying the structural properties of the NDP and for constructing valid inequalities. Most of the algorithmic development, on the other hand, will be in terms of the Arc-flow formulation.

As mentioned earlier, it is possible to express the NDP in ways that are different from the two formulations that we have discussed so far. For example, Peterson [1980] has proposed a Cut-flow formulation for the problem in which the total flow on each arc is a decision variable; the demand requirements are expressed in terms of minimum required flows across all cutsets of the given graph.

1.4 Variants of the Network Design Problem

In this section, we discuss some variants and special cases of the NDP

that are particularly relevant for our subsequent discussions. These include problems defined over uncapacitated and undirected networks, those with additional side constraints and the Plant Location problem. (For a comprehensive discussion of other variants, see Magnanti and Wong [1984a]).

(a) Uncapacitated networks:

We will refer to the NDP in which unlimited flow is permitted on all arcs that are included in the design as the UNCAPACITATED NETWORK DESIGN problem (abbreviated as UNDP). For this problem, the capacity constraints (1.4) and (2.4) of formulations [NDAF] and [NDPF] are not necessary. Alternatively, the weak formulation of this problem (without the disaggregated forcing constraints (1.3) and (2.3)) is obtained by setting the coefficient B_{ij} in constraint (1.4) equal to the sum of the demands, for all arcs (i,j). The formulation of the UNDP can be simplified further by scaling down all the flow variables corresponding to commodity k by the demand R_k and multiplying the corresponding flow cost coefficients (c^k_{ij} and $c_{p(k)}$) by R_k, for all commodities k. In terms of these new variables, the demand for each commodity is 1 unit and the parameter $d^k_{ij} = \text{Min } \{R_k, B_{ij}\}$ is 1 for all arcs (i,j) and commodities k. (Note that, when the arcs are capacitated, the coefficients in the capacity constraints (1.4) and (2.4) may become fractional or greater than one if the flow variables are scaled.) We will denote the transformed Arc-flow and Path-flow formulations of the UNDP as [UNDAF] and [UNDPF] respectively. The Uncapacitated Network design problem, unlike the capacitated version, always has an an optimal solution in which all flows are integral because the UNDP can be viewed as a Concave-cost Network flow problem (with just the flow conservation constraints) whose extreme points are all integral.

(See, for example, Zangwill [1968].) Consequently, the flow variables can also be restricted to be integers, thus converting the problem into a pure integer program. We exploit this property to construct valid inequalities for the problem in Chapter 3. Finally, under the assumption that all fixed costs are non-negative, the linear programming relaxation of the UNDP has an optimal solution in which all the design variables y_{ij} are less than or equal to 1. Therefore, the upper bound of 1 on all the y-variables need not be included in the linear programming relaxation of the UNDP; these upper bounds are, however, essential in the linear programming relaxation of the capacitated problem.

Most of the results that we develop in the next few chapters apply to the Uncapacitated Network Design problem. Whenever we specifically account for the arc capacity constraints, we will refer to the problem as the CAPACITATED NETWORK DESIGN problem (abbreviated as CNDP).

(b) Undirected networks:

In the UNDIRECTED NETWORK DESIGN problem, flow on each edge (i,j) is permitted in either direction once this edge is included in the network design. Also, B_{ij} is now interpreted as the maximum permissible total flow on edge (i,j) summed over both directions. In the Arc-flow formulation of this problem, we include two flow variables x^k_{ij} and x^k_{ji} for all commodities $k \epsilon K$, in order to account for flow in both directions on each undirected edge (i,j) of A; we require only one binary design variable corresponding to each undirected edge, however. Assuming that all the flow cost coefficients c^k_{ij} are non-negative, the problem has an optimal solution in which at least one of the two flow variables x^k_{ij} and x^k_{ji} is zero, for all

commodities k and edges (i,j). Hence, the Arc-flow formulation is obtained by replacing x^k_{ij} in the forcing constraints of the formulation [NDAF] with the sum $(x^k_{ij} + x^k_{ji})$. Similarly, for the Path-flow formulation of this problem, we let P(k) be the set of all undirected paths from O(k) to D(k), for all commodities $k \epsilon K$. The variable $f_{p(k)}$ is defined as the flow of commodity k on the undirected path p(k). Then, the formulation [NDPF] models the Undirected Network Design problem in path-flow form. In general, for any two commodities k' and k" that do not share a common origin or destination, both $x^{k'}_{ij}$ and $x^{k''}_{ji}$ could be positive for some edge (i,j) of the graph in an optimal solution of the Undirected Network Design problem. Consequently, the Undirected Network Design problem cannot, in general, be viewed as a special case of the directed version. In contrast, for problems such as the Steiner tree problem, the Traveling Salesman problem and the Shortest path problem, the undirected version can be transformed into an equivalent problem defined over a directed graph. In this thesis, we will be mainly concerned with Network Design problems defined over directed graphs.

(c) Additional side constraints:

In many applications, the network design and the flow profile must satisfy some additional constraints besides the flow conservation equations and the forcing constraints. For instance, in the BUDGET DESIGN problem, the fixed cost of building the network is incorporated as a budget constraint rather than as a term in the objective function. If B is the total budget available for construction, then this constraint takes the form

$$\sum_{(i,j)} F_{ij} y_{ij} \leq B .$$

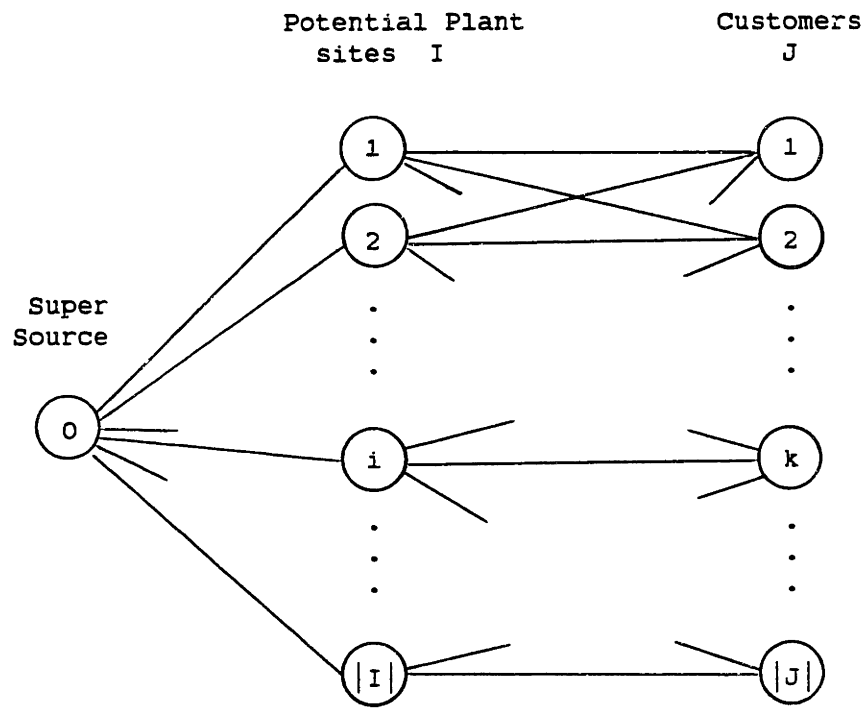
Similarly, to ensure that arc (i,j) is always included in the design, we must add the constraint $y_{ij} = 1$ to the formulation. Problems of this type have often been referred to in the literature as NETWORK IMPROVEMENT models. Other side conditions might include constraints on the usage of common resources, precedence constraints, logical and multiple choice restrictions, degree constraints and other topological conditions imposed upon the configuration of the network. In Chapters 2 and 3 we discuss additional constraints that are added to strengthen the linear programming relaxation of the NDP.

(d) The Plant Location problem:

Of all the special cases of the NDP, the Plant Location problem [PLP] is of particular interest to us because, like the NDP, the central issue that it addresses is the tradeoff between the investment and operating costs. Many of the results and algorithms that we develop in subsequent chapters are generalizations of analagous results derived in the literature for the Plant Location problem. The PLP involves selecting a subset of plants, from a list I of potential sites, that should be built in order to satisfy the demand of customers belonging to the set J. The PLP can be represented as a Network Design problem in the following manner: add a supersource, designated as node 0, and arcs (0,i) from this source to each of the plant sites $i \in I$. The fixed cost for opening the i^{th} facility is the fixed cost for including arc (0,i) in the design, and the cost of supplying 1 unit from plant i to customer j is the flow cost of the corresponding arc (i,j) in the network. There are $|J|$ commodities, one corresponding to each customer. Node 0 serves as the origin for all commodities, and the commodities are indexed so that the destination for commodity k is the

demand (or customer) node $k \epsilon J$. Figure 1 illustrates the construction of the network corresponding to a PLP.

Figure 1: The Plant Location problem as a Network Design problem



Commodity k has origin O(k) = 0 and destination D(k) = k.

We can model plant capacities by imposing flow capacity constraints on the corresponding arcs (0,i). On the other hand, if the plants are uncapacitated, we can, as before, scale the flow variables so that the demand for each customer is 1 unit. This version of the problem is often referred to as the Simple Plant Location problem. We will be mainly concerned with this uncapacitated problem in our subsequent discussions.

Because of the special structure of the network over which the PLP is defined, its Arc-flow and Path-flow formulations are equivalent even in terms of their respective linear programming relaxations. This equivalence follows from the fact that all paths for each commodity $k \epsilon J$ are of the form 0-i-k, $i \epsilon I$. Thus, $f_{i(k)}$, the flow of commodity k on path 0-i-k, is equal to x^k_{0i}, the number of units of commodity k routed on arc (0,i).

So far, we have described four variants and special cases of the Network Design problem that we will refer to later on in this thesis. However, as mentioned before, there are numerous other well-known optimization problems, such as the Steiner tree problem and the Traveling Salesman problem, that arise as special cases of the NDP when the cost structure, the demand pattern, and the network topology are specialized in various ways. We next briefly review the Network Design literature.

1.5 Literature Review

In this section, we sketch the evolution of the literature on the generic Network Design problem and its variants. We emphasize the main methodological contributions rather than specific applications. The general Network Design model is the logical extension of a long hierarchy of design problems that includes the Fixed-charge transportation problem, various facility location models, and the Budget Design problem.

The history of these problems dates back to 1954, when Hirsch and Dantzig [1954] formulated the fixed-charge problem with linear constraints

and showed that the optimal solution occurs at an extreme point of the convex set of feasible solutions. Intrigued by the similarity between this problem and linear programming problems, various researchers devised heuristic and optimization procedures which exploit this extreme point property. Cooper and Drebes [1967], Denzler [1969], Steinberg [1970], Walker [1976], and others developed heuristic adjacent extreme point algorithms for this problem; all these methods identify locally optimal solutions. Murty [1968], and McKeown [1975] proposed vertex-ranking procedures to obtain optimal solutions and Steinberg [1970] outlined a branch-and-bound scheme.

The Fixed-charge Transportation problem was one of the first network specializations of the general fixed-charge problem to be considered in the literature. Algorithms designed specifically for this problem include heuristics by Balinski [1961], and Kuhn and Baumol [1962], and exact approaches, all based on branch-and-bound, by Gray [1971], Kennington and Unger [1976], and Barr et al. [1981]. Gray developed the first optimizing code for the Fixed-charge Transportation problem; subsequently, Kennington and Unger and Barr et al. designed more efficient algorithms that further exploit the special structure of the transportation problem for generating lower and upper bounds. Malek-Zaverei and Frisch [1972] showed how any Fixed-charge Transshipment problem can be converted into an equivalent Fixed-charge Transportation problem defined on a modified network with (m+n) nodes and 2m arcs (where m and n are the number of arcs and nodes, respectively, in the original problem).

Facilities location has long been an area of fertile research and

continues to be of central interest. The literature in this area is very extensive; Francis and White [1974], Handler and Mirchandani [1979], and Tansel et al. [1983] have surveyed the numerous problem types and algorithms that have been proposed in this area. All the well-known large-scale integer programming techniques - Branch-and-bound (e.g., Davis and Ray [1969], Akinc and Khumawala [1977]), Benders' decomposition (e.g., Geoffrion and Graves [1974]), Dual Ascent (e.g., Erlenkotter [1978], Guignard and Spielberg [1979]), and Lagrangian relaxation (e.g., Geoffrion and McBride [1978]) - have been used in the Facility Location context. Cornuejols et al. [1977b] have derived worst-case bounds for a greedy heuristic for the Uncapacitated Plant Location problem. Recently, researchers have focused attention on deriving facets and valid inequalities that strengthen the formulations of some facility location models (e.g., Guignard [1980], Cornuejols and Thizy [1982]).

Fixed-charge Network Design and Budget Design problems have been studied by numerous researchers, primarily in the context of transportation planning. The NDP and several of its special cases - such as the Steiner tree problem (Garey and Johnson [1979]) and the Uncapacitated Budget Design problem (Johnson et al. [1980]) - are known to be NP-hard. Wong [1980] has shown that even the problem of finding ϵ-optimal solutions to a Budget Design problem with unit demands is NP-hard. Heuristics incorporating Add, Drop, Interchange, and Flow deviation procedures have been proposed by Scott [1969], Dionne and Florian [1979], Boffey and Hinxman [1979], Billheimer and Gray [1973], Los and Lardinios [1982], Gerla and Kleinrock [1977], and Gallo and Sodini [1979] for approximately solving uncapacitated design problems. Most of the algorithms proposed in the literature for

solving the NDP optimally are based on branch-and-bound. These include the algorithms discussed by Hoang [1973], Boyce et al. [1973], Dionne and Florian [1979], LeBlanc [1975], and Los and Lardinios [1982]. These procedures differ primarily in the way in which they obtain lower bounds; Magnanti and Wong [1984a] have interpreted the different bounding methods as alternative Benders' cut generation schemes. The special structure of the NDP makes the use of large-scale decomposition techniques - such as Lagrangian relaxation and Benders' decomposition - especially attractive. Geoffrion [1977] has outlined a Tandem Lagrangian relaxation scheme for obtaining good lower bounds and Magnanti and Wong [1984b] have proposed a Dual Ascent scheme for this purpose. Magnanti et al. [1984] have incorporated this scheme within the framework of a Benders' decomposition procedure for solving the Uncapacitated Network Design problem. Erickson et al. [1981] have developed a dynamic programming algorithm for solving general concave-cost network problems. Except for Geoffrion [1977], most other researchers have focused on the uncapacitated version of the NDP.

This review of solution techniques for generic network design related models is not exhaustive. For example, we have not discussed algorithms that have been devised in the context of specific applications and for solving some important special cases of the NDP such as the Traveling Salesman, Vehicle Routing and Production lot sizing problems. Also, our emphasis has been on algorithmic rather than analytical results, such as those dealing with worst-case analysis and the structure of fixed-charge polytopes. In subsequent chapters we will review some additional results that are relevant for our later discussions.

In Chapters 2 and 3 we study the structure and properties of the integer and linear programming polytopes of the Uncapacitated Network Design problem. Most of the results that we discuss are extensions and generalizations of similar properties discussed in the literature for the Plant Location problem.

CHAPTER 2

FRACTIONAL EXTREME POINTS OF THE UNCAPACITATED NETWORK DESIGN PROBLEM

The linear programming relaxation plays an important role in most of the successful algorithms for large-scale mixed-integer programming problems like the Network Design problem. Invariably, these algorithms rely heavily on obtaining good lower bounds and, in most instances, the lower bounding techniques that are used can be related to the optimal LP solution. For instance, when Lagrangian relaxation is used for deriving lower bounds, if the corresponding Lagrangian subproblem satisfies the Integrality property (Geoffrion [1974]), the best lower bound that can be obtained is equal to the optimal value of the LP relaxation. Similarly, in the Benders' decomposition scheme, cuts for the master problem can be generated using optimal solutions of the LP dual. For the NDP, in particular, Magnanti and Wong [1984a] have shown that many of the lower bounding inequalities outlined in the literature can be interpreted as Benders' cuts derived from the dual solution. Finally, Erlenkotter [1978] and Magnanti et al. [1984] have observed that Dual Ascent, a technique for solving the LP dual approximately, yields extremely tight lower bounds for the Plant Location and Uncapacitated Network Design problems. Therefore, an understanding of the structure and properties of the LP relaxation for such problems could potentially contribute toward improving algorithmic performance significantly. For instance, suppose we are able to devise, using a characterization of LP extreme points, a procedure that generates

valid inequalities that cut off any given fractional extreme point. We can then use this procedure to iteratively improve the LP lower bound by adding, to the original formulation, valid, violated inequalities at each stage. This idea of iteratively strengthening the LP relaxation forms the basis of traditional Cutting plane and Group theoretic methods (see, for instance, Garfinkel and Nemhauser [1972] and Shapiro [1979]) for solving integer programming problems. Recently, Martin and Schrage [1982] have proposed a convergent algorithm for mixed-integer programming problems, based on the same principle, that relies on a constraint aggregation and coefficient reduction procedure for generating cuts. Crowder et al. [1983] have successfully employed a similar technique for solving large-scale zero-one integer programming problems.

In this chapter, we characterize fractional extreme points for the Path-flow formulation of the Uncapacitated Network Design problem. Based on this characterization, we construct two classes of valid inequalities which, under certain circumstances, eliminate the corresponding fractional extreme point. Our development closely parallels the approach adopted by Cornuejols, Fisher and Nemhauser [1977b] for characterizing fractional extreme points of the uncapacitated Plant Location problem. Indeed, we show that many of our results are generalizations of similar properties outlined by Cornuejols et al. In Chapter 3, using a Set packing equivalent of the Path-flow formulation, we derive several additional and more general families of valid cuts for the UNDP that contain the inequalities discussed in this chapter.

2.1 A Characterization of Fractional LP Extreme Points

Consider the Path-flow formulation [UNDPF] of the Uncapacitated Network Design problem (with 'scaled' unit demands), which is repeated below for convenience.

[UNDPF]

$$\text{Minimize} \quad \sum_{k} \sum_{p(k)} c_{p(k)} f_{p(k)} + \sum_{(i,j)} F_{ij} y_{ij} \tag{2.1}$$

subject to

$$\sum_{p(k)} f_{p(k)} = 1 \qquad \text{all } k \in K \tag{2.2}$$

$$f_{p(k)} \leq y_{ij} \qquad \text{all } k \in K,\ (i,j) \in A,\ p(k) \in P_{ij}(k) \tag{2.3}$$

$$f_{p(k)} \geq 0 \qquad \text{all } k \in K,\ p(k) \in P(k) \tag{2.5a}$$

$$y_{ij} \geq 0 \qquad \text{all } (i,j) \in A \tag{2.5b}$$

$$y_{ij} \text{ integer} \qquad \text{all } (i,j) \in A \tag{2.5c}$$

where $P(k)$ is the set of all paths $p(k)$ for commodity k (from origin $O(k)$ to Destination $D(k)$),

$P_{ij}(k)$ is the subset of paths for commodity k that contain arc (i,j),

y_{ij} is the design variable denoting whether or not arc (i,j) is included in the network design, and

$f_{p(k)}$ is the flow of commodity k on path $p(k)$.

Notice that, since the cost coefficients $c_{p(k)}$ and F_{ij} are non-negative by assumption, we have omitted the upper bounding constraints

$$y_{ij} \leq 1 \qquad \text{for all } (i,j) \in A$$

in this formulation.

The LP relaxation of [UNDPF] is obtained by relaxing the integrality

constraints (2.5c). We let [L1] denote this LP relaxation and let P_{L1} be its feasible region, i.e.,

$$P_{L1} = \{(f,y) : (f,y) \text{ satisfies (2.2) to (2.5b)}\}.$$

As mentioned earlier, since [UNDPF] involves minimizing a concave function, some integer extreme point of the polytope P_{L1} is optimal for the original IP problem. The LP optimum, however, need not necessarily occur at this extreme point. Our objective, therefore, is to characterize (i.e., derive necessary and sufficient conditions for) fractional extreme points of P_{L1}.

We will argue as follows: Suppose (f,y) is a given fractional solution of [L1]. We will attempt to construct two other feasible solutions (f^+,y^+) and (f^-,y^-) with

$$f^+_{p(k)} = f_{p(k)} + g_{p(k)},\ f^-_{p(k)} = f_{p(k)} - g_{p(k)} \quad \text{for all } k\epsilon K \text{ and } p(k)\epsilon P(k),$$
$$y^+_{ij} = y_{ij} + h_{ij}\ ,\ y^-_{ij} = y_{ij} - h_{ij} \quad \text{for all } (i,j)\epsilon A, \tag{2.6}$$

where $g = \{g_{p(k)}\}$ and $h = \{h_{ij}\}$ are two suitably chosen perturbation vectors that are unrestricted in sign. Observe that, by construction,

$$(f,y) = 1/2(f^+,y^+) + 1/2(f^-,y^-).$$

Therefore, the given solution (f,y) is an extreme point of the polytope P_{L1} if and only if our assumptions imply that

$$(f,y) = (f^+,y^+) = (f^-,y^-),$$

i.e., if and only if $g = 0$, $h = 0$ is the only possible "feasible" perturbation vector with respect to the solution (f,y). Our strategy, then, is to find necessary and sufficient conditions that ensure this condition. Before presenting the main result, we require some additional definitions and preliminary results.

Definition 2.1:

Given a feasible solution (f,y) of [L1], an arc $(i,j) \epsilon A$ is said to be TIGHT for some path $p(k) \epsilon P_{ij}(k)$ containing this arc if $y_{ij} = f_{p(k)}$ in the solution (f,y).

Lemma 1:

Let (f,y) be an extreme point of P_{L1}. Then, for every arc $(i,j) \epsilon A$,

$$y_{ij} = \max_{k} \max_{p(k) \epsilon P_{ij}(k)} f_{p(k)} \tag{2.7}$$

i.e., every arc (i,j) is tight for at least one of the paths p(k) that contain this arc in any extreme point of P_{L1}.

Proof:

Suppose not, i.e., suppose that (2.7) is not valid for some arc (i',j'). Let (f,y^1) and (f,y^2) be defined by setting $y^1_{ij} = y^2_{ij} = y_{ij}$ for all arcs $(i,j) \neq (i',j')$, and $y^1_{i',j'} = y_{ij} + \epsilon$ and $y^2_{i',j'} = y_{ij} - \epsilon$. Then, for some $\epsilon > 0$, both (f,y^1) and (f,y^2) are feasible in [L1]. Since $(f,y) = 1/2((f,y^1) + (f,y^2))$, (f,y) is not an extreme point of P_{L1}.

Lemma 2:

Let (f,y) be <u>any</u> feasible solution of [L1] and let (f^+,y^+) and (f^-,y^-) be two other perturbed feasible solutions that are constructed as in (2.6). If arc $(i,j) \epsilon A$ is tight for some path p(k) (containing this arc) in the solution (f,y), then it is necessarily tight for p(k) in the solutions (f^+,y^+) and (f^-,y^-) as well, i.e., if $y_{ij} = f_{p(k)}$ for some $k \epsilon K$ and $p(k) \epsilon P_{ij}(k)$, then $y^+_{ij} = f^+_{p(k)}$ and $y^-_{ij} = f^-_{p(k)}$.

Notice that Lemma 2 does not require the solution (f,y) to be an

extreme point of P_{L1}. Also, the result applies to the following more general situation: Let $X = \{x_j\}$ be any given solution to the set of inequalities $AX \leq b$ and consider two other perturbed solutions $X^+ = X + e$ and $X^- = X - e$, where the perturbation vector $e = \{e_j\}$ is chosen to ensure the feasibility of X^+ and X^-. Then, if any constraint $a^r X \leq b_r$ of the set $AX \leq b$ is tight (or binding) with respect to the solution X, it must necessarily be binding for the solutions X^+ and X^- as well. (Since $a^r X^+ \leq b_r$ and $a^r X^- \leq b_r$, with one inequality holding strictly, implies that $a^r(1/2[X^+ + X^-]) = a^r X < b_r$). For Lemma 2, $a^r X \leq b_r$ becomes $f_{p(k)} \leq y_{ij}$.

Definition 2.2:

> Two paths p(k) and p'(k') are said to be LINKED with respect to a given solution (f,y) if they share a common arc (i,j) that is tight (in the solution (f,y)) for both the paths, i.e., if (i,j) belongs to p(k) and p'(k'), and $f_{p(k)} = y_{ij} = f_{p'(k')}$.
>
> A path p(k) is UNLINKED with respect to (f,y) if it is not linked to any other path.

Lemma 2 implies that, for the perturbed solutions (f^+,y^+) and (f^-,y^-) to be feasible, the perturbation vectors g and h must be such that, if arc (i,j) is tight for path p(k), then

$$h_{ij} = g_{p(k)}. \tag{2.8}$$

Consequently, for any two paths p(k) and p'(k') that are linked in the given solution,

$$g_{p(k)} = g_{p'(k')}. \tag{2.9}$$

Now, by Lemma 1, in any extreme point solution, every arc (i,j) is tight for at least one path that contains it. Therefore, for any given vector of flow perturbations, condition (2.8) completely determines the perturbation

vector h, when the solution (f,y) is an extreme point of P_{L1}.

In addition to the above restrictions, the perturbation vectors g and h must satisfy the following two conditions in order to ensure that the perturbed solutions (f^+,y^+) and (f^-,y^-) are feasible.

(a) Since the perturbed solutions are required to be non-negative,

$$|g_{p(k)}| \leq f_{p(k)} \quad \text{for all } k\epsilon K \text{ and } p(k)\epsilon P_{ij}(k), \text{ and}$$

$$|h_{ij}| \leq y_{ij} \quad \text{for all } (i,j)\epsilon A. \tag{2.10a}$$

In particular,

$$g_{p(k)} = 0 \quad \text{if} \quad f_{p(k)} = 0, \text{ and}$$

$$h_{ij} = 0 \quad \text{if} \quad y_{ij} = 0. \tag{2.10b}$$

(b) (f,y) is a feasible solution for [L1]; hence,

$$\sum_{p(k)} f_{p(k)} = 1 \quad \text{for all } k\epsilon K.$$

To ensure that the flow vectors f^+ and f^- also satisfy the demand requirements (constraints (2.2) of [UNDPF]), we must impose the condition

$$\sum_{p(k)} g_{p(k)} = 0 \quad \text{for all } k\epsilon K, \tag{2.11}$$

on the perturbation vector g.

We now express these restrictions on g and h in the context of an Auxiliary graph G_a and a related matrix R that are defined below. This construction enables us to exploit the conditions to develop the desired characterization.

Definition 2.3:

A path p(k) is said to be a FLOW CARRYING PATH if $f_{p(k)} > 0$ in the given solution (f,y).

The AUXILIARY GRAPH $G_a:(V_a,E_a)$:

Given any feasible solution (f,y) of the LP relaxation [L1], the corresponding AUXILIARY GRAPH $G_a:(V_a,E_a)$ is an Undirected graph that is constructed in the following manner:

- the vertex set V_a contains one vertex, designated as vertex [p(k)], corresponding to each flow carrying path p(k) of the original graph G.
- for any two vertices [p(k)] and [p'(k')] belonging to V_a, the edge set E_a contains the edge ([p(k)],[p'(k')]) if the corresponding paths p(k) and p'(k') are linked.

The matrix R:

Let M be the number of components (maximal connected subgraphs) in the Auxiliary graph G_a corresponding to a given solution (f,y). We will use the index m to identify components of G_a, and we will denote the m^{th} component of G_a as S_m, for $1 \leq m \leq M$. The matrix R contains one row for each commodity $k \in K$ and one column for each component S_m, for $1 \leq m \leq M$; the element r_{km} in the k^{th} row and m^{th} column of matrix R is defined as follows:

r_{km} = the number of vertices [p(k)] corresponding to commodity k in component S_m of G_a.

The Auxiliary graph G_a and the matrix R allow us to express the restrictions on the perturbation vectors g and h in the following more

convenient form.

(1) By condition (2.10b), $g_{p(k)}$ must be zero if path p(k) does not carry any flow in the given solution. Hence, $g_{p(k)}$ can be nonzero only if the corresponding vertex [p(k)] belongs to the vertex set V_a.

(2) $g_{p(k)}$ must be equal to $g_{p'(k')}$ if the Auxiliary graph G_a contains the edge ([p(k)],[p'(k')]). (This condition follows from (2.9) and the fact that the vertices [p(k)] and [p'(k')] are adjacent in G_a only if the corresponding paths p(k) and p'(k') are linked.) Therefore, if any two vertices [p(k)] and [p'(k')] are connected in G_a (i.e., if there is a path from [p(k)] to [p'(k')] in G_a), then

$$g_{p(k)} = g_{p'(k')}.$$

Consequently, for all vertices [p(k)] that belong to the same component S_m of G_a, the corresponding flow perturbations $g_{p(k)}$ must have the same value. We will refer to this common perturbation value for all the vertices belonging to component S_m as the PERTURBATION FLOW ASSOCIATED with component S_m and denote it as e_m.

(3) By definition, r_{km} is the number of vertices [p(k)] corresponding to commodity k in component S_m. Based on the above observations, we can express the equations (2.11) as

$$\sum_{m=1}^{M} r_{km} e_m = 0 \qquad \text{for all } k \epsilon K. \tag{2.12}$$

If the columns of the matrix R are linearly independent then the unique solution for system of equations (2.12) is e = 0.

We now present the main result.

PROPOSITION 2:

Any given solution (f,y) of [L1] is an extreme point of P_{L1} if and only if

(a) $y_{ij} = \underset{k}{\text{Max}} \underset{p(k)\epsilon P_{ij}(k)}{\text{Max}} f_{p(k)}$ for all $(i,j)\epsilon A$, and

(b) the matrix R corresponding to the solution (f,y) has linearly independent columns.

Proof:

We have already shown (in Lemma 1) that, if

$$y_{ij} > \underset{k}{\text{Max}} \underset{p(k)\epsilon P_{ij}(k)}{\text{Max}} f_{p(k)} \quad \text{for some arc } (i,j)\epsilon A,$$

then (f,y) is not an extreme point of P_{L1}. We will now show that any solution (f,y) that satisfies (a) is an extreme point of P_{L1} if and only if (b) is satisfied. As mentioned earlier, our strategy will be to construct two other feasible solutions $(f^+,y^+) = (f+g,y+h)$ and $(f^-,y^-) = (f-g,y-h)$ using appropriately chosen perturbation vectors $g = \{g_{p(k)}\}$ and $h = \{h_{ij}\}$. Then, the given solution (f,y) is an extreme point if and only if $g = 0$, $h = 0$ is the unique vector for which the solutions (f^+,y^+) and (f^-,y^-) feasible.

First, note that since (f,y) satisfies (a), for any given vector $g = \{g_{p(k)}\}$, the values of h_{ij} are uniquely determined, by Lemma 2, as

$$h_{ij} = g_{p(k)} \quad \text{for all } p(k)\epsilon T_{ij},$$

where T_{ij} is the set of all paths for which arc (i,j) is tight in the solution (f,y).

Suppose the columns of the matrix R are linearly independent. Then, $e_m = 0$, for $1 \leq m \leq M$, is the unique solution to the system of

equations (2.12). Consequently, by the arguments that preceded this proposition, $g = 0$ and $h = 0$ is the unique 'feasible' perturbation vector; therefore, (f,y) is an extreme point of P_{L1}. On the other hand, if the columns of R are linearly dependent, then there exists a nonzero solution $e = \{e_m\}$ to the system (2.12). We can then find a multiplier λ that is small enough, yet positive, such that the nonzero perturbation vectors g and h with

$$g_{p(k)} = \begin{cases} 0 \text{ if } [p(k)] \notin V_a \\ \lambda e_m \text{ if vertex } [p(k)] \text{ belongs to component } S_m \text{ of } G_a \end{cases}$$

$$h_{ij} = g_{p(k)} \quad \text{for all } p(k) \epsilon T_{ij},$$

are feasible, i.e., the corresponding perturbed solutions $(f^+,y^+) = (f+g,y+h)$ and $(f^-,y^-) = (f-g,y-h)$ are contained in the polytope P_{L1}. Since

$$(f^+,y^+) \neq (f,y) \neq (f^-,y^-) \text{ and}$$

$$(f,y) = 1/2\{(f^+,y^+) + (f^-,y^-)\},$$

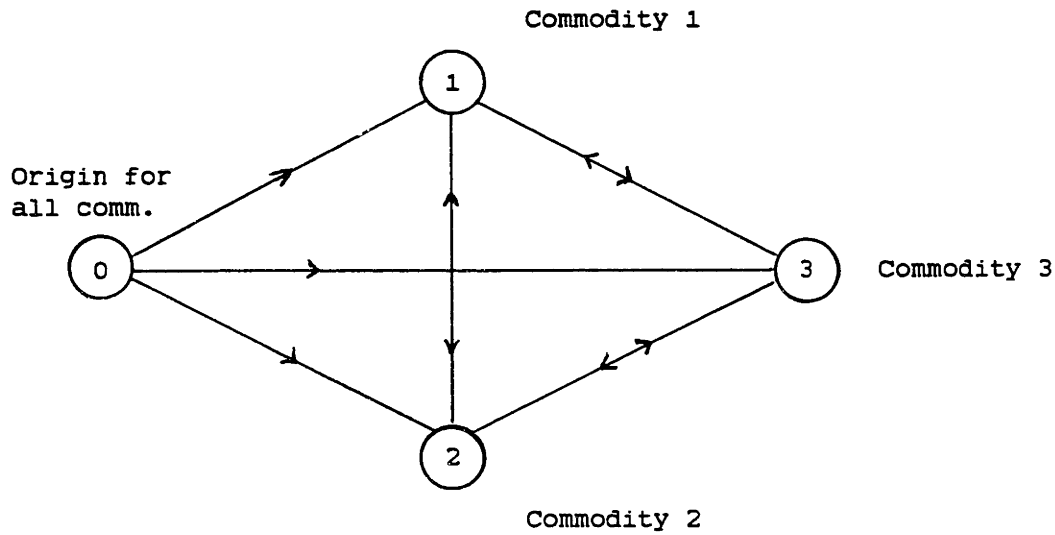
the given solution (f,y) is not an extreme point of P_{L1}.

Next, we will discuss an example that illustrates this characterization.

Example: LP extreme point characterization

Consider the 3 commodity Uncapacitated Network Design problem that is defined over the network shown in Figure 2. The 3 commodities, designated as commodities 1, 2 and 3, flow from node 0 to nodes 1, 2 and 3,

Figure 2: Example for LP extreme point characterization



respectively. For q = 1,2,3, and k = 1,2,3, we let pq(k) denote the path 0-q-k for commodity k. Thus, p3(2) is the path 0-3-2 for commodity 2 and p3(3) denotes the path 0-3. (Each commodity has other paths containing more than one intermediate node; however, we do not consider such paths in our illustration.)

First, let us determine using Proposition 2 whether the solution

$$f_{p1(1)} = 1/2 \qquad f_{p1(2)} = 1/2 \qquad f_{p1(3)} = 1/2$$

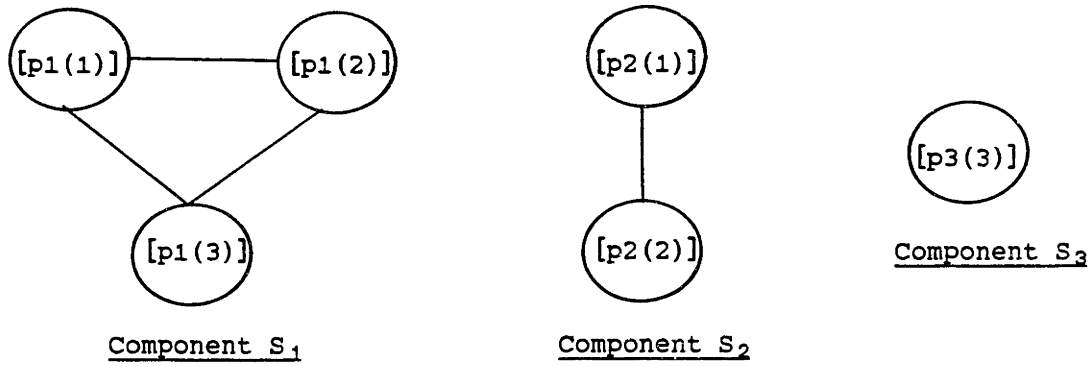
$$f_{p2(1)} = 1/2 \qquad f_{p2(2)} = 1/2 \qquad f_{p3(3)} = 1/2$$

$$y_{01} = y_{02} = y_{03} = y_{12} = y_{21} = 1/2$$

is an extreme point of the polytope P_{L1}. The arc (1,2) is contained in and is tight (with respect to this solution) for the three flow carrying paths p1(1), p1(2), and p1(3). Therefore, the three vertices [p1(1)], [p1(2)],

and [p1(3)] are adjacent in the corresponding Auxiliary graph G_a. Similarly, paths p2(1) and p2(2) are linked (since arc (1,3) is tight for both of them); path p3(3), however, is unlinked. For this solution, therefore, the Auxiliary graph G_a is

Figure 2A: Auxiliary graph for the given LP solution



Component S_1 Component S_2 Component S_3

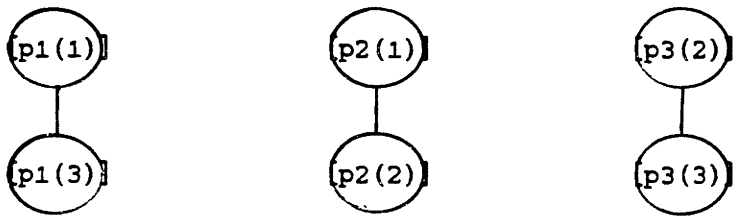
The corresponding matrix R is

$$R = \begin{vmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{vmatrix}.$$

Clearly, the above solution satisfies part (a) of Proposition 2; however, the columns of R are not linearly independent. Hence, the given solution is not an extreme point of the LP polyhedron.

Consider now the modified solution obtained by routing 1/2 unit of commodity 2 on path p3(2) (i.e., the path 0-3-2) instead of path p1(2). Figure 2B shows the Auxiliary graph for this solution.

Figure 2B: Auxiliary graph for the revised LP solution



Component S_1 Component S_2 Component S_3

The corresponding matrix R is

$$R = \begin{vmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{vmatrix}.$$

Notice that this matrix is a cyclic matrix with 2 consecutive 1's in each row; its columns are linearly independent. Hence, by Proposition 2, this revised LP solution is an extreme point of the polytope P_{L1}.

Variants of Propostion 2:

We will now examine some of the variants and implications of Proposition 2. Recall that, a path p(k) is said to be unlinked if none of the arcs that are tight for this path are tight for any other path, i.e., p(k) is not linked to any other path. (In particular, if no arc of path p(k) is tight for the path, then p(k) is unlinked.) Therefore, any vertex [p(k)] in V_a that corresponds to an UNLINKED path p(k) is an isolated node of the Auxiliary graph G_a, i.e., vertex [p(k)] constitutes a separate component, say S_m, of G_a. Correspondingly, the m^{th} column of the matrix R is the k^{th} unit vector e^k of $R^{|K|}$. The following two corollaries of

Proposition 2 are then immediate.

Corollary 2.1:

If (f,y) is an extreme point of P_{L1}, then each commodity $k \epsilon K$ has at most one flow carrying, unlinked path.

Corollary 2.2:

If (f,y) is a fractional extreme point of P_{L1}, then there is at least one commodity $k \epsilon K$ that has no flow carrying, unlinked path.

Proof:

If every commodity has a flow carrying, unlinked path, the corresponding nodes of G_a constitute $|K|$ separate components. Furthermore, since (f,y) is fractional, at least one commodity must flow on two or more paths. Hence, G_a must contain at least 2 vertices corresponding to this commodity, and one of these vertices does not belong to these $|K|$ isolated components. Therefore, G_a has more than $|K|$ components, implying that the columns of R are linearly dependent - a contradiction if (f,y) is an extreme point.

In fact, we can prove the following stronger result (using Corollary 2.3 discussed next).

Corollary 2.2a:

Let (f,y) be a fractional extreme point of P_{L1} and let K_F be the (nonempty) set of commodities with fractional flows. Then, least one commodity $k \epsilon K_F$ does not have any flow carrying unlinked path.

Next, we will express the characterization of Proposition 2 in terms of two successively smaller submatrices, R_F and R_L, of the matrix R. This reformulation will enable us to relate the results developed so far in this section to the characterization of fractional extreme points for the Plant Location problem discussed by Cornuejols et al. [1977b].

For any given solution (f,y) of the LP relaxation [L1], let

K_I = set of commodities with integer flows in the solution (f,y), and

$K_F = K \setminus K_I$.

Notice that G_a contains exactly one vertex, say $[p^*(k)]$, corresponding to each commodity $k \epsilon K_I$. Let $S_{m(k)}$, for all $k \epsilon K_I$, be the unique component of G_a containing vertex $[p^*(k)]$. For any $k \epsilon K_I$, the subset of commodities $\{k' \epsilon K : r_{k'm(k)} \neq 0\}$ must necessarily be contained in K_I (since, vertex $[p(k')]$ belonging to $S_{m(k)}$ implies that $f_{p(k')} = f_{p^*(k)} = 1$). Now, consider the submatrix R_F obtained from R by deleting row k and column m(k), for all $k \epsilon K_I$. By construction, the columns of R are linearly independent if and only if the columns of the submatrix R_F are linearly independent, leading to the following corollary of Proposition 2.

Corollary 2.3:

Any fractional solution (f,y) of [L1] is an extreme point of P_{L1} if and only if

(a) $y_{ij} = \max_k \max_{p(k) \epsilon P_{ij}(k)} f_{p(k)}$ for all $(i,j) \epsilon A$, and

(b) the columns of the submatrix R_F corresponding to the given solution (f,y) are linearly independent.

As an aside, we note that if (f,y) is an integer solution (i.e., if all flows $f_{p(k)}$ and design variables y_{ij} are either 0 or 1), then it is an extreme point of P_{L1} if and only if the solution satisfies condition (a) of Corollary 2.3.

We next consider a submatrix R_L of R_F that is defined as follows: Let

K_L = index set of commodities with fractional flows but <u>without any flow carrying, unlinked</u> paths with respect to the given solution (f,y).

By Corollary 2.2a, the set $K_L \subseteq K_F$ is nonempty. Each commodity $k \notin K_L \cup K_I$ has exactly one flow carrying, unlinked path (by Corollary 2.1); let the isolated node corresponding to this unlinked path constitute the $m(k)^{th}$ component of G_a. Then, submatrix R_L is obtained from R_F by deleting row k and column m(k) for all $k \notin K_L \cup K_I$. The columns of R_F are linearly independent if and only if the columns of R_L are. Consequently,

<u>Corollary 2.4</u>:

Any fractional solution (f,y) of [L1] is an extreme point of P_{L1} if and only if

(1) $y_{ij} = \max_k \max_{p(k) \in P_{ij}(k)} f_{p(k)}$ for all $(i,j) \in A$,

(2) each commodity has at most one flow carrying, unlinked path, and

(3) the columns of the submatrix R_L corresponding to the given solution (f,y) are linearly independent.

We will now specialize the above results to the uncapacitated Plant Location problem.

Special case: The Plant Location problem:

Recall that, in the Plant Location problem (PLP), we are given a set I of potential plant sites and a set J of customers, each with a demand of 1 unit. The objective is to select a subset of locations for the plants and to assign customers to these plants to minimize the sum of the fixed costs (for building the plants) and the transportation costs. We have already discussed (in Section 1.4(d)) how the PLP can be modeled as a Network Design problem by adding a super-source, node 0, and an arc (0,i), for all $i \epsilon I$, with a fixed charge F_{0i} equal to the cost of building the plant at site i. The demand by each customer $k \epsilon J$ is treated as a separate commodity, designated as commodity k, with node 0 as the origin and node k as the destination. Each commodity $k \epsilon J$ has exactly $|I|$ paths, 0-i-k, for all $i \epsilon I$; we let i(k) denote the path 0-i-k.

Because of the special structure of the Plant Location network, the Auxiliary graph G_a and the matrix R corresponding to any given LP solution of the PLP have certain distinctive features. We now list some of these special characteristics.

(a) For any commodity $k \epsilon J$, the $|I|$ paths i(k), $i = 1,2,\ldots,|I|$, are arc-disjoint. Hence, no two vertices of G_a corresponding to commodity k can be adjacent to each other. Therefore, all the elements r_{km} of the matrix R are either 0 or 1.

In the PLP, only arcs of the form (0,i), that are incident <u>from</u> the source node 0, have fixed charges. We will adopt the convention that arcs with zero fixed-charge (such as (i,k), $i \epsilon I$, $k \epsilon J$ in the PLP) cannot be tight

for any path; in the problem formulation, we do not include design variables corresponding to these arcs. (Note that, with this definition of tightness, we must modify Lemma 1 to read: "Every arc with a positive fixed-charge must be tight for at least one path that contains it in any extreme point solution (f,y)". Similar modifications must be made in Corollaries 2.1 and 2.2). Then,

(b) if any two nodes [i'(k')] and [i"(k")] are adjacent in the Auxiliary graph G_a, then i' must be equal to i", i.e., the corresponding paths i'(k') and i"(k") must necessarily be of the form 0-i-k' and 0-i-k" respectively, for some $i \in I$ and $k', k" \in J$, $k' \neq k"$. Consequently, every component S_m is a clique of G_a; furthermore, there is an unique arc, designated as arc $(0,i_m)$, incident from the source, that is contained in all the paths p(k) whose corresponding vertices [p(k)] belong to component S_m. We will refer to this unique arc $(0,i_m)$ as the arc ASSOCIATED with component S_m of G_a. Notice that, if an arc (0,i) is "associated" with $p > 1$ components of G_a, then at least (p-1) of these components are isolated nodes of G_a.

In order to relate our results to the characterization outlined by Cornuejols et al. [1977b], we introduce an additional definition.

Definition 2.4:

Given any LP solution (f,y) of [L1], a path p(k) is said to be LOOSE if none of its arcs are tight (with respect to (f,y)) for that path.

Notice that, if a path is loose with respect to a given solution (f,y), then it is necessarily unlinked. In the PLP context, a path i(k) is loose if and only if $0 < f_{i(k)} < y_{0i}$ in the given solution (f,y). Corollaries

2.1, 2.2, and 2.4 can be easily reexpressed in terms of loose rather than unlinked paths. For Corollary 2.4, however, we must define a submatrix R_T of R in place of the submatrix R_L defined earlier. We will now show that the modified form of Corollary 2.4 is identical to Cornuejols et al's characterization.

The submatrix R_T of R (which is the counterpart of the submatrix R_L in Corollary 2.4) is defined as follows. The rows of R_T correspond to commodities k with fractional flows (in the given PLP solution (f,y)) that do not have any flow carrying, loose paths. In other words, the index set of rows of R_T is

$$K_T = \{k\epsilon J : f_{i(k)} = 0 \text{ or } y_{Oi} \text{ for all } i\epsilon I,\ 0<y_{Oi}<1\}.$$

The columns of R_T correspond to components S_m of the Auxiliary graph G_a that contain at least one node $[i(k)]$ for some $k\epsilon K_T$. It follows from our earlier discussion that these components can be identified by their unique "associated" arcs $(0,i_m)$, where i_m belongs to the subset $I_T = \{i\epsilon I : 0<y_{Oi}<1\}$. Thus, the elements r_{ki} of the submatrix R_T for the PLP are

$$r_{ki} = \begin{cases} 1 & \text{if } f_{i(k)} > 0 \\ 0 & \text{if } f_{i(k)} = 0 \end{cases} \qquad \text{for all } k\epsilon K_T \text{ and } i\epsilon I_T$$

Consequently, from Proposition 2, we have

Corollary 2.5:

A fractional solution (f,y) of the LP relaxation of the uncapacitated Plant Location problem is an extreme point of the LP polytope if and only if

(1) $y_{Oi} = \underset{k\epsilon J}{\text{Max}}\ f_{i(k)}$ for all $i\epsilon I$,

(2) for each $k \epsilon J$, there is at most one node $i \epsilon I$ with $0 < f_{i(k)} < y_{Oi}$, and

(3) the rank of the submatrix R_T is $|I_T|$.

This result is identical to the characterization of fractional extreme points for the LP relaxation of the Uncapacitated Plant Location problem given by Cornuejols et al. [1977b]. Notice that, since the Arc-flow and Path-flow formulations are identical for the PLP, Corollary 2.5 applies to both the formulations.

Cornuejols et al. considered the subset of fractional extreme points for which R_T contains a s x s square submatrix C^{st}, with $s = |I_T|$, $s<t$ and s and t relatively prime; the rows (and columns) of C^{st} are 0-1 vectors in which t contiguous ones are successively moved one position to the right (down). For this special class of fractional extreme points, they constructed a cut that, when added to the LP relaxation excludes the corresponding extreme point. In the next section, we show how this inequality can be obtained using the characterization of of Proposition 2.

2.2 Constructing Violated, Valid Inequalities for the UNDP

At the beginning of this chapter, we motivated our effort to characterize fractional extreme points of P_{L1} by citing one of its uses, namely for generating valid cuts ('valid' in the sense that these cuts do not exclude any feasible integer solutions of [UNDPF]) that are violated by some of the fractional extreme points. By adding these cuts to the formulation, we can strengthen the LP relaxation and possibly improve the

lower bound. For the Plant Location problem, Cornuejols, Fisher and Nemhauser [1977b] generated a class of such inequalities using their characterization of fractional extreme points. Guignard [1980] generalized this class, showed that some of the cuts in this class are in fact facets of the plant location IP polytope and described a family of problems with large duality gaps that are bridged by these facets. In the same spirit, we now use Proposition 2 to construct violated, valid inequalities for the Path-flow formulation of the UNDP. We describe two different types of inequalities, one based on the topology of the Auxiliary graph G_a and the other based on the dual solution of a related linear program. In Chapter 3, we present several general families of valid inequalities for the UNDP that contain the cuts described in this section. Both the inequalities outlined in this section can be interpreted as constraint aggregation procedures as described by Martin and Schrage [1982].

We begin by making a few observations related to Proposition 2. Recall that, by construction, if the Auxiliary graph G_a corresponding to any given solution (f,y) of [L1] contains an edge of the form ([p(k)],[p'(k')]), then

(a) at least one arc (i,j) of the original graph is tight for both the paths p(k) and p'(k'). We will refer to any such arc (i,j) of the original graph as an arc ASSOCIATED WITH edge ([p(k)],[p'(k')]) of G_a, and

(b) $f_{p(k)} = f_{p'(k')} = y_{ij}$ for all arcs (i,j) associated with the edge ([p(k)],[p'(k')]).

Therefore, if there is a path in G_a from vertex [p(k)] to another vertex [p'(k')], then

$$f_{p(k)} = f_{p'(k')},$$

implying that for all vertices [p(k)] that belong to the same component S_m, for $1 \leq m \leq M$, of G_a,

$$f_{p(k)} = \text{a constant, say } v_m, > 0.$$

We will refer to v_m as the FLOW ON COMPONENT S_m. (Note: Earlier, we discussed a similar condition that the flow perturbations $g_{p(k)}$ must satisfy.) Also note that for all arcs (i,j) of the original graph that are associated with at least one of the edges of component S_m, y_{ij} must be equal to v_m. We will use these facts to prove that the cuts that we discuss next are violated by the current fractional extreme point solution (f,y).

2.2.1 Method 1:

Given a fractional extreme point (f,y) of P_{L1}, our objective is to identify cuts that exclude the current solution when added to the LP relaxation [L1]. For the first class of such inequalities, we will assume that at least one element r_{km} of the matrix R corresponding to solution (f,y) is greater than or equal to 2. We use the Auxiliary graph G_a to identify the variables that must be included in the inequality and prove, using Proposition 2, that this cut is violated by the current solution.

Suppose $r_{km} \geq 2$ for some commodity $k \epsilon K$ and some component index m, for $1 \leq m \leq M$. Then, for at least 2 paths, say p'(k) and p"(k), for commodity k, the corresponding vertices [p'(k)] and [p"(k)] belong to the same component S_m of G_a. Before discussing the general form of the cut that can be constructed in this situation, we will deal with some special cases.

Case 1:

Suppose component S_m contains the edge $([p'(k)],[p''(k)])$, i.e., the vertices $[p'(k)]$ and $[p''(k)]$ are adjacent to each other in G_a. Let (i,j) be any arc of the original graph that is associated with this edge, i.e., arc (i,j) is tight for both the paths $p'(k)$ and $p''(k)$. Then, we claim that the inequality

$$f_{p'(k)} + f_{p''(k)} + \bar{y}_{ij} \leq 1, \qquad (3.1)$$

$$\text{where } \bar{y}_{ij} = 1 - y_{ij},$$

is valid for [UNDPF] and eliminates the current solution (f,y). This assertion can be proved as follows: Since arc (i,j) is contained in both $p'(k)$ and $p''(k)$, if $\bar{y}_{ij} = 1$ (and hence $y_{ij} = 0$) in any solution of [UNDPF], then both $f_{p'(k)}$ and $f_{p''(k)}$ must be zero (in order to satisfy the forcing constraints (2.3) of [UNDPF]), implying that (3.1) is satisfied. On the other hand, if $\bar{y}_{ij} = 0$ (i.e., $y_{ij} = 1$), the demand constraints (2.2) of [UNDPF] ensure that:

$$f_{p'(k)} + f_{p''(k)} \leq 1;$$

again, this solution satisfies (3.1). Hence, the inequality (3.1) does not exclude any feasible integer solution of [UNDPF]. We now show that inequality (3.1) is also violated by the current fractional solution (f,y). In the current solution, since vertices $[p'(k)]$ and $[p''(k)]$ belong to the same component S_m of G_a, and since (i,j) is the arc associated with the edge $([p'(k)],[p''(k)])$, we must have

$$f_{p'(k)} = f_{p''(k)} = y_{ij} = v_m,$$

where v_m is the flow on component S_m. Therefore,

$$f_{p'(k)} + f_{p''(k)} + \bar{y}_{ij} = 1 + v_m > 1,$$

since $v_m > 0$, implying that the current solution (f,y) violates (3.1).

Thus, we have shown that if two vertices corresponding to some commodity k are adjacent in the Auxiliary graph G_a, then an inequality of the form (3.1) is a violated, valid cut. We will now extend the above idea to the situation in which two vertices, say [p'(k)] and [p"(k)], corresponding to commodity k are not necessarily adjacent, but are connected in G_a. Before examining the general case, we will consider the case when the path from [p'(k)] to [p"(k)] in G_a passes through a single intermediate vertex.

Case 2:

Suppose the path from vertex [p'(k)] to vertex [p"(k)] in S_m passes through a single intermediate vertex, say $[p(k_1)]$. Without loss of generality, we can assume that $k_1 \neq k$ and that [p'(k)] and [p"(k)] are not adjacent to each other in G_a; otherwise, a violated, valid inequality of the form (3.1) can be formulated. Let (i_0,j_0) and (i_1,j_1) be any two arcs of G that are associated with the edges $([p'(k)],[p(k_1)])$ and $([p(k_1)],[p"(k)])$ of G_a respectively. Then, the inequality

$$f_{p'(k)} + f_{p"(k)} + f_{p(k1)} + \bar{y}_{i0,j0} + \bar{y}_{i1,j1} \leq 2 \qquad (3.2)$$

is valid for [UNDPF] and cuts off solution (f,y). (Note: For convenience, we will denote the flow on a path $p(k_t)$ as $f_{p(kt)}$ and the design variable for arc (i_t,j_t) as $y_{it,jt}$.) The validity of (3.2) for [UNDPF] is easily verified. To check that (f,y) violates (3.2), observe that

$$f_{p'(k)} = f_{p"(k)} = f_{p(k1)} = y_{i0,j0} = y_{i1,j1} = v_m > 0$$

in the current solution (f,y). Consequently, the left-hand side of inequality (3.2) evaluated at the current solution is $(2 + v_m)$, which is greater than 2.

We now consider the general form of the inequality in this class. Recall that we have assumed that some element r_{km} of the matrix R is greater than or equal to 2, and we let [p'(k)] and [p"(k)] denote two of the r_{km} vertices corresponding to commodity k in component S_m of G_a. Given any two vertices that are connected in G_a, we will refer to the path in G_a between these two nodes with the minimum number of arcs as the MINIMUM HOP path. Suppose a minimum hop path from [p'(k)] to [p"(k)] in S_m passes through T intermediate vertices $[p(k_1)],\ldots,[p(k_T)]$, corresponding to commodities $k_t \neq k$ for $t = 1,2,\ldots,T$. Let (i_0,j_0), (i_t,j_t), $1\leq t<T$, and (i_T,j_T) be any arcs of the original graph that are ASSOCIATED WITH the edges $([p'(k)],[p(k_1)])$, $([p(k_t)],[p(k_{t+1})])$, for $1\leq t<T$, and $([p(k_T)],[p"(k)])$ of G_a respectively.

Claim:

The (T+1) associated arcs (i_t,j_t), for $0\leq t\leq T$, of the original graph must necessarily be distinct.

Proof:

Suppose not, i.e., suppose for some $0\leq t'<t"\leq T$, the associated arcs $(i_{t'},j_{t'})$ and $(i_{t"},j_{t"})$ are identical. Then, this arc is tight for both the paths $p(k_{t'})$ and $p(k_{t"})$; hence, the corresponding vertices must be adjacent in G_a (while they are not adjacent in the path from [p'(k)] to [p"(k)] that we first considered). Therefore, there must be a path in G_a from [p'(k)] to [p"(k)] with a fewer number of edges than the minimum hop path that we considered above - a contradiction.

For the moment, we will assume that all the commodities k_t, for

t=1,2,...,T, are distinct; later, we will justify this assumption.

PROPOSITION 3:

For any given fractional extreme point (f,y) of P_{L1}, let the path from node [p'(k)] to [p"(k)] in the corresponding Auxiliary graph G_a be of the form discussed above. Then, the inequality

$$f_{p'(k)} + f_{p"(k)} + \sum_{t=1}^{T} f_{p(kt)} + \sum_{t=0}^{T} \bar{y}_{it,jt} \leq T+1, \qquad (3.3)$$

is valid for [UNDPF] and is violated by the current solution (f,y).

Proof:

The validity of (3.3) for [UNDPF] stems from the following facts:

(a) The demand constraints require that

$$f_{p'(k)} + f_{p"(k)} \leq 1, \text{ and}$$

$$f_{p(kt)} \leq 1 \quad \text{for all } 1 \leq t \leq T, \text{ and}$$

(b) the forcing constraints (2.3) of [UNDPF] ensure that

if $\bar{y}_{i0,j0} = 1$, then $f_{p'(k)} = f_{p(k1)} = 0$

if $\bar{y}_{it,jt} = 1$, then $f_{p(kt)} = f_{p(kt+1)} = 0$ for all $1 \leq t \leq T-1$, and

if $\bar{y}_{iT,jT} = 1$, then $f_{p(kT)} = f_{p"(k)} = 0$.

Therefore, if in any integer solution of [UNDPF], $T' \leq T+1$ of the $\bar{y}$ variables that are included in the inequality (3.3) are equal to 1, then the sum of the flow variables in the left-hand side of (3.3) must necessarily be less than or equal to (T+1-T'). Alternatively, we can show that (3.3) is valid by demonstrating that it can be obtained as a weighted sum of selected constraints in [UNDPF], with the right-hand side constant of the resulting aggregated constraint rounded down. The operation of rounding down the right-hand side constant is legitimate

since [UNDPF] has an optimal solution in which all the f-variables are integral. We elaborate on this alternative interpretation in Chapter 3.

To verify that (3.3) cuts off the current solution (f,y), we note that in the current solution

$$v_m = f_{p'(k)} = f_{p''(k)} = f_{p(kt)}, \text{ for } 1 \leq t \leq T, \text{ and}$$

$$v_m = y_{it,jt}, \text{ for } 0 \leq t \leq T,$$

where v_m is the positive flow on component S_m. Substituting these values in the left-hand side of (3.3), we obtain

$$v_m + v_m + \sum_{t=1}^{T} v_m + \sum_{t=0}^{T} (1-v_m) = (T+1) + v_m > (T+1).$$

Hence, the current solution violates inequality (3.3).

We will now address the assumption that all "intermediate" commodities k_t, for $1 \leq t \leq T$, are distinct. Observe that we did not use this assumption at any step of the preceding proof. Hence, the inequality (3.3) is valid and violated by the sclution (f,y) even if the distinctness assumption does not apply. However, we will show now that if the intermediate commodities are not distinct, then there is a valid, violated inequality similar to (3.3), but with a smaller right-hand side constant, that is at least as tight as (3.3). Suppose the t'^{th} and t''^{th} vertices on the minimum hop path from p'(k) to p"(k), for some $1 \leq t' < t'' \leq T$, correspond to the same commodity, say k' (i.e., $k_{t'} = k_{t''} = k'$). We will denote these two vertices as [p'(k')] and [p"(k')], respectively. Without loss of generality, assume that the subpath from [p'(k')] to [p"(k')] in G_a satisfies the distinctness assumption, i.e., k_t, for $t' < t < t''$ are distinct commodities. This subpath from [p'(k')] to [p"(k')] in G_a must necessarily be a minimum hop path; hence, an inequality similar to (3.3), formulated with respect to this

subpath (rather than for the original path from [p'(k)] to [p"(k)]), is a violated, valid inequality. We will refer to the inequalities (3.3) corresponding to the path from [p'(k)] to [p"(k)] and the subpath from [p'(k')] to [p"(k')] as (3.3k) and (3.3k'), respectively. Also, for convenience, we will assume that t'=1 and t"=T, i.e., $k_1 = k_T = k'$, with k_t, for 1<t<T, being distinct commodities. Then, the inequality (3.3k) is a linear combination of (3.3k') and the following forcing constraints of [UNDPF]:

$$f_{p'(k)} \leq y_{i0,j0}, \text{ and}$$

$$f_{p"(k)} \leq y_{iT,jT}.$$

Thus, the inequality (3.3k') is at least as strong as (3.3k) for the LP relaxation of [UNDPF]. This argument can be readily extended to the more general case in which $1 \leq t' < t" \leq T$. In summary, we have just shown that if the cuts (3.3) are formulated for all paths of G_a that satisfy the distinctness assumption for intermediate commodities, then the inequalities such as (3.3k) that correspond to other paths in G_a are redundant even in the LP relaxation of [UNDPF].

Similarly, we can show that

(1) it suffices to consider inequalities (3.3) corresponding to minimum hop paths from [p'(k)] to [p"(k)]. While the inequality formulated with respect to any other path with a larger number of arcs is valid for [UNDPF], it is redundant in the LP relaxation [L1] when the corresponding "minimum hop path" inequality is also added, and

(2) if the minimum hop path from [p'(k)] to [p"(k)] in S_m passes through a third node [p*(k)] corresponding to commodity k (in which case r_{km} must be ≥ 3), then it suffices to consider cuts (3.3) that correspond

to the minimum hop subpaths in G_a from $[p'(k)]$ to $[p^*(k)]$ and from $[p^*(k)]$ to $[p''(k)]$.

We now summarize the foregoing cut-generation procedure for the case when at least one element of the matrix R corresponding to the given fractional extreme point (f,y) is greater than or equal to 2.

Procedure:

- For the given fractional extreme point (f,y) of P_{L1}, construct the corresponding Auxiliary graph G_a and the matrix R.
- For each component S_m, for $1 \leq m \leq M$, of G_a, let

$$K_m = \{k \epsilon K : r_{km} \geq 2\}.$$

For each commodity $k \epsilon K_m$, and for every pair of vertices $[p'(k)]$, $[p''(k)]$ belonging to component S_m

(1) Find the minimum hop path from $[p'(k)]$ to $[p''(k)]$ in S_m.

(2) If all the intermediate vertices in this path correspond to distinct commodities $k_t \neq k$, then add the inequality (3.3) corresponding to this path to the LP relaxation [L1] of [UNDPF].

This procedure generates all the violated, valid inequalities of the type discussed in Proposition 3. However, if we want to identify only the most violated inequality in this class, we can use the following method:

- Let v_m, for $1 \leq m \leq M$, be the flow on component S_m in the current solution (f,y).
- Find the component $S_{m'}$ for which

$$v_{m'} = \text{Max } \{v_m : r_{km} \geq 2 \text{ for some } k \epsilon K\}.$$

For any commodity $k \epsilon K$ with $r_{km'} \geq 2$, find two vertices, say $[p'(k)]$ and $[p''(k)]$, corresponding to commodity k in $S_{m'}$, and the minimum hop path

between these two vertices that satisfies the conditions of Proposition 3. Then, the inequality (3.3) corresponding to this path is a "most violated" inequality.

The choice of the component $S_{m'}$ in this procedure ensures that the difference between the left-hand side of (3.3) evaluated at the current solution (f,y) (= $T+1+v_{m'}$) and the right-hand side of (3.3) (= T+1) is the maximum possible among all such inequalities.

The procedures that we have discussed so far address the so-called CONSTRAINT IDENTIFICATION PROBLEM, i.e., the problem of determining valid inequalities that eliminate a <u>given fractional extreme point</u> (f,y). However, we can generate, a priori, all possible inequalities of the type (3.3) just by examining the structure of the original graph G (over which the problem is defined) and without recourse to devices such as the Auxiliary graph G_a. This construction can be done in the following manner:

- Construct the graph G':(N',E') containing:
  - one vertex [p(k)] for each path $p(k) \epsilon P(k)$, for all $k \epsilon K$, in G,
  - edge ([p(k)],[p'(k')]) if the corresponding paths p(k) and p'(k') have at least one arc (i,j) in common.
- Apply the cut-generation procedure discussed earlier using the graph G' instead of the Auxiliary graph G_a.

This procedure will yield <u>all</u> the valid inequalities of the form (3.3). However, the number of such inequalities is likely to be very large, and many of them may not be binding at the extreme points that are of interest to us. The advantage of using the Auxiliary graph G_a (instead of G') is that it helps to identify all the 'relevant' inequalities iteratively,

resulting in a "strengthened" relaxation of [UNDPF] with fewer constraints.

So far, we have restricted our attention to the case in which at least one element r_{km} of the matrix R corresponding to the given solution (f,y) is greater than or equal to 2. It is possible to extend this procedure for generating violated, valid inequalities to situations in which all the elements of the matrix R are either 0 or 1. This extension involves adding some selected arcs to the Auxiliary graph G_a, and identifying, in the new graph G^*, paths between pairs of nodes that correspond to the same commodity (just as in the previous case). However, the paths that are traced in G^* must satisfy some additional conditions for the resulting cut (which is similar in form to (3.3)) to exclude the current solution (f,y). The details of this procedure are rather involved; we describe the procedure in Appendix A.

In Chapter 3, we transform the Path-flow formulation of the UNDP into a Set packing problem; we then demonstrate that the inequalities discussed so far are, in fact, a subset of a certain family of facets for the Set packing polytope. We also show that the same cuts can be obtained using a constraint aggregation procedure, thus integrating the three seemingly diverse cut-generation techniques.

2.2.2 Method 2:

We now turn to a second family of valid inequalities that are violated by the given fractional extreme point solution (f,y). The method that we use to generate these inequalities is significantly different from the

approach just outlined. We first determine the optimal dual solution for a system of equations that is based on the structure of the Auxiliary graph G_a. Using the dual values as multipliers for certain constraints of [UNDPF], we construct an aggregate constraint whose coefficients can be reduced. This procedure results in a valid inequality for [UNDPF] that is violated by the current solution. The method is applicable only under certain assumptions about the dual solution and the structure of the Auxiliary graph G_a.

Recall from our earlier discussions that, for any given extreme point solution (f,y), the flows $f_{p(k)}$ on all paths p(k) whose corresponding vertices [p(k)] belong to a particular component S_m of G_a must be the same. We referred to this common value v_m as the FLOW ON COMPONENT S_m. Now, the element r_{km} of matrix R represents the number of nodes in G_a corresponding to commodity k that are contained in component S_m, for all $k \epsilon K$ and $1 \leq m \leq M$. Also, G_a contains vertices [p(k)] for all flow carrying paths p(k). Hence, the original demand equations

$$\sum_{p(k)} f_{p(k)} = 1 \qquad \text{for all } k \epsilon K$$

can be alternatively expressed in terms of r_{km} and v_m as

$$\sum_{m} r_{km} v_m = 1 \qquad \text{for all } k \epsilon K.$$

Thus, the vector of component flows $V = [v_1, \ldots, v_M]$ is any solution to the system of equations

$$RV = 1 \tag{3.4}$$

where 1 is a vector of $|K|$ ones and R is the matrix corresponding to the given fractional vertex (f,y). Furthermore, since (f,y) is an extreme point of P_{L1}, by Proposition 2 we know that the only solution to the

homogeneous system of equations

$$R\theta = 0, \quad \theta \epsilon R^M$$

is $\theta = 0$. This result implies that (3.4) has a unique solution V^* (which in turn uniquely determines the corresponding extreme point solution (f,y)). Since $f = \{f_{p(k)}\}$ is a feasible flow for [L1], v_m^* must be less than or equal to 1, for all $1 \leq m \leq M$; moreover, since (f,y) is fractional, at least one component of V^* must be strictly less than 1. (Notice that (f,y) fractional and extreme implies that f is fractional; otherwise, condition (a) of Proposition 2 is violated). Therefore,

$$\sum_{m=1}^{M} v_m^* < M \ .$$

Now, consider the system of equations

$$\begin{aligned} RV &= 1 \\ \sum_{m=1}^{M} v_m &= M \end{aligned} \tag{3.5}$$

where M is the total number of components in the Auxiliary graph G_a, and

R is the matrix corresponding to the given solution (f,y).

The argument given previously implies that (3.5) does not have any feasible solution. Hence, the corresponding dual

$$\begin{aligned} &\text{Minimize} \quad \sum_k u_k + Mz_0 \\ &\text{subject to} \\ &\sum_k r_{km} u_k + z_0 \geq 0 \qquad \text{for all } m = 1,2,\ldots,M \end{aligned} \tag{3.6}$$

is either infeasible or unbounded. However, the solution $u_k = 0$, for all $k \epsilon K$, and $z_0 = 0$ is a feasible solution; therefore, (3.6) must be unbounded. This conclusion implies the existence of a vector (u,z_0) satisfying

$$\sum_k r_{km}u_k + z_0 = 0 \qquad \text{for all } m = 1,2,\ldots,M$$
$$\sum_k u_k + Mz_0 < 0 \quad . \tag{3.7}$$

We let (u^*,z_0^*) denote such a vector. Note that at least one component of u^* must be nonzero.

Our strategy is to use the values of u^* and z_0^* as multipliers for aggregating certain constraints in the formulation [UNDPF]. Before proceeding further, we make two assumptions that are required for our construction.

Assumption 1:

For each component S_m of G_a, there is at least one arc, say (i_m,j_m) that is "associated" with all the edges $([p(k)],[p'(k')])$ of S_m.

Recall, that arc (i,j) of the original graph is said to be "associated with" edge $([p(k)],[p'(k')])$ of G_a if arc (i,j) is tight (with respect to solution (f,y)) for both the paths p(k) and p'(k'). This assumption is thus equivalent to the statement that

"for all vertices [p(k)] belonging to component S_m, for $1 \le m \le M$, of G_a, arc (i_m,j_m) of G belongs to path p(k) and $y_{i_m,j_m} = f_{p(k)}$".

(Again, y_{i_m,j_m} denotes the design variable corresponding to arc (i_m,j_m)). As mentioned earlier, the Auxiliary graph G_a corresponding to the uncapacitated Plant Location problem always satisfies this assumption.

Assumption 2:

$u_k^* \ge 0$, for all $k \in K$, where (u^*,z_0^*) is a solution to the system of

equations (3.7).

Under this assumption, z_0^* must be negative in order to satisfy the last inequality in (3.7). Also, since $\Sigma_m r_{km} \geq 1$, for all $k \epsilon K$ (i.e., since at least one node corresponding to each commodity k is contained in G_a), we must have

$$- z_0^* - u_k^* \geq 0 \quad \text{for all } k \epsilon K.$$

For ease of exposition, we next define two subsets of paths. Let,

$Q_m = \{p(k) : k \epsilon K,\ p(k) \epsilon P(k)$, and $[p(k)]$ is a vertex of G_a belonging to component $\omega_m\}$, for all $m = 1,2,\ldots,M$, and

$Q_m(k) = \{p(k) : p(k) \in Q_m\}$ for all $m = 1,2,\ldots,M$, and $k \epsilon K$.

Consider the following forcing and demand constraints of [UNDPF].

(a) The forcing constraints

$$f_{p(k)} \leq y_{i_m, j_m} \quad \text{for all } m = 1,\ldots,M, \text{ and } p(k) \epsilon Q_m.$$

Note that, by Assumption 1, arc (i_m, j_m) belongs to path $p(k)$ for all $p(k) \epsilon Q_m$; hence, these constraints are included in [UNDPF].

(b) The demand restrictions

$$\sum_{m=1}^{M} \sum_{p(k) \epsilon Q_m(k)} f_{p(k)} \leq 1 \quad \text{for all } k \epsilon K.$$

Since $Q_m(k) \subseteq P(k)$, these constraints are relaxed versions of the demand equations in [UNDPF].

Since $u_k^* \geq 0$, for all $k \epsilon K$ (by Assumption 2), we can multiply each of the inequalities (a) corresponding to commodity k by u_k^* to obtain

$$u_k^* f_{p(k)} \leq u_k^* y_{i_m, j_m} \quad \text{for all } m=1,\ldots,M, \text{ and } p(k) \epsilon Q_m.$$

Summing these constraints over all indices $m = 1,2,\ldots,M$ and $p(k) \epsilon Q_m$, we obtain

$$\sum_m \sum_{p(k)\epsilon Q_m} u_k^* f_{p(k)} + \sum_m \sum_{p(k)\epsilon Q_m} u_k^* \bar{y}_{im,jm} \leq \sum_m \sum_{p(k)\epsilon Q_m} u_k^* ,$$

where $\bar{y}_{ij} = 1 - y_{ij}$. Now,

$$\sum_m \sum_{p(k)\epsilon Q_m} u_k^* = \sum_m \left(\sum_k u_k^* \left[\sum_{p(k)\epsilon Q_m(k)} 1 \right] \right)$$

$$= \sum_m \sum_k r_{km} u_k^*$$

since $r_{km} = |Q_m(k)|$, the number of nodes corresponding to commodity k in component S_m.

But, $\sum_k r_{km} u_k^* = -z_0^*$ for all $1 \leq m \leq M$,

since (u^*, z_0^*) is a solution of (3.7). Therefore,

$$\sum_m \sum_{p(k)\epsilon Q_m} u_k^* = \sum_m (-z_0^*) = -Mz_0^*.$$

Similarly,

$$\sum_m \sum_{p(k)\epsilon Q_m} u_k^* \bar{y}_{im,jm} = \sum_m (-z_0^*) \bar{y}_{im,jm} .$$

Substituting these values in the last inequality gives

$$\sum_m \sum_{p(k)\epsilon Q_m} u_k^* f_{p(k)} + \sum_m (-z_0^*) \bar{y}_{im,jm} \leq -Mz_0^* \qquad (3.8)$$

Now, multiply each of the demand inequalities (2) by $(-z_0^* - u_k^*)$ (which is non-negative, by Assumption 2) to get

$$\sum_m \sum_{p(k)\epsilon Q_m(k)} (-z_0^* - u_k^*) f_{p(k)} \leq -z_0^* - u_k^* \quad \text{for all } k\epsilon K .$$

Summing these for all $k\epsilon K$ and adding the resulting inequality to (3.8) gives

$$\sum_m \sum_{p(k)\epsilon Q_m} (-z_0^*) f_{p(k)} + \sum_m (-z_0^*) \bar{y}_{im,jm} \leq -Mz_0^* - |K| z_0^* - \sum_k u_k^*$$

Since $z_0^* < 0$ (from Assumption 2), we can divide throughout by $-z_0^*$; the resulting inequality is

$$\sum_m \sum_{p(k)\epsilon Q_m} f_{p(k)} + \sum_m \bar{y}_{im,jm} \leq |K| + (Mz_0^* + \sum_k u_k^*)/z_0^* \qquad (3.9)$$

So far, we have chosen appropriate multipliers for certain valid

inequalities for [UNDPF] and aggregated the weighted inequalities to obtain a single constraint that is valid for [UNDPF]. This process is referred to as CONSTRAINT AGGREGATION. Our purpose now is to exploit the integrality of an optimal solution for [UNDPF] in order to "reduce" or round down the coefficients of the aggregate constraint such as (3.9). This last operation will constitute the COEFFICIENT REDUCTION phase of the cut-generation procedure.

We know that [UNDPF] has an optimal solution in which all the $f_{p(k)}$ and y_{ij} values (and hence the $\bar{y}_{ij}$ values) are either 0 or 1. Since the coefficients of the f and $\bar{y}$ variables in the left-hand side of inequality (3.9) are all integers, we can round down the right-hand side constant of (3.9) without excluding any integer solutions of [UNDPF]. Thus, the inequality

$$\sum_m \sum_{p(k)\in Q_m} f_{p(k)} + \sum_m y_{im,jm} \leq \lfloor K \rfloor + \lfloor (Mz_0^* + \sum_k u_k^*)/z_0^* \rfloor \qquad (3.10)$$

where $\lfloor a \rfloor$ is the largest integer less than or equal to a, is valid for [UNDPF].

In order to show that the valid inequality (3.10) cuts off the current fractional extreme point (f,y), we must make the following additional assumption about the dual solution (u^*, z_0^*).

Assumption 3:

The solution (u^*, z_0^*) of (3.7) satisfies

$$\sum_k u_k^* + (M-1)z_0^* > 0.$$

Notice that under this assumption,

$$\sum_k u_k^* + Mz_0^* > z_0^*$$

$$\Longrightarrow \quad (\sum_k u_k^* + Mz_0^*)/z_0^* < 1 \qquad \text{[since } z_0^* < 0 \text{ by Assumption 2]}$$

$$\Longrightarrow \quad |(\sum_k u_k^* + Mz_0^*)/z_0^*| \leq 0.$$

But, from (3.7),

$$\sum_k u_k^* + Mz_0^* < 0$$

implying that

$$|(\sum_k u_k^* + Mz_0^*)/z_0^*| = 0.$$

Substituting this expression in inequality (3.10), we get the cut

$$\sum_m \sum_{p(k)\in Q_m} f_{p(k)} + \sum_m \bar{y}_{im,jm} \leq |K|. \qquad (3.11)$$

PROPOSITION 4:

Under assumptions 1, 2 and 3, inequality (3.11) is valid for [UNDPF] and is violated by the current fractional extreme point solution (f,y) of [L1].

Proof:

We have already proved the validity of (3.11) for [UNDPF]. In order to show that (3.11) cuts off the current solution (f,y), we observe that

$$f_{p(k)} = v_m^* \quad \text{for all } 1 \leq m \leq M \text{ and } p(k) \in Q_m, \text{ and}$$

$$y_{im,jm} = v_m^* \quad \text{for all } 1 \leq m \leq M,$$

where v_m^* is the flow on component S_m of G_a for the solution (f,y). Substituting these values in the left-hand side of (3.11) gives

$$\sum_m \sum_{p(k)\epsilon Q_m} v_m^* + \sum_m (1-v_m^*)$$

$$= \sum_m \Big(\sum_k r_{km}\Big) v_m^* + \sum_m (1-v_m^*)$$

$$= \sum_k \sum_m r_{km} v_m^* + \sum_m (1-v_m^*)$$

$$= \sum_k 1 + \sum_m (1-v_m^*) \qquad \Big[\text{since } \sum_m r_{km} v_m^* = 1 \text{ for all } k\epsilon K\Big]$$

$$= |K| + M - \sum_m v_m^*$$

$$> |K|,$$

since $\sum_m v_m^* < M$ for the <u>fractional</u> solution (f,y).

Thus, the current solution (f,y) violates inequality (3.11).

We will now show that, for the special case of the uncapacitated Plant Location problem, inequality (3.10) reduces to the cut that was originally proposed by Cornuejols et al. [1977b] and later generalized by Guignard and Cornuejols and Thizy [1982].

Consider the Plant Location problem in which I is the set of potential plant sites, J is the set of customer locations and commodity k flows from the source node 0 to customer node $k\epsilon J$. Let i(k) denote the path 0-i-k, $i\epsilon I$, $k\epsilon J$ for commodity k; it carries a flow of $f_{i(k)} = x^k_{0i}$ units. As mentioned earlier, the Auxiliary graph G_a for this problem satisfies Assumption 1, i.e., some arc $(0,i_m)$ of the original graph is associated with all the edges in any given component S_m of G_a. (In fact, every component S_m is a clique of G_a containing only vertices of the form $[i_m(k)]$, $k\epsilon J$). Furthermore, all the elements r_{km} of the of the corresponding matrix R are either 0 or 1 for any LP extreme point of the

PLP.

Consider any fractional extreme point (f,y) of the PLP polytope and the corresponding Auxiliary graph G_a. Let $S \leq M$ be the number of components S_m of G_a that are not isolated nodes and whose corresponding flows v_m^* are fractional. For convenience, we will assume that these "fractional" components are indexed from 1 to S. The fact that these components are not isolated nodes implies that all the associated arcs $(0,i_m)$, for $1 \leq m \leq S$ of the original graph are distinct. Since (f,y) is an extreme point of the LP relaxation of PLP, by Proposition 2, the corresponding matrix R must have linearly independent columns. Consequently, there must be a p x p submatrix, say B, of R which is nonsingular. The fractional part of the component flow vector $V^* = [v_1^*,\ldots,v_m^*]$ corresponding to solution (f,y) is then the unique solution to the system of equations

$$BV^f = 1 \in R^p.$$

(Note that these observations are valid for any Uncapacitated Network Design problem).

Suppose B is a cyclic matrix, i.e., the rows of B are 0-1 vectors containing t $(1 < t < p)$ consecutive ones that are successively moved one position to the right. Then, B is nonsingular if and only if t and p are relatively prime. An example of a cyclic matrix, with p=4 and t=3 is shown below:

$$B = \begin{vmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{vmatrix}.$$

Let $K_B \subseteq J$, $|K_B| = p$ be the set of commodities whose corresponding rows are included in the submatrix B. Note that all commodities belonging to the set K_B must have fractional flows in the solution (f,y). Consider the following solution (u^*, z_0^*) to the system of equations (3.7):

$$u_k^* = \begin{cases} 1 & \text{if } k \in K_B \\ 0 & \text{otherwise} \end{cases}$$

$$z_0^* = -t \ .$$

Since $\sum_{k \in KB} r_{km} = -t$, for all $1 \leq m \leq M$, this solution satisfies the equations

$$\sum_k r_{km} u_k^* + z_0^* = 0 \quad \text{for all } 1 \leq m \leq M.$$

Also,
$$\sum_k u_k^* + M z_0^* = \sum_{k \in KB} u_k^* - Mt$$
$$= p - Mt \leq M(1-t) < 0 \qquad [\text{since } t>1].$$

Thus, (u^*, z_0^*) solves the system (3.7). Moreover, this solution satisfies Assumption 2 since $u_k^* \geq 0$, for all $k \in K$. Therefore, the aggregate constraint (3.10) constructed using the dual solution (u^*, z_0^*) is a valid inequality for the PLP. In the current context, this constraint can be expressed in the following form:

$$\sum_k \sum_m r_{km} f_{i_m(k)} + \sum_m \bar{y}_{0,i_m} \leq |K| + |(-Mt + p)/-t|$$
$$= |K| + M - |p/t|, \qquad (3.12)$$

where $(0,i_m)$ is the unique arc of the original graph that is associated with component S_m of G_a, and

$|a|$ is the smallest integer greater than or equal to a.

We claim that inequality (3.12) implies the following cut proposed by Cornuejols et al. [1977b]:

$$\sum_{k \in KB} \sum_{m=1}^{p} b_{km} f_{i_m(k)} + \sum_{m=1}^{p} \bar{y}_{0,i_m} \leq 2p - |p/t|, \qquad (3.13)$$

where the coefficients b_{km} are the elements of the cyclic matrix B.

To establish this result, we use the following inequalities that are satisfied by any feasible solution of the PLP:

(1) $\sum_{m} r_{km} f_{im(k)} \leq 1$ for all $k \notin K_B$

(2) for all components S_m, with $p<m\leq M$, that are isolated nodes of G_a and that have $r_{km} = 1$ for some $k \epsilon K_B$,

$$f_{im(k)} \leq y_{0,im} ,$$

$$\text{i.e.,} \quad f_{im(k)} + \bar{y}_{0,im} \leq 1 .$$

(3) for all other components S_m with index $m>p$,

$$\bar{y}_{0,im} \leq 1 .$$

Substituting these in inequality (3.12) gives

$$\sum_{k \epsilon KB} \sum_{m=1}^{p} r_{km} f_{im(k)} + \sum_{m=1}^{p} \bar{y}_{0,im} \leq |K|+M-|p/t|-(|K|-|K_B|)-(M-p)$$

$$= |K_B| + p - |p/t| = 2p - |p/t|,$$

which is the same as (3.13) since $b_{km} = r_{km}$ for all $k \epsilon K_B$, and $1 \leq m \leq p$.

We have thus shown that the class of inequalities (3.10) that we constructed using a Constraint Aggregation/Coefficient Reduction procedure (based on appropriate multipliers derived from the characterization of Proposition 2) subsumes the inequalities related to cyclic matrices that Cornuejols et al [1977b] derived for the Plant Location problem. In the next chapter we show that similar inequalities based on certain cyclic matrices can be derived for the more general Uncapacitated Network Design problem using a Set packing interpretation of the Path-flow formulation. This second family of inequalities that we have discussed so far is applicable even when all the elements of the matrix R are either 0 or 1 (as was the case for the Plant Location problem); however, to formulate such inequalities we require a dual solution that satisfies the three assumptions.

2.3 Concluding Remarks

In this chapter, we have examined the structure of the LP polytope of the Path-flow formulation for the UNDP. We first characterized the extreme points of this polytope in terms of an Auxiliary graph G_a and a related matrix R. This result generalizes the characterization discussed by Cornuejols et al. [1977b] for the uncapacitated Plant Location problem. Next, we used the extreme point characterization to develop two constraint identification procedures for generating valid inequalities that are violated by a given fractional extreme point. We indicated alternative methods for generating and interpreting the two classes of inequalities outlined in this chapter; some of these other techniques are discussed in Chapter 3. Finally, we showed how the second class of inequalities, when specialized to the Plant Location problem, subsumes the cuts proposed by Cornuejols et al. [1977b].

The results of this chapter are of greater interest theortically than they are for algorithmic development for the following reasons:

(1) Throughout this chapter, we have focussed on the weak disaggregated Path-flow formulation of the UNDP rather than on either the Arc-flow formulation or the strong disaggregated Path-flow formulation (that contains the tighter forcing constraints (2.3a) in place of the constraints (2.3) in [UNDPF]). The main reason for this choice is the analytical tractability of the first model vis-a-vis the others and the fact that the results do not seem to extend in any straightforward

manner to either of the other two strong formulations. From an algorithmic point of view, the Path-flow formulation is inherently difficult to work with because of the large number of path-flow variables that would be required for problems of reasonable/realistic size. A column generation technique must, therefore, be used to solve the UNDP in this form.

(2) The characterization of Proposition 2 is applicable only when there are no additional side constraints, including the valid, violated inequalities discussed in this chapter, in the formulation [UNDP]. Consequently, the two constraint generation procedures can be used only when the current solution is an extreme point of the original polytope. Therefore, even if we have an algorithm for solving the Path-flow formulation, we cannot directly couple that algorithm with the constraint identification procedures to iteratively improve the lower bound. One alternative would be to use a Lagrangian-based scheme such as the following:

Step 0: Find the optimal extreme point (f,y) of the LP relaxation [L1] of [UNDPF] (without any additional side constraints).
Generate valid inequalities that exclude the current solution (f,y).

Step 1a: Dualize all the valid inequalities generated so far using some Lagrange multipliers.
Find the optimal extreme point solution (f,y) to the Lagrangian subproblem.

Step 1b: Modify, if possible, the Lagrange multipliers to increase the Lagrangian objective function value.

Step 2: If (f,y) is fractional, generate, if possible, new valid inequalities that exclude (f,y).
If new valid inequalities were generated in this step, return to Step 1.
Otherwise, STOP; either the procedure has found an optimal integer solution or it cannot increase the lower bound further.

The Lagrangian subproblem has the same constraints as [L1]; hence,

the constraint generation procedures discussed in this chapter are applicable, once the optimal Lagrangian solution is found. Notice that the inequalities generated in Step 2 are valid even if the optimal Lagrange multipliers are not determined in Step 1B. Step 1B can be implemented using a subgradient or other ascent procedure.

(3) The first constraint identification procedure always generates at least one violated, valid inequality whenever the matrix R has some element r_{km} that is greater than one. However, neither the generalization of this procedure (described in Appendix 2) nor the second approach is guaranteed to generate such a cut when all the elements of R are either 0 or 1. Therefore, any procedure (such as the one outlined above) that iteratively adds the two types of inequalities described in this chapter to improve the lower bound might terminate prematurely without finding the optimal integer solution to the UNDP.

(4) Ideally, we would like to generate and add the deepest possible cuts, namely facets of the IP polytope, at each stage. However, we have not been able to show that the inequalities discussed in this chapter are facets. The disadvantage of non-facial inequalities is that their addition, while excluding fractional extreme points of the original LP polytope, might create new fractional extreme points.

So far, we have commented on the usefulness of the results outlined in this chapter for algorithmic development. The results by themselves are of theoretical interest, however, especially since they generalize some of the earlier analyses of the Plant Location problem. Also, the discussions of this and the next chapter serve to integrate several alternative constraint

generation techniques. In the next chapter, by developing several classes of valid inequalities for the strong disaggregated Path-flow formulation as well as the Arc-flow formulation of the UNDP, we overcome some of the shortcomings mentioned above. We also show that some of the inequalities in Chapter 3 generalize the cuts of this section and are facets of a Set packing polytope that is closely related to the Path-flow formulation.

CHAPTER 3

VALID INEQUALITIES FOR THE NETWORK DESIGN PROBLEM

Model formulation has a significant impact on algorithmic performance for most large-scale mixed-integer programming applications. We have already seen one modeling variation, namely the addition of disaggregated forcing constraints, that is now widely accepted as the preferred formulation for the Plant Location problem (see, for example, Williams [1974], Magnanti and Wong [1981]). For general mixed integer programming problems, Meyer [1981] and Jeroslow and Lowe [1983] have used the theory of disjunctive sets to study integer modeling in a systematic way and derive good formulations. Balas [1983] has shown how a hierarchy of increasingly stronger relaxations can be obtained using the constructs of disjunctive programming.

In general, and especially when using a branching or enumeration scheme with linear programming-based lower bounds, we would like to add to the formulation 'valid' inequalities that

(1) do not exclude any feasible integer solutions of the original problem, and

(2) are not redundant in the LP relaxation of the problem.

Thus, appending these constraints to the formulation might tighten the linear programming relaxation, and thereby yield better lower bounds. Ideally, we must add cuts that are as 'deep' as possible, i.e., they should be facets of the integer polyhedron corresponding to the original problem. There are, in general, two methods for generating such valid inequalities:

(1) by better understanding the structure of the feasible regions for the given mixed-integer problem and its relaxations, and

(2) by using objective function and dual information.

In this chapter, we focus on facet generation methods belonging to the first category.

Recently, there has been renewed interest in using facet generation techniques because of the methods' success in significantly improving solution capabilities for many applications, most notably for the Traveling Salesman problem (see, Crowder and Padberg [1980] and Padberg and Hong [1982]). Several authors have investigated the facial structure of integer polyhedra associated with problems such as the Traveling Salesman problem (Grotschel and Padberg [1979a],[1979b]), the Set packing problem ([Padberg [1973],[1979], Balas and Zemel [1977]), the Uncapacitated Plant Location problem (Guignard [1980], Cornuejols and Thizy [1982], Cho et al. [1983a],[1983b]), the Fixed-charge network problem (Padberg, Van Roy and Wolsey [1982]), the Production lot-sizing problem (Barany, Van Roy and Wolsey [1983]), Scheduling problems (Balas [1984]), and the Linear ordering problem (Grotschel et al. [1984]). Since most of these problems can be modeled as special cases of the Network Design problem, one might expect that some of the facet-generation techniques that have been devised for these problems can be extended to the NDP. In this chapter, we show that this extension is indeed possible; for example, we translate some of the facets for the Set packing polytope into corresponding inequalities for the NDP.

Chvatal [1973] and Martin and Schrage [1982] describe an alternative approach for generating valid inequalities that does not require knowledge

about the facial structure of the integer polyhedra. Their method uses constraint aggregation and coefficient reduction. Martin and Schrage's subset coefficient reduction technique is an extension, to mixed zero-one programming problems, of the Minimal cover cuts proposed by Crowder, Johnson and Padberg [1983]. We show that some of the inequalities derived using the Set packing equivalent of the Uncapacitated Network Design problem can be obtained using the constraint aggregation method as well.

The inequalities derived by all the methods that we discuss in this chapter are based solely on the structure of the constraint set for the given problem. However, objective function and linear programming dual information can also be used for constructing valid inequalities. For example, Magnanti and Wong [1984b] have shown that, for the undirected, uncapacitated Network Design problem, if k' and k" are any two commodities with either a common origin or a common destination, then every optimal solution must satisfy the inequalities

$$x_{ij}^{k'} + x_{ji}^{k''} \leq y_{ij} \quad \text{for all } (i,j) \epsilon A,$$

assuming that $c_{ij} + c_{ji} \geq 0$ for all arcs. Notice that in order to formulate such inequalities we need some information about the objective function (in this case, the assumption that the flow costs are non-negative and are the same for all commodities) and some insight about the structure of optimal solutions. In Chapter 4, we discuss methods for generating inequalities and fixing variables for the NDP using objective function and dual information.

This chapter is organized as follows: We begin by transforming the strong disaggregated Path-flow formulation of the UNDP into a Set packing

problem. We then define the Intersection graph associated with this Set packing problem and explore some of its distinctive features. Next, we consider a class of valid inequalities that are induced by certain subgraphs, called odd holes, of the Intersection graph. We discuss lifting procedures for this class of inequalities and show that the first constraint generation procedure discussed in Chapter 2 is, in fact, equivalent to a method for identifying certain special types of odd holes in the Intersection graph. We also show how the odd hole inequalities can be generated using a constraint aggregation/coefficient reduction procedure. Next, we consider certain specially structured Network Design problems, including those defined over graphs with a single source. We exploit the similarity between these problems and the Plant Location problem to extend some of Cornuejols and Thizy's [1982] results on facial inequalities for the Plant Location problem; one of these inequalities is the cut based on cyclic matrices discussed in Chapter 2. We also show how the lifted versions of these inequalities can be transformed into valid cuts for the Arc-flow formulation of the UNDP. We conclude this chapter by describing some "cutset" inequalities for the Capacitated Network Design problem.

As is evident from this summary, we will rely quite heavily on the characterizations of the facial structure of Set packing polyhedra obtained by Chvatal [1975], Balas and Zemel [1977], Padberg [1973], and Zemel [1978] Cornuejols and Thizy [1982] used these results to develop facial inequalities for the Uncapacitated Plant Location problem. In the same spirit, we apply the Set packing results to the Network Design problem. Indeed, all but the last class of inequalities discussed in this chapter

have the same form as those given by Cornuejols and Thizy. We emphasize that the major focus of this chapter is on describing various classes of valid inequalities, rather than identifying those that are violated by a given fractional extreme point; also, we will be mainly working with the strong disaggregated Path-flow formulation of the UNDP.

3.1 The Uncapacitated Newtork Design Problem in Set Packing form

Let [UNDSPF] denote the strong disaggregated Path-flow formulation of the Uncapacitated Network Design problem, that is obtained by replacing the forcing constraints

$$f_{p(k)} \leq y_{ij} \qquad \text{for all } k \epsilon K,\ p(k) \epsilon P(k), \text{ and } (i,j) \epsilon p(k) \tag{2.3}$$

by the tighter constraints

$$\sum_{p(k) \epsilon P_{ij}(k)} f_{p(k)} \leq y_{ij} \qquad \text{for all } k \epsilon K \text{ and } (i,j) \epsilon A \tag{2.3a}$$

where $P_{ij}(k)$ is the set of all paths for commodity k that contain arc (i,j).

In this section, we first transform the formulation [UNDSPF] into an equivalent Set packing problem [SP] and define the Intersection graph G_I corresponding to this problem. This Intersection graph is useful because of the one-to-one correspondence between its vertex packings (or independent sets) and the feasible solutions of [SP]. Thus, inequalities that do not exclude any vertex packing of G_I are valid for [SP]. We describe some special features of the Intersection graph corresponding to

the NDP, and outline the general form of inequalities that are induced by subgraphs of this Intersection graph. In subsequent sections, we consider specializations of these inequalities corresponding to several specially structured subgraphs of G_I. We conclude this section by discussing some properties of facets for the Set packing polytope.

We can convert [UNDSPF] into a Set packing problem by performing the following steps.

(1) Substitute $y_{ij} = 1 - \bar{y}_{ij}$ for all arcs $(i,j) \epsilon A$. Constraints (2.3a) can then be rewritten as

$$\sum_{p(k) \epsilon P_{ij}(k)} f_{p(k)} + \bar{y}_{ij} \leq 1 \quad \text{for all } k \epsilon K \text{ and } (i,j) \epsilon A.$$

(2) Add non-negative 'artificial' variables s_k, for all $k \epsilon K$, to the demand constraints (2.2); let the objective function coefficients for each of these artificial variables be a very large number M that is greater than, say, $\Sigma_k \Sigma_{p(k)} c_{p(k)} + \Sigma_{(i,j)} F_{ij}$. Thus, constraints (2.2) are now of the form:

$$\sum_{p(k)} f_{p(k)} + s_k = 1 \qquad \text{for all } k \epsilon K. \qquad (2.2a)$$

Our choice of the coefficient M ensures that each s_k will be 0 in any optimal solution of the transformed problem; thus, every optimal solution of this new problem will be feasible in our original problem.

(3) Use equations (2.2a) to substitute for the variables s_k in the objective function and the constraints. The non-negativity constraints on the artificial variables are thus transformed into the constraints

$$\sum_{p(k)} f_{p(k)} \leq 1 \qquad \text{for all } k \epsilon K.$$

As a result of these operations, [UNDSPF] can be rewritten as

[SP]

$$\text{Maximize} \quad \sum_{k} \sum_{p(k)} (M - c_{p(k)}) f_{p(k)} + \sum_{(i,j)} F_{ij} \bar{y}_{ij} \tag{4.1}$$

subject to

$$\sum_{p(k)} f_{p(k)} \leq 1 \qquad \text{all } k \epsilon K \tag{4.2}$$

$$\sum_{p(k) \epsilon P_{ij}(k)} f_{p(k)} + \bar{y}_{ij} \leq 1 \qquad \text{all } k \epsilon K, \ (i,j) \epsilon A \tag{4.3}$$

$$f_{p(k)} = 0 \text{ or } 1 \qquad \text{all } k \epsilon k, \ p(k) \epsilon P(k) \tag{4.4a}$$

$$\bar{y}_{ij} = 0 \text{ or } 1 \qquad \text{all } (i,j) \epsilon A \tag{4.4b}$$

We have omitted the constant terms in the objective function and represented the problem in maximization form by multiplying all the objective function coefficients by -1. Also notice that we have restricted the path-flow variables $f_{p(k)}$ to be either 0 or 1; this restriction is valid since [UNDSPF] has an optimal solution in which all the flow values are integral.

The Intersection graph and an equivalent Vertex Packing problem:

In the zero-one programming problem [SP], all the constraint coefficients are either 0 or 1, and all the constraints are "$\leq$" inequalities with right-hand side values of 1. [SP] is, therefore, a Set packing formulation. Any Set packing problem can be transformed into an equivalent optimization problem defined over a related undirected graph called the Intersection graph, which is defined below.

Definition 3.1:

> The INTERSECTION GRAPH corresponding to any given zero-one matrix A contains a vertex for every column of A and an edge connecting nodes j

and j' if and only if the corresponding columns j and j' of A have a nonzero inner product.

Observe that, since A is a zero-one matrix, columns j and j' have a nonzero inner product if and only if, for at least one row i of A, $a_{i,j} = a_{i,j'} = 1$. Consequently, any feasible zero-one solution to the system of equations $Ax \leq 1$ corresponds to an assignment of 0's and 1's to the respective vertices of the Intersection graph such that no two adjacent vertices are both assigned the value 1. We elaborate on this correspondence later.

We let $G_I:(V_I,E_I)$ denote the Intersection graph corresponding to the Set packing problem [SP]. This undirected graph contains

(a) one vertex, designated as vertex [p(k)], corresponding to each path-flow variable $f_{p(k)}$, for all $k \epsilon K$ and $p(k) \epsilon P(k)$. We will refer to these vertices as f-nodes,

(b) one vertex, denoted as [ij] and called a y-node, for each 'design' variable $\bar{y}_{ij}$, for all $(i,j) \epsilon A$,

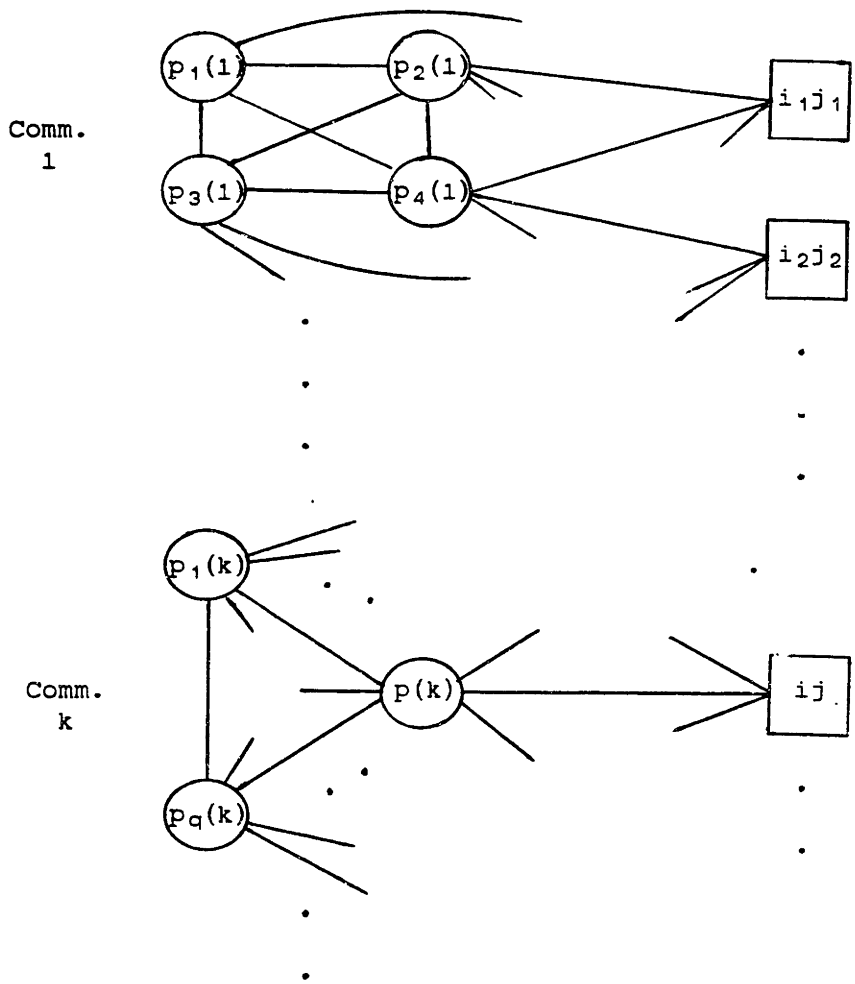
(c) an edge between every pair of f-nodes, say nodes [p(k)] and [p'(k)], that correspond to the same commodity k, for all $k \epsilon K$ and $p(k), p'(k) \epsilon P(k)$. These edges are induced by the 'demand' constraints (4.2) (i.e., they are included in G_I because both the flow variables $f_{p(k)}$ and $f_{p'(k)}$ have a coefficient of 1 in the demand constraint (4.2) corresponding to commodity k), and

(d) an edge from f-node [p(k)] to y-node [ij], for $k \epsilon K$, $p(k) \epsilon P(k)$, and $(i,j) \epsilon A$, if and only if arc (i,j) belongs to path p(k) in the original graph. These edges are induced by the forcing constraints (4.3) of [SP].

We will denote the set of all f-nodes of G_I as F and the set of all y-nodes as Y. Thus, $V_I = F \cup Y$.

Figure 3 illustrates the structure of a typical Intersection graph G_I corresponding to the Set packing problem [SP]. Because of the special

Figure 3: Structure of the Intersection Graph G_I



structure of the Path-flow formulation [UNDSPF], and hence of [SP], G_I has certain distinctive characteristics. For instance,

(1) no two y-nodes are adjacent in G_I,

(2) f-nodes that correspond to different commodities, such as [p(k)] and [p'(k')], are not adjacent in G_I, and

(3) for each commodity $k \epsilon K$, the subgraph G_I induced by the $|P(k)|$ corresponding f-nodes [p(k)], for all $p(k) \epsilon P(k)$, is a COMPLETE graph.

We will exploit these properties of G_I as we develop valid inequalities for [UNDSPF] later in this chapter. As an aside, we note that the Intersection graph corresponding to the Plant Location problem is considerably simpler than that for the general NDP. For example, in the Plant Location Intersection graph, each f-node [i(k)] is adjacent to exactly one y-node, namely the node [0i]. The more complex structure of the general graph G_I complicates the analysis somewhat.

Valid Inequalities induced by subgraphs of G_I:

Let us now further explore the relationship between the Set packing problem [SP] and the associated Intersection graph G_I.

Definition 3.2:

> A VERTEX PACKING of a graph G' is an assignment of 0's and 1's to the vertices of G' such that no two adjacent vertices of G' are both assigned the value 1. The subset of vertices with value 1 in any vertex packing of G' is referred to as an INDEPENDENT or STABLE set.

Because of the manner in which the Intersection graph G_I is constructed, every packing of G_I is a feasible solution of [SP] (i.e., the solution $(f, \bar{y})$, in which the values of the variables $f_{p(k)}$ and $\bar{y}_{ij}$ are the same as those that are assigned to the respective f- and y- nodes of G_I, in any

vertex packing of G_I, must be feasible in [SP]). Similarly, for every feasible solution of [SP] there is a corresponding vertex packing of G_I. Consequently, the Set packing problem [SP] is equivalent to finding the maximum weight independent set of G_I, when the f-nodes [p(k)] and the y-nodes [ij] are, respectively, assigned the weights $(M-c_{p(k)})$ and F_{ij}. Furthermore, any valid constraint on the assignment of 0's and 1's to the vertices of G_I (i.e., a constraint that does not exclude any independent set of G_I) can be translated into a valid inequality for the formulation [SP], and hence for [UNDSPF]. It is this last relationship between [SP] and G_I that is of greatest interest to us.

We now illustrate how constraints for the Vertex packing problem defined over G_I can be translated into valid inequalities for [SP].

Definition 3.3:

> The INDEPENDENCE NUMBER of a graph G', denoted as IND(G'), is the maximum cardinality of any independent set in G'.

Consider any subgraph G' of the Intersection graph G_I. Clearly, the number of vertices in G' that belong to any independent set of G_I must not exceed IND(G') (since the subset of vertices of G' in any independent set of G_I must be independent in G'). Therefore, if F' and Y' are the subset of f- and y- nodes respectively of G', then the inequality

$$\sum_{p(k)\in F'} f_{p(k)} + \sum_{ij\in Y'} \bar{y}_{ij} \leq IND(G') \qquad (5.1)$$

is valid for [SP]. We will refer to constraint (5.1) as the inequality INDUCED by the subgraph G' of G_I. Every subgraph of G_I induces such a valid inequality for [SP]. However, many of these inequalities are

redundant even in the LP relaxation of [SP] and some of them already belong to the formulation [SP]. For instance, suppose G_I contains the following cycle with 4 nodes:

$$[p(k)] - [i1,j1] - [p'(k')] - [i2,j2] - [p(k)].$$

The independence number of this cycle is 2 and the inequality (5.1) induced by it is

$$f_{p(k)} + f_{p'(k')} + \bar{y}_{i1,j1} + \bar{y}_{i2,j2} \leq 2.$$

However, this inequality is just the sum of the two constraints

$$f_{p(k)} + \bar{y}_{i1,j1} \leq 1, \text{ and}$$

$$f_{p'(k')} + \bar{y}_{i2,j2} \leq 1,$$

that are implied by the forcing constraints (4.2) of [SP]; hence, this cycle of G_I induces an inequality that is redundant in the LP relaxation of [SP]. In fact, we can extend this argument to show that the inequality induced by any <u>even</u> cycle (i.e., a cycle with an even number of edges and vertices) must be redundant in [SP]. Similarly, in the next section we show that all the inequalities that are induced by cliques of G_I are already included in the formulation [SP].

Our strategy, then, is to identify subgraphs of G_I that induce valid inequalities (5.1) that are not redundant in the LP relaxation of [SP]. Trotter [1975] has developed such inequalities for the general Set packing problem based on certain regular subgraphs called webs. Similarly, Cornuejols and and Thizy [1982] use specially structured subgraphs of the of the Intersection graph to generate several classes of nonredundant valid inequalities for the Plant Location problem. We will identify similar subgraphs in the Intersection graph G_I of the NDP. For all the subgraphs that we consider, the Independence numbers can be determined easily. In

addition, the inequalities induced by these subgraphs are the tightest possible in the sense that they define facets of certain corresponding lower dimensional integer polytopes. We will also discuss lifting procedures that extend the basic inequality induced by any subgraph G' of G_I to include variables whose corresponding vertices do not belong to G'.

We next outline some general properties of facets for the Set packing polytope after introducing some notation and terminology. For any subgraph G' of G_I, let

V(G') = set of all vertices of G', $V(G') \subseteq V_I$,

E(G') = set of all edges of G', $E(G') \subseteq E_I$,

F(G') = set of all f-nodes of G', i.e., F(G') = F V(G'), and

Y(G') = set of all y-nodes in G', i.e., Y(G') = Y V(G').

Definition 3.4:

> A subgraph G':(V(G'),E(G')) is said to be a NODE-INDUCED subgraph of G_I if E(G') contains all arcs of E_I, both of whose terminal vertices belong to the subset of vertices $V(G') \subseteq V_I$. In this case, we say that the node subset V(G') INDUCES subgraph G'.

It is easy to show that, among all the subgraphs of G_I with the same vertex set $V' \subseteq V_I$, the subgraph that is induced by V' produces the strongest inequality of the form (5.1). Therefore, we will henceforth consider only node-induced subgraphs of G_I.

Facets of the Set packing polytope:

Let P_I be the convex hull of the vertex packings of G_I. Because of the correspondence between Set packing solutions and the vertex packings of

the corresponding Intersection graph, P_I is also the convex hull of the feasible (integer) solutions of [SP]. We will refer to P_I as the SET PACKING or VERTEX PACKING POLYTOPE. The zero vector and the $(|F|+|Y|)$ unit vectors (in $R^{|F|+|Y|}$) are affinely independent solutions of [SP]; hence, P_I is full-dimensional, i.e., $\dim(P_I) = |F| + |Y|$. For any given subgraph G' of G_I, let

$$P_I^{G'} = \text{conv}\{(f, \bar{y}): (f, \bar{y}) \text{ satisfies (4.2)-(4.4b), and}$$
$$f_{p(k)} = \bar{y}_{ij} = 0 \text{ for all } [p(k)],[ij] \notin V(G') \} .$$

Thus, $P_I^{G'}$ is the convex hull of all feasible solutions of [SP] with the variables corresponding to nodes not belonging to G' equal to zero.

Definition 3.5:

The intersection of P_I and the hyperplane

$$\sum_{p(k)} a_{p(k)} f_{p(k)} + \sum_{(i,j)} b_{ij}\bar{y}_{ij} = b_0$$

is called a FACET of the polytope P_I if

(1) the inequality $af + b\bar{y} \leq b_0$ is satisfied by all the points in the polytope P_I, and

(2) exactly $d = \dim(P_I)$ affinely independent points $(f^i, \bar{y}^i)$, for $i = 1,2,....,d$, of P_I satisfy $af^i + b\bar{y}^i = b_0$.

For brevity, we will often refer to the inequality $af + b\bar{y} \leq b_0$ itself as the facet of P_I.

The inequalities (5.1) induced by all the subgraphs G' that we consider in the next few sections are facets of the lower dimensional polytope $P_I^{G'}$. These inequalities can be lifted into facets of P_I using the lifting procedure described later. Notice that, since P_I is full-dimensional, all its facets are uniquely defined (up to a positive multiplicative constant). Furthermore, if we let P_I' denote the integer polytope of the original

strong Path-flow formulation [UNDSPF], then every facet of P_I' is contained in at least one facet of P_I.

We now make a few observations about the general form of the inequalities $af + b\bar{y} \leq b_0$ that define facets of P_I. For general Set packing problems, Balas and Padberg [1976] have noted that the non-negativity constraints define "trivial" facets of the Vertex packing polytope. Furthermore, these constraints are the only facets of P_I with zero right-hand side constant (i.e., with $b_0 = 0$). Thus, the inequalities

$$f_{p(k)} \geq 0 \qquad \text{for all } [p(k)] \in F, \text{ and}$$

$$\bar{y}_{ij} \geq 0 \qquad \text{for all } [ij] \in Y$$

define the trivial facets of P_I. Also, in any nontrivial facet of the Vertex packing polytope, the coefficients in the left-hand side must be non-negative and the right-hand side constant must be strictly positive (See, Padberg [1979]). For illustrative purposes, we present a proof of this claim.

Lemma 3: (Padberg [1979])

Every nontrivial facet $af + b\bar{y} \leq b_0$ of P_I satisfies

(1) $a_{p(k)} \geq 0$, $b_{ij} \geq 0$, for all $[p(k)] \in F$ and $[ij] \in Y$, and

(2) $b_0 > 0$.

Proof:

We will use the fact that if $af + b\bar{y} \leq b_0$ is a nontrivial facet of P_I, then for each path $p(k)$, there is at least one solution $(f, \bar{y}) \in P_I$ with $f_{p(k)} = 1$ and satisfying $af + b\bar{y} = b_0$ (i.e. lying on the facet of interest). (Otherwise, all points satisfying this equality must lie on the trivial facet $f_{p(k)} = 0$, implying that the facets are not uniquely

defined - a contradiction). Similarly, corresponding to each arc $(i,j) \epsilon A$, there is at least one solution $(f, \bar{y})$ on the facet with $\bar{y}_{ij} = 1$.

Suppose $af + b\bar{y} \leq b_0$ is a facet of P_I with $a_{p(k)} < 0$ for some path $p(k)$. By the previous assertion, there must be at least one feasible Set packing solution $(f, \bar{y})$ with $f_{p(k)} = 1$ and satisfying $af + b\bar{y} = b_0$. The solution remains feasible in [SP] if $f_{p(k)}$ is set equal to 0; however, for the new solution we have

$$af + b\bar{y} = b_0 - a_{p(k)} > b_0, \quad \text{since } a_{p(k)} < 0 \text{ by assumption,}$$

which implies that the inequality $af + b\bar{y} \leq b_0$ is not valid for P_I - a contradiction. Hence, $a \geq 0$. Similarly, we can show that the coefficients b_{ij} must all be non-negative. Now, $f = 0$, $\bar{y} = 0$, is a feasible solution of [SP]; hence, b_0 must be greater than or equal to zero in every facet of P_I. Since, the non-negativity constraints are the only facets with $b_0 = 0$, every nontrivial facet must have $b_0 > 0$.

Notice also that, in any facet of P_I, all the left-hand side coefficients $a_{p(k)}$ and b_{ij} must be less than or equal to the right-hand side coefficient b_0. Otherwise, suppose for some path $p(k)$, $a_{p(k)} > b_0$. Then, any solution of [SP] in which $f_{p(k)}$ is 1 violates the facial inequality $af + b\bar{y} \leq b_0$; in other words, all feasible Set packing solutions must lie on the trivial facet $f_{p(k)} = 0$ - a contradiction. Similarly, we can show that $b_{ij} \leq b_0$, for all arcs $(i,j) \epsilon A$, in every facial inequality.

We will now begin examining subgraphs of G_I that induce facets of the

polytope P_I. For general Set packing problems, Trotter [1975] has described a large class of node-induced subgraphs called WEBS whose corresponding inequalities (5.1) define facets of P_I. Cliques and odd holes are special cases of such webs. We will first discuss the nature of the inequalities generated by these two facet-producing subgraphs. Subsequently, we will show that, for the special Set packing derived from the Network Design problem, the corresponding Intersection graph G_I does not contain any other types of webs.

3.2 Valid Inequalities Induced by Cliques of the Intersection Graph G_I

Definition 3.6:

A CLIQUE of a graph G is a maximal complete subgraph of G.

The Independence number of every clique is 1, since cliques are complete subgraphs. Therefore, the inequality (5.1) induced by any clique C of G_I is

$$\sum_{p(k)\in F(C)} f_{p(k)} + \sum_{ij\in Y(C)} \bar{y}_{ij} \leq 1 \qquad (5.2)$$

Besides showing that (5.2) is a facet of P_I, Padberg [1973] has proved that all facets of P_I that have zero-one coefficients must necessarily be induced by cliques of G_I. This result is expressed as the following theorem.

Theorem: (Padberg [1973])

Let F' and Y' be subsets of f- and y- nodes, respectively, of the Intersection graph G_I. Then, the inequality

$$\sum_{p(k)\epsilon F'} f_{p(k)} + \sum_{ij\epsilon Y'} \bar{y}_{ij} \leq 1$$

is a facet of P_I if and only if F' U Y' is the node set of a clique of G_I.

Recall, from our earlier discussion, that in every facet of P_I, the coefficients on the left-hand side must be non-negative and less than or equal to the right-hand side constant b_0. Therefore, this theorem characterizes all facets of P_I that have integer coefficients and a right-hand side value of 1.

Let us now apply this result to the Set packing problem [SP]. The Intersection graph G_I corresponding to [SP] contains two types of cliques.

(a) For each commodity $k\epsilon K$, the set of all f-nodes [p(k)] corresponding to commodity k induces a clique of cardinality $|P(k)|$.

As noted earlier, for any commodity $k\epsilon K$, the subgraph induced by the f-nodes [p(k)], for all $p(k)\epsilon P(k)$, is a complete subgraph of G_I. However, this subgraph is maximal (i.e., it is not a proper subgraph of a 'larger' complete subgraph of G_I) only under the following assumption:

> In the original network, for all arcs $(i,j)\epsilon A$ and for each commodity $k\epsilon K$, there is at least one path p(k) that does not contain (i,j).

As the following argument demonstrates, there is no loss of generality in making this assumption. Suppose some arc (i,j) belongs to all the paths $p(k)\epsilon P(k)$, for some commodity $k\epsilon K$. Then, arc (i,j) must necessarily be included in the Network design, and the corresponding design variable y_{ij} can be eliminated (by setting it equal to 1) from the UNDP formulation. The assumption will then be valid for the new formulation.

(b) For each arc $(i,j)\epsilon A$ and each commodity $k\epsilon K$, the y-node [ij] together with the set of f-nodes [p(k)], for all paths p(k) that contain arc (i,j) (i.e., for all $p(k)\epsilon P_{ij}(k)$), induces a clique of cardinality $(|P_{ij}(k)| + 1)$.

(Recall that $P_{ij}(k)$ is the set of all paths for commodity k that contain arc (i,j)).

Therefore, for the Set packing problem [SP], the inequalities (5.2) induced by the two types of cliques of G_I are

$$\sum_{p(k)} f_{p(k)} \leq 1 \qquad \text{for all } k \epsilon K, \text{ and}$$

$$\sum_{p(k) \epsilon P_{ij}(k)} f_{p(k)} \leq y_{ij} \qquad \text{for all } (i,j) \epsilon A \text{ and } k \epsilon K.$$

Both these constraints, however, already belong to the formulation [SP] (constraints (4.2) and (4.3)). Therefore, cliques of G_I do not produce any new facets of P_I. Next, we will examine inequalities induced by subgraphs called odd holes of G_I.

3.3 Valid Inequalities Induced by Odd Holes of the Intersection Graph G_I

Definition 3.7:

A node-induced subgraph H that is a simple cycle with $h \geq 4$ edges is called a HOLE. H is an ODD HOLE if h is odd.

The Independence number IND(H) of any odd hole H is (h-1)/2, where h is the number of vertices (and edges) of H. Thus, odd holes of the Intersection graph G_I induce the inequality

$$\sum_{p(k) \epsilon F(H)} f_{p(k)} + \sum_{ij \epsilon Y(H)} \bar{y}_{ij} \leq (h-1)/2 \qquad (5.3)$$

Padberg [1973] has shown that (5.3) is a facet of the lower-dimensional polytope P_I^H; he has also proved the existence of at least one lifting of (5.3) that is a facet of the Set packing polytope P_I.

Theorem: (Padberg [1973])

Let H be an odd hole of G_I induced by the vertex set

$V(H) = F(H) \cup Y(H)$, $F(H) \subseteq F$, $Y(H) \subseteq Y$. Then, (5.3) is a facet of P_I^H. Furthermore, this inequality is a face of P_I, and there exist integer coefficients $a_{p(k)}$ and b_{ij}, $0 \leq a_{p(k)}, b_{ij} \leq (h-1)/2$, such that

$$\sum_{p(k) \in F(H)} f_{p(k)} + \sum_{p(k) \in F(H)} a_{p(k)} f_{p(k)} + \sum_{ij \in Y(H)} \bar{y}_{ij} + \sum_{ij \in Y(H)} b_{ij} \bar{y}_{ij} \leq (h-1)/2 \qquad (5.4)$$

is a facet of P_I.

We now consider the special structure of odd holes in the Intersection graph G_I corresponding to the Set packing problem [SP]. Subsequently, we discuss a facet lifting procedure and examine the structure of lifted facets obtained from the inequalities (5.1) induced by two special types of odd holes. We then show that the first constraint generation procedure discussed in Chapter 2, in fact, identifies odd holes of G_I belonging to the first of these families; also, the second class of odd holes are related to certain subgraphs with cyclic incidence matrices, similar to those described in Chapter 2.

3.3.1 The structure of odd holes in G_I:

As mentioned earlier, the Intersection graph G_I corresponding to the Set packing formulation [SP] has several distinctive features. For instance, no two y-nodes are adjacent in G_I, f-nodes corresponding to different commodities are not adjacent in G_I, and every f-node [p(k)] is adjacent to each of the other f-nodes corresponding to the same commodity k. Because of these characteristics, odd holes of G_I have certain special properties. We next discuss some of these properties.

Suppose the subset of vertices $V(H) = F(H) \cup Y(H)$ induces an odd hole H of G_I, with $h = |V(H)| = |F(H)|+|Y(H)| \geq 5$ and odd. (F(H) and Y(H) are, respectively, the set of f- and y-nodes of H.) Let K' be the subset of commodities, at least one of whose f-nodes belongs to the set F(H), i.e.,

$$K' = \{k \epsilon K: [p(k)] \epsilon F(H) \subseteq F\} .$$

Then,

(a) For each commodity $k \epsilon K'$, F(H) contains at most two f-nodes corresponding to commodity k .

Proof:
Suppose not, i.e., suppose F(H) contains more than two f-nodes corresponding to some commodity $k \epsilon K'$. Let [p(k)], [p'(k)], and [p"(k)] be three of these nodes. These nodes induce the cycle [p(k)]-[p'(k)]-[p"(k)]-[p(k)] in G_I; this cycle must be a proper subgraph of H, since $h \geq 4$. Therefore, H is not a simple cycle - a contradiction.

Notice that, if F(H) contains two f-nodes corresponding to some commodity $k \epsilon K'$, then these two nodes must necessarily be adjacent in the odd hole H. Also, (a) implies that

$$|F(H)| = \text{the number of f-nodes in H} \leq 2|K'|.$$

(b) $|K'| \geq 2$; i.e., F(H) must contain f-nodes corresponding to at least 2 commodities.

Proof:
Suppose $|K'| = 0$; then the set F(H) is empty and V(H) consists of only y-nodes. But then the subgraph induced by V(H) is not connected, since none of the $h \geq 5$ y-nodes of H are connected in G_I. If $|K'| = 1$, then from (a) we know that $|F(H)|$ must be ≤ 2. Therefore, V(H) contains at least 3 y-nodes and at most 2 f-nodes (which are adjacent), implying that the subgraph induced by V(H) is not a cycle. Therefore, F(H) must contain f-nodes corresponding to at least 2 commodities in order for the induced subgraph to be an odd hole.

Using similar arguments, we can prove that the subset of nodes V(H) that induces the odd hole H must necessarily satisfy the following conditions:

(c) $|Y(H)| \geq 2$; i.e., y-nodes corresponding to at least 2 different arcs of the original network must be included in V(H).

(d) $|Y(H)| = |K'|$; i.e., the number of y-nodes in V(H) is equal to the number of commodities whose f-nodes are contained in V(H).

The f-nodes corresponding to the $|K'|$ commodities and the y-nodes corresponding to the $|Y(H)|$ arcs alternate in the odd hole H.

(e) Let $K'' \subseteq K'$ be the subset of commodities, exactly two of whose f-nodes belong to F(H). Then, $|K''|$ must be odd (and hence $|K''| > 1$).

This result follows from the relationships

$$h = |F(H)| + |Y(H)|$$

$$= |K'| + |K''| + |Y(H)|$$

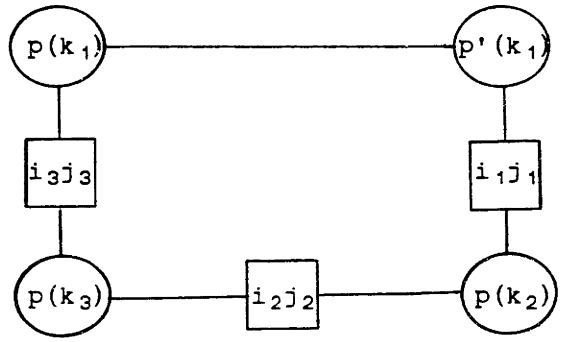
$$\Longrightarrow \quad \underline{h = 2|Y(H)| + |K''|} \qquad \text{since } |K'| = |Y(H)| \text{ from (d)}$$

$$\Longrightarrow \quad h \text{ is odd if and only if } |K''| \text{ is odd.}$$

Notice that, in any odd hole H of G_I, each y-node [ij] is adjacent to exactly two f-nodes, say [p(k)] and [p'(k')], that correspond to different commodities. Figure 4 contains two examples that illustrate the structure of odd holes in G_I. In all the diagrams, we use squares to represent y-nodes and circles to represent f-nodes.

For convenience, we will assume that odd holes of G_I have the structure shown in Figure 5. Thus, V(H) contains f-nodes corresponding to q commodities, the first r of which have exactly two corresponding f-nodes in V(H), designated as $[p(k_t)]$ and $[p'(k_t)]$, for $1 \leq t \leq r$. Let $[p(k_t)]$ denote the single f-node in V(H) corresponding to the commodities k_t, for $r < t \leq q$. The odd hole H contains q y-nodes, indexed as $[i_t, j_t]$, $1 \leq t \leq q$; the t^{th} y-node $[i_t, j_t]$ is adjacent in H to the f-nodes $[p'(k_t)]$ and $[p'(k_{t+1})]$ if $t \leq r$; if $t > r$, $[i_t, j_t]$ is adjacent to $[p(k_t)]$ and $[p(k_{t+1})]$. With this structure,

Figure 4: Examples of Odd Holes in the Intersection Graph G_I

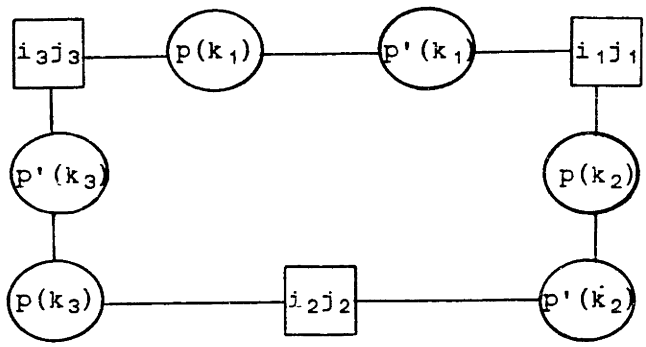


$K' = \{k_1, k_2, k_3\}$

$K'' = \{k_1\}$

$F(H) = \{[p(1)], [p'(1)], [p(2)], [p(3)]\}$

$Y(H) = \{[i_1j_1], [i_2j_2], [i_3j_3]\}$

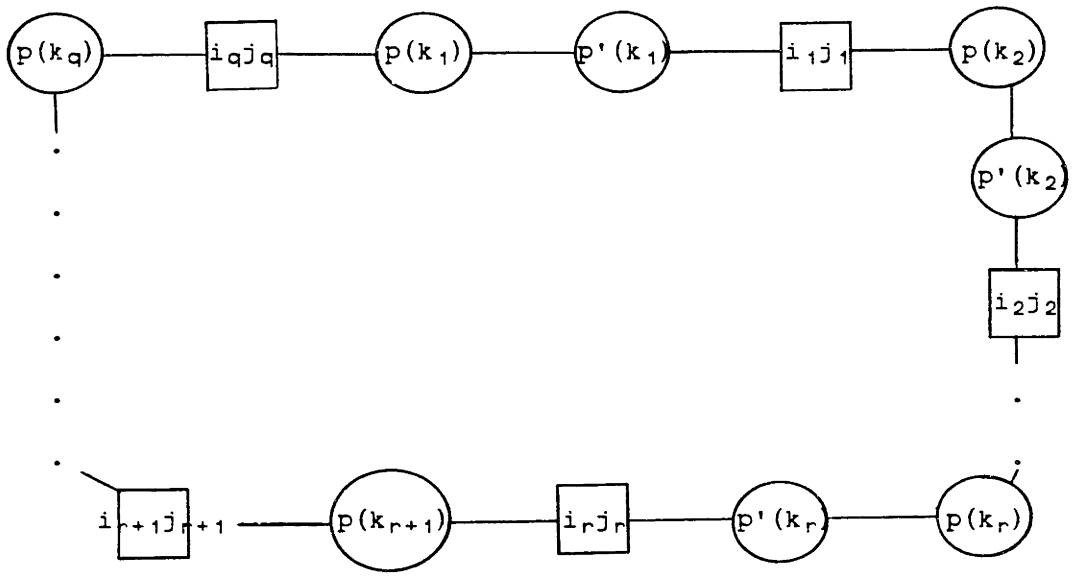


$K' = K'' = \{k_1, k_2, k_3\}$

$F(H) = \{[p(k_t)], [p'(k_t)], t=1,2,3\}$

$Y(H) = \{[i_t, j_t], t=1,2,3\}$

Figure 5: Structure of General Odd Holes



$$K' = \{k_t : t=1,2,\ldots,q\},$$

$$K'' = \{k_t : t=1,2,\ldots,r\}, \text{ with } r \leq q,$$

$$F(H) = \{[p(k_t)], [p'(k_t)] : 1 \leq t \leq r\} \cup \{[p(k_t)] : r < t \leq q\}, \text{ and}$$

$$Y(H) = \{(i_t, j_t) : 1 \leq t \leq q\}.$$

Consequently, $r = |K''|$ must be odd and the Independence number of H is

$$IND(H) = (h-1)/2 = (2q + r - 1)/2.$$

Having identified the special properties of odd holes in the Intersection graph G_I, we now want to determine the form of the lifted facets (5.4) derived from the inequalities (5.3) that are induced by these odd holes. Before discussing the lifted inequality, however, we will briefly indicate how the inequality (5.3) induced by any odd hole H of G_I

can be generated using constraint aggregation.

3.3.2 Deriving (5.3) by constraint aggregation:

We will now demonstrate how the inequalities (5.3) induced by odd holes of the Intersection graph G_I can alternatively be obtained by aggregating some selected constraints (and rounding down the right-hand side constant of the aggregated constraint) of the Path-flow formulation of the UNDP. Recall that, a y-node [ij] is adjacent to a f-node [p(k)] in the Intersection graph G_I if and only if, in the original graph G, arc (i,j) belongs to path p(k). Because of the structure of odd holes that we outlined in the last subsection, the following inequalities are contained in, or are valid for, the Path-flow formulation of the UNDP.

(a) $f_{p(k1)} \leq y_{iq,jq}$,

(b) $f_{p(kt)} \leq y_{it-1,jt-1}$ for all $2 \leq t \leq r$,

(c) $f_{p'(kt)} \leq y_{it,jt}$ for all $1 \leq t \leq r$,

(d) $f_{p(kt)} \leq y_{it-1,jt-1}$ for all $r < t \leq q$,

(e) $f_{p(kt)} \leq y_{it,jt}$ for all $r < t \leq q$, and

(f) $f_{p(kt)} + f_{p'(kt)} \leq 1$ for all $1 \leq t \leq r$.

(Note: For convenience, we denote the flow on path $p(k_t)$ as $f_{p(kt)}$ and the design variable corresponding to arc (i_t,j_t) as $y_{it,jt}$).

Adding all these constraints we obtain the composite inequality.

$$2 \sum_{t=1}^{r} (f_{p(kt)} + f_{p'(kt)}) + 2 \sum_{t=r+1}^{q} f_{p(kt)} + 2 \sum_{t=1}^{q} \bar{y}_{it,jt} \leq 2q + r,$$

where $\bar{y}_{ij} = 1 - y_{ij}$.

Note that the right-hand side constant of this constraint is odd since r,

the number of commodities for which two corresponding f-nodes belong to the odd hole H, is odd. We can divide the entire constraint by 2 and round down the right-hand side, since there is an optimal UNDP solution in which all the y- and f- variables are integer. As a result, we obtain the cut

$$\sum_{p(k)\in F(H)} f_{p(k)} + \sum_{ij\in Y(H)} \bar{y}_{ij} \leq q + (r-1)/2 = IND(H),$$

which is identical to the inequality (5.3) induced by the odd hole H. This equivalence highlights the relationship between the two constraint generation procedures. This relationship is not surprising, however, since Chvatal [1973] has shown that all valid inequalities and facets for any bounded polyhedron can be generated by iteratively aggregating constraints and reducing coefficients. Finally, note that the sets of constraints (a) - (e) that we aggregated are relaxations of the strong forcing constraints (2.3a) in [UNDSPF]. Thus, this procedure may be viewed as a process of both relaxation and aggregation of constraints in the original IP formulation.

3.3.3 A Sequential Facet Lifting procedure:

As we have noted previously, Padberg [1973] has shown how to lift the odd hole-induced inequality (5.3), using a sequential procedure, to produce a facet (5.4) of the Set packing polytope. We first describe this procedure in its general form; later, we show that when this procedure is specialized to the odd hole inequalities (5.3), the coefficients of the corresponding lifted facets (5.4) must necessarily satisfy certain conditions. The sequential lifting procedure can also be applied to the more general subgraph-induced inequality (5.1). We will, therefore, refer to the general procedure subsequently as well, when we consider other

facet-inducing subgraphs of the Intersection graph G_I.

Given a lower dimensional facet or valid inequality, the process of extending this inequality to other variables (that are not included in the given inequality) is called a LIFTING PROCEDURE. The objective, while 'lifting' any given inequality, is to determine coefficients for the additional variables so that the resulting inequality is as tight as possible. Lifting procedures discussed in the literature fall into two categories - Simultaneous procedures and Sequential procedures. As the names imply, simultaneaous procedures attempt to determine the coefficients for several additional variables at every stage, while sequential procedures extend the inequalities one variable at a time. Facet lifting procedures have been proposed in the literature for the Set packing problem (Padberg [1973], Nemhauser and Trotter [1974], Trotter [1975]), the Traveling Salesman problem (Grotschel and Padberg [1979b]), and for general Zero-One programming problems (Zemel [1978], Padberg [1973], Martin and Schrage [1982], Crowder et al. [1983]).

In the current context, given a facet of the lower dimensional polytope $P_I^{G'}$ that is induced by a subgraph G' of the Intersection graph G_I, we want to determine lifted inequalities that are facets of P_I. We next present the procedure proposed by Padberg [1973] for generating such facets for Set packing polytopes. This procedure is sequential, and Padberg has shown that, starting with a facet of $P_I^{G'}$, the procedure always generates at least one facet of P_I; also, all the coefficients in the resulting lifted facet are integral.

For ease of exposition, we will adopt the following notation. Given any subgraph G' of G_I, let V'(G') denote the set of all vertices of G_I that do not belong to G'. Suppose the nodes of V'(G') are ordered arbitrarily. We will let u_t denote the t^{th} vertex of V'(G'); thus, u_t could either be a f-node or a y-node that does not belong to the subgraph G'. Let x_t be the f or $\bar{y}$ variable corresponding to vertex u_t.

Suppose the inequality (5.1) induced by a subgraph G' of the Intersection graph is a facet of $P_I^{G'}$. Then, the following sequential lifting procedure proposed by Padberg [1973] generates a lifted inequality of the form

$$\sum_{p(k)\in F(G')} f_{p(k)} + \sum_{ij\in Y(G')} \bar{y}_{ij} + \sum_{t=1}^{T} d_t x_t \leq IND(G') \qquad (5.5)$$

where $T = |V'(G')|$ = number of vertices of G_I not in G'.

Sequential Lifting procedure:

Step 0: Set t = 1

Step 1: Set d_t, the coefficient of variable x_t in the lifted inequality, to be the largest integer such that

$$\sum_{p(k)\in F(G')} f_{p(k)} + \sum_{ij\in Y(G')} \bar{y}_{ij} + \sum_{r=1}^{t} d_r x_r \leq IND(G') \qquad (5.6)$$

is a valid inequality for P_I.

Step 2: If t = T, Stop. Otherwise, increment t by 1 and return to Step 1.

Implementing Step 1:

In the t^{th} iteration of the Sequential lifting procedure, the (t-1) coefficients $d_1,\ldots,d_{t-1}$, corresponding to the variables $x_1,\ldots,x_{t-1}$,

have already been determined, and the 'partially lifted' inequality is similar to (5.6) (with the index of summation going from r = 1 to (t-1) instead of 1 to t in the third term of the left-hand side). The t^{th} coefficient d_t is the largest possible integer for which the inequality (5.6) is valid for the Set packing problem [SP]. This coefficient can be determined by solving the follcwing Weighted Vertex packing problem $[VP_t]$.

> The graph G_t for the Vertex packing problem $[VP_t]$ is the subgraph of G_I induced by the set of nodes $V_t = F(G') \cup Y(G') \cup \{u_r : r=1,\ldots,t\}$. The current vertex u_t is assigned the weight 0, the nodes of G' are all assigned a weight of 1, and each of the vertices u_r, for $1 \leq r < t$ is assigned the weight d_r (the coefficients in (5.6) that have already been determined). Then, $[VP_t]$ is the problem of finding the maximum weight vertex packing of the subgraph induced by V_t, containing node u_t.

Let z_t be the optimal value of the problem $[VP_t]$. Using an inductive argument it is possible to show that z_t must be less than or equal to IND(G'). Then, in the t^{th} iteration of the Sequential lifting procedure, the largest possible value of the coefficient d_t that ensures the validity of (5.6) is determined as

$$d_t = IND(G') - z_t \ .$$

Note that, in general, starting with the same initial inequality, this sequential procedure might generate different lifted facets depending on the sequence in which the vertices of V'(G') are considered. In fact, all sequential lifting procedures possess this characteristic. Before examining the nature of the liftings derived from odd hole-induced inequalities using the Sequential lifting procedure, we discuss conditions under which some of the lifting coefficients d_t must necessarily be zero.

3.3.4 Conditions for zero lifting coefficients:

In any sequential or simultaneous lifting of a subgraph-induced inequality such as (5.3), it is possible to set the lifting coefficients d_t corresponding to some of the additional variables x_t to zero a priori. These variables, then, need not be considered in the lifting procedure. Lemma 4 gives a sufficient condition, proposed by Cornuejols and Thizy [1982], for establishing that any given coefficient is zero. We include a proof of this lemma for completeness.

Lemma 4: (Cornuejols and Thizy [1982])

Suppose the inequality (5.1) corresponding to a node-induced subgraph G' of the Intersection graph G_I is a facet of $P_I^{G'}$. If for any vertex $u_t \epsilon V'(G')$, the Independence number of the subgraph of G_I induced by $\{V(G') \cup u_t\}$ is greater than IND(G'), then the corresponding coefficient d_t must be zero in any lifting (5.5).

Proof:

Let G_t'' be the subgraph of G_I induced by the set of vertices $\{V(G') \cup u_t\}$, for any vertex u_t in the set V'(G'). The Independence number of G_t'' must be less than or equal to (IND(G') + 1), since G_t'' contains exactly one vertex more than G'; further, if $IND(G_t'') = IND(G') + 1$, then every maximum independent set of G_t'' must contain the added vertex u_t. Consider any vertex u_t for which $IND(G_t'')$ is greater than IND(G'), and let R_t be any maximum independent set of G_t''. Clearly, R_t is also independent in G_I and must, therefore, satisfy inequality (5.5). However, the left-hand side of inequality (5.5)

evaluated for the feasible solution R_t is $(IND(G') + d_t)$. Hence, d_t must be zero for (5.5) to be a valid inequality for [SP].

Notice that this proof did not depend on the method used for lifting the inequality (5.1); therefore, the result holds for both sequential and simultaneous liftings of (5.1). We can also express the condition of Lemma 4 in the following alternate form.

Corollary 4.1:

Let G' be any node-induced subgraph of G_I whose corresponding inequality (5.1) is a facet for the lower dimensional polytope $P_I^{G'}$. Let R' be any maximum independent set of G'. Then, if any vertex $u_t \epsilon V'(G')$ is not adjacent in G_I to any vertex of R', the corresponding coefficient d_t in the lifted inequality (5.6) must be zero.

This corollary uses the fact that the independence number of G_t'', for any vertex $u_t \epsilon V'(G')$, is equal to $(IND(G') + 1)$ if and only if there is a maximum independent set R' of G' containing only vertices that are not adjacent in G_I to vertex u_t.

We will use Lemma 4 and Corollary 4.1 to fix at zero several lifting coefficients in subgraph-induced facets for the Set packing problem [SP]. Next, we examine the structure of the lifted inequalities derived by the Sequential lifting procedure when it is applied to inequalities (5.3) that are induced by odd holes of the Intersection graph G_I.

3.3.5 Sequential Liftings of the odd hole inequalities (5.3):

Consider any odd hole H of G_I that is induced by the set of vertices $V(H) = F(H) \cup Y(H)$. As before, we assume that V(H) contains f-nodes corresponding to q different commodities indexed as $k_1,\ldots,k_q$ (i.e., $K' = \{k_1,\ldots,k_q\}$, where K' is the set of commodities with at least one corresponding f-node in F(H)), and that for the first r of these commodities exactly two corresponding f-nodes belong to V(H) (i.e., $K'' = \{k_1,\ldots,k_r\}$). Because of the special structure of odd holes, many lifting coefficients in (5.4) can be fixed at zero a priori using Lemma 4 and Corollary 4.1. For instance, given any vertex u belonging to the set V(H), it is always possible to find a maximum independent set R' of H that does not include vertex u; consequently, any vertex of V'(H) that is adjacent in G_I to only vertex u of V(H) is not adjacent to any of the vertices of R'. Corollary 4.1, therefore, implies that for every vertex $u_t \epsilon V'(H)$ that is adjacent to exactly one vertex of V(H) in G_I, the coefficient d_t must be zero in the lifted facet (5.4). This property is not necessarily true for general node-induced subgraphs G' of G_I. For odd holes of G_I, however, the following stronger result applies.

Lemma 5:

For any given odd hole H of G_I, if any vertex $u_t \epsilon V'(H)$ is adjacent in G_I to at most two nodes of H, then d_t must be zero in every lifted inequality (5.4) that is derived from the odd hole inequality (5.3) induced by H.

Proof:

If u_t is not adjacent to any node of H, then R' U $\{u_t\}$ is an independent set of G_t'' with cardinality (IND(H) + 1), where R' is any maximum independent set of H. Similarly, if u_t is adjacent to exactly one node of H, it is possible to identify at least one maximum independent set of G_t'' with cardinality greater than IND(H) (as discussed before the statement of this lemma).

Finally, let us consider the case when u_t is adjacent to exactly two nodes, say nodes a and b, of H. We will show that G_t'' has a maximum independent set of cardinality (IND(H) + 1) that does not contain either a or b. Let V_1 be the subset of nodes of H (excluding nodes a and b) that are encountered while traversing from a to b in the clockwise direction along the arcs of H. Similarly, let V_2 be the subset of nodes encountered in the counterclockwise direction. Thus

$$V(H) = V_1 \cup V_2 \cup \{a,b\},\ V_1 \cap V_2 = \emptyset \text{ , and}$$

$$|V_1| + |V_2| = (|V(H)|+2) \text{ is odd.}$$

(Note that when the nodes a and b are adjacent, either V_1 or V_2 would be empty.) Since $|V_1| + |V_2|$ is odd, either $|V_1|$ or $|V_2|$ must be odd. Suppose $|V_1|$ is odd. Then, the subgraph of G_I induced by V_1 is a simple path with an odd number of vertices and edges. Hence, there is a unique maximum independent set, say R_1, of this subgraph whose cardinality is $(|V_1|+1)/2$. The subgraph induced by V_2 is also a simple path, but with even an number of vertices and edges. Let R_2 be a maximum independent set, with cardinality $|V_2|/2$, of this subgraph. Then, $R_1 \cup R_2 \cup \{u_t\}$ is a vertex packing of G_t'' with cardinality

$$|R_1| + |R_2| + 1 = (|V_1|+1)/2 + |V_2|/2 + 1$$

$$= (V(H)-1)/2 + 1 \quad \text{since } |V(H)| = |V_1|+|V_2|+2$$

$$= IND(H) + 1.$$

Thus, we have shown that, if the vertex $u_t \epsilon V'(H)$ is adjacent in G_I to at most 2 nodes of H, there is a vertex packing of size (IND(H) + 1) in the subgraph G''_t. Therefore, by Lemma 4, the corresponding coefficient d_t must be zero in any lifting of the odd hole inequality (5.3).

Lemma 5 implies that, while lifting the odd hole inequalities (using the sequential or any other lifting procedure), we need only consider those additional variables whose corresponding vertices of G_I are adjacent to at least three vertices of the given odd hole H. In particular,

(1) for all f-nodes $[p(k)] \notin F(H)$, the corresponding lifting coefficient $a_{p(k)}$ must be zero if

   (a) $k \notin K'$ (i.e., no f-node corresponding to commodity k belongs to the given odd hole H) and at most 2 arcs of Y(H) belong to the path p(k) in the original graph,

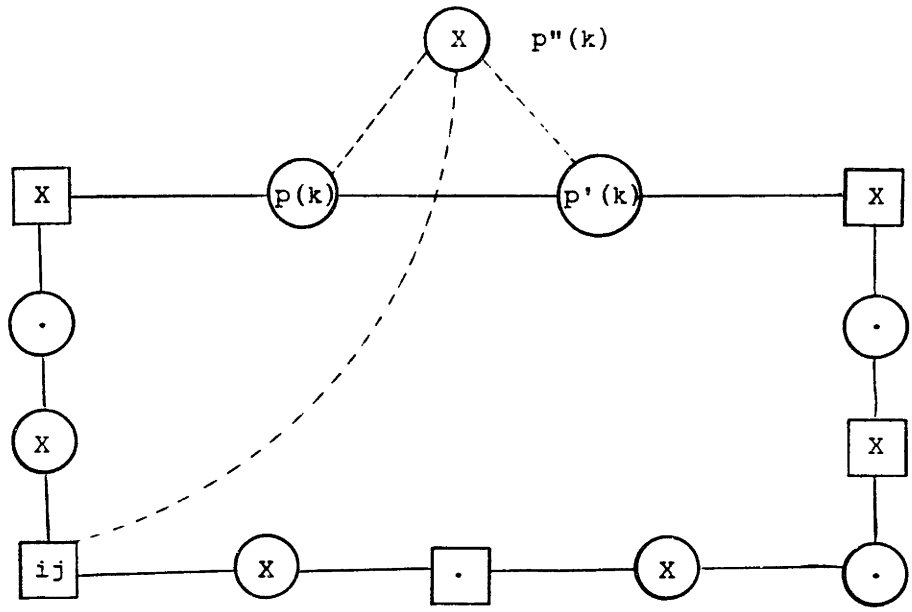
   (b) $k \epsilon K' \backslash K''$ (i.e., exactly one f-node corresponding to commodity k belongs to H) and at most one arc of Y(H) belongs to path p(k) in the original graph, or

   (c) $k \epsilon K''$ (i.e., two f-nodes corresponding to commodity k belong to H) and no arc of Y(H) belongs to path p(k), and

(2) for all vertices $[ij] \notin Y(H)$, if arc (i,j) is contained in at most two paths of F(H), then the corresponding lifting coefficient b_{ij} must be zero.

Besides these cases, there could be other instances in which the lifting coefficients $a_{p(k)}$ or b_{ij} must necessarily be zero. For instance, consider the example in Figure 6. The f-node [p(k)] in the dotted circle in the figure is not in the odd hole H and is adjacent to 3 nodes of H. Yet, its lifting coefficient $a_{p(k)}$ must be zero since the vertices marked with 'x' constitute an independent set with cardinality (IND(H)+1) (with IND(H)=6 in

Figure 6: Lifting Odd Hole Inequalities



this example) of the subgraph induced by V(H) U {[p(k)]}.

In general, we can use Corollary 4.1 to systematically identify all such variables with zero lifting coefficients in the following manner. Each odd hole H has exactly |V(H)| different maximum independent sets, and these can be easily enumerated. Suppose, for each maximum independent set R', we identify all the vertices in V'(H) that are not adjascent to any vertex of R'. Then, by Corollary 4.1, the lifting coefficient corresponding to each of these vertices must be zero. Next, we will show that when the sequential lifting procedure is applied to odd hole-induced inequalities, the lifting coefficients must satisfy certain upper bounds.

Suppose the subset of vertices $V(H) = F(H) \cup Y(H)$ induces an odd hole H of G_I. Let $Y^+ \subseteq Y \backslash Y(H)$ be the subset of y-nodes not contained in H and for which the lifting coefficients can be positive (i.e., the y-nodes whose coefficients are not fixed at zero using the procedures just described). Similarly, let $F^+ \subseteq F \backslash F(H)$ be the subset of f-nodes not in H that can have positive coefficients in the lifted inequality (5.4). We let V^+ denote the set $Y^+ \cup F^+$, and let the vertices in V^+ be arranged in some arbitrary order before the sequential lifting procedure described in Section 3.3.3 is applied. We briefly recapitulate this procedure. At the t^{th} stage of the algorithm, the lifting coefficient d_t corresponding to the vertex u_t is calculated by solving the weighted vertex packing subproblem $[VP_t]$. This subproblem involves finding the maximum weight vertex packing that contains the 'current' vertex u_t, of the subgraph G_t induced by the set of nodes $V(H) \cup \{u_r : r=1,\ldots,t-1\} \cup u_t$. The vertex u_t is assigned a weight of zero, while the coefficients of the current partially lifted inequality serve as the weights for the other vertices of this subgraph. (In particular, vertices of H are assigned unit weights.) If z_t is the optimal value of this subproblem, the t^{th} lifting coefficient d_t is determined as $d_t = IND(H) - z_t$. The following proposition gives upper bounds on the values of the lifting coefficients that are obtained when this sequential procedure is applied to the odd hole induced inequalities (5.3). These upper bounds are independent of the order in which the vertices of V^+ are considered.

PROPOSITION 5:

When the inequality (5.3) induced by an odd hole H is lifted using the Sequential procedure, the lifting coefficients $a_{p(k)}$ and b_{ij} must

satisfy the following bounds:

(a) for all y-nodes $[ij] \in Y^+$, $b_{ij} \leq (|K''|-1)/2$, and

(b) for all f-nodes $[p(k)] \in F^+$,

if $k \notin K'$, $a_{p(k)} \leq (|K''|-1)/2$, and

if $k \in K'$, $a_{p(k)} \leq (|K''|-1)/2 + 1$,

where K" is the set of all commodities with exactly 2 corresponding f-nodes in H.

Proof:

(a) Let the y-node $[ij] \in Y^+$ be the vertex that is considered at the t^{th} stage of the Sequential lifting procedure and let G_{ij} denote the subgraph of G_I induced by the set of nodes V(H) U {[ij]}. Notice that G_{ij} is always a node-induced subgraph of the graph over which the weighted vertex packing subproblem $[VP_t]$ (in the sequential lifting procedure) is defined, regardless of the position t of the vertex [ij] in the sequence V^+. Hence, any vertex packing of G_{ij} containing node [ij] must be a feasible solution for $[VP_t]$; Y(H) U {[ij]} is one such vertex packing. (Recall that Y(H) is an independent set of H and that [ij] is not adjacent to any vertex of Y(H).) The weight of this packing is $|Y(H)|$, since all the vertices of V(H) are assigned a weight of 1, while node [ij] has zero weight. Therefore, the optimal value z_t of the t^{th} subproblem $[VP_t]$ must be greater than or equal to $|Y(H)|$, implying that

$$b_{ij} = IND(H) - z_t = |Y(H)| + (|K''|-1)/2 - z_t$$
$$\leq (|K''|-1)/2 \qquad \text{since } z_t \geq |Y(H)| .$$

(b) Let us now consider f-nodes [p(k)] belonging to the set F^+. Suppose no f-node corresponding to commodity k belongs to H, i.e., $k \notin K'$. Then,

the set of nodes $\{[p(k_r) : r=1,\ldots,q\} \cup \{[p(k)]\}$ is a vertex packing of the subgraph $G_{p(k)}$ induced by $V(H) \cup \{[p(k)]\}$. Furthermore, if $[p(k)]$ is the t^{th} vertex in the sequence V^+, then this vertex packing is a feasible solution for the t^{th} subproblem $[VP_t]$, implying that $z_t \geq |Y(H)|$. Thus, as shown before, the lifting coefficient $a_{p(k)}$ must be less than or equal to $(|K"|-1)/2$. On the other hand, if $k \epsilon K'$ (i.e., at least one f-node corresponding to commodity k belongs to H), then the set of vertices $\{[p(k_r)], r=1,\ldots,q, k_r \neq k\} \cup \{[p(k)]\}$ is a feasible solution of the subproblem $[VP_t]$, with an objective function value $z_t = |Y(H)| - 1$; hence, $a_{p(k)} \leq (|K"|-1)/2 + 1$.

Let us briefly examine the implication of Proposition 5. Recall that, for any odd hole H,

$$IND(H) = (|V(H)|-1)/2 = |Y(H)| + (|K"|-1)/2, \text{ and}$$

$$|Y(H)| = |K'| \geq |K"|.$$

Therefore,

$$IND(H) \geq 3(|K"|-1)/2 .$$

We had asserted earlier that for any subgraph induced inequality (5.1), the lifting coefficients of the variables $f_{p(k)}$ and $\bar{y}_{ij}$ can never exceed the right-hand side value b_0, which is IND(H) for odd holes. In Proposition 5, we have reduced this upper bound substantially (by almost $2/3^{rd}$) for sequentially lifted facets of the Set packing polytope derived from odd hole inequalities.

Next, we consider two special types of odd holes in the Intersection graph G_I. For these two subgraphs, we prove that all the lifting

coefficients are either 0 or 1; further, for the second class of subgraphs, we show that the sequentially lifted facet is unique regardless of the order in which the vertices u_t are considered in the sequential lifting procedure. Finally, we illustrate how odd holes of the first type are identified by the first constraint generation procedure discussed in Chapter 2; we also relate the second type of odd hole to the subgraphs with cyclic incidence matrices described in Chapter 2.

3.3.6 Special Case 1: 1-holes of G_I:

Consider the special type of odd hole H whose vertex set V(H) contains only one pair of f-nodes that correspond to the same commodity, say k_1 (i.e., $K'' = \{k_1\}$); all the other f-nodes in V(H) correspond to distinct commodities $k_2,\ldots,k_q$. We refer to such odd holes as 1-HOLES, and denote a 1-hole with $|K'| = |Y(H)| = q$ as H^{1q}. As before, we assume for notational convenience that the vertices of G_I are labeled such that

$$F(H) = \{[p(k_1)], [p'(k_1)], [p(k_2)], \ldots, [p(k_p)]\},$$

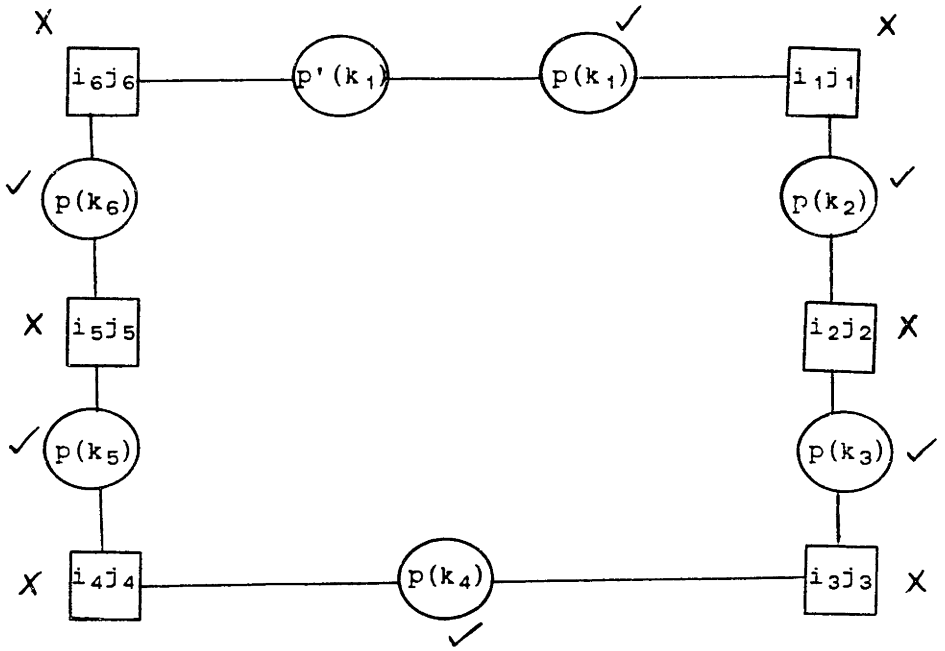
$$Y(H) = \{[i_1,j_1], \ldots, [i_p,j_p]\}, \text{ and}$$

$[i_t,j_t]$, for $1 \leq t < q$, is adjacent to $[p(k_t)]$ and $[p(k_{t+1})]$, and,

$[i_p,j_p]$ is adjacent to $[p(k_q)]$ and $[p'(k_1)]$ in G_I .

Figures 7 illustrates the structure of the 1-hole H^{16}; H^{13} is shown in Figure 4a. Note that with $|K'| = |Y(H)| = q$, the number of nodes in H^{1q} is $(2q + 1)$; hence, the independence number $IND(H^{1q})$ of H^{1q} is q. Before examining the special form of the lifted facets derived from 1-hole inequalities, we will describe how 1-holes can be identified using the Auxiliary graph G_a and the procedure described in Section 2.2

Figure 7: Example of a 1-hole



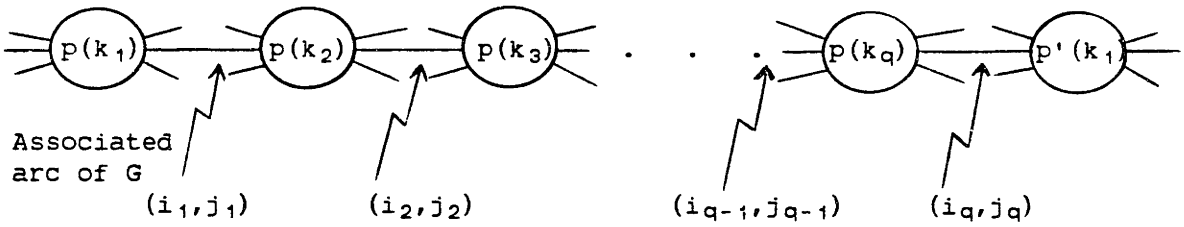
For the 1-hole H^{16} : q = 6, K" = $\{k_1\}$, K' = $\{k_t, t=1,2,\ldots,6\}$
Nodes marked 'x' and '✓', respectively, designate two maximum independent sets of H^{16}.

Identifying 1-holes using the Auxiliary graph G_a:

Let us briefly review the first cut-generation procedure described in Section 2.2. For any given solution (f,y) of the Path-flow formulation [UNDPF], the Auxiliary graph G_a contains vertices [p(k)] corresponding to all paths p(k) that carry positive flow. G_a has an edge connecting two vertices [p(k)] and [p'(k')] if and only if the two corresponding paths p(k) and p'(k') have at least one arc (i,j) of the original graph G in common that is TIGHT for both these paths; this arc (i,j) is said to be

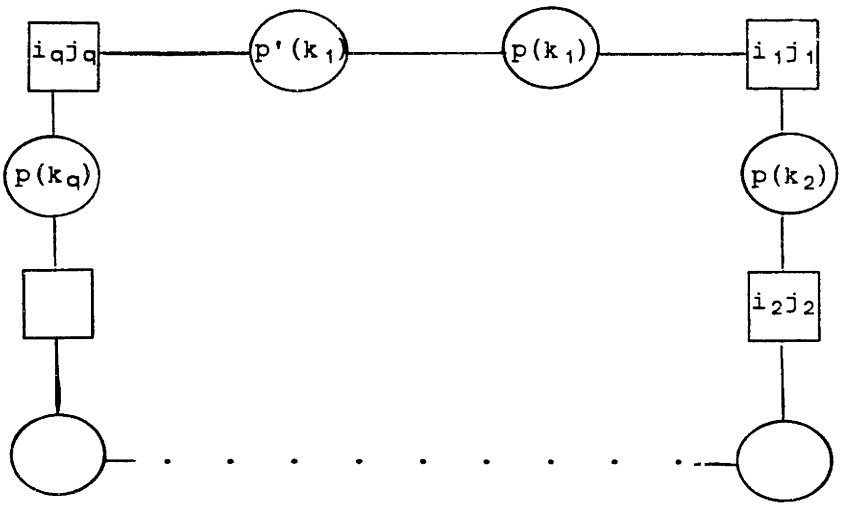
associated with the edge $([p(k)],[p'(k')])$ of G_a. Suppose some component S_m of G_a contains two nodes, say $[p(k_1)]$ and $[p'(k_1)]$ that correspond to the same commodity k. For generating a valid, violated inequality of [UNDPF], we trace a path with the least number of intermediate edges from the vertex $[p(k_1)]$ to vertex $[p'(k_1)]$ in S_m. Suppose, the intermediate nodes on this path are $[p(k_t)]$ for $1<t\leq q$; without loss of generality, we assume that all the commodities k_t for $1\leq t\leq q$ are distinct. Let $(i_t,j[t])$, for $1\leq t\leq q$, be the arcs of the original graph that are associated with the edges of the path from $[p(k_1)]$ to $[p'(k_1)]$ that we traced in S_m. Since this path is a minimum hop path, the associated arcs (i_t,j_t), for $1\leq t\leq q$, must all be distinct. Figure 8 illustrates the structure of a typical path from vertex $[p(k_1)]$ to vertex $[p'(k_1)]$ in G_a.

Figure 8: Path from $p(k_1)$ to $p'(k_1)$ in the Auxiliary Graph G_a



Letting $F(H) = \{[p(k_1)],p'(k_1)\} \cup \{[p(k_t)] : 1<t\leq q\}$, and $Y(H) = \{[i_t,j_t] : 1\leq t\leq q\}$, we see that the subgraph of the Intersection graph G_I induced by the subset of nodes $V(H) = F(H) \cup Y(H)$ is indeed a 1-hole of G_I. Figure 9 shows the 1-hole corresponding to the "path" of Figure 8. Thus, it is clear that by tracing the minimum hop path from

Figure 9: Odd hole corresponding to path in G_a



vertex $[p(k_1)]$ to $[p'(k_1)]$ in the Auxiliary graph, we are in effect identifying a 1-hole of the Intersection graph G_I. Indeed, the valid, violated inequality (3.3) presented in Proposition 3 is identical to the inequality (5.1) induced by the corresponding 1-hole of G_I. We can demonstrate in a similar manner that the general cut-generation procedure of Appendix A is essentially a method for identifying general odd holes in the Intersection graph.

Sequential liftings of 1-hole inequalities:

Having discussed the motivation for considering 1-holes and a procedure for identifying them, we now examine the special features of sequentially lifted facets derived from 1-hole induced inequalities. It is

instructive for this purpose to examine the structures of some of the maximum independent sets in any given 1-hole H^{1q}. There are three maximum independent sets of H^{1q} that contain either y-nodes alone or f-nodes alone. These are (refer Figure 7)

(1) $Y(H^{1q})$, the set of all y-nodes in $V(H^{1q})$,

(2) $F(H^{1q})\setminus\{[p'(k_1)]\}$, the set of all f-nodes in $V(H^{1q})$, excluding the vertex $[p'(k_1)]$, and

(3) $F(H^{1q})\setminus\{[p(k_1)]\}$, the set of all f-nodes in $V(H^{1q})$, excluding the vertex $[p(k_1)]$.

By Corollary 4.1, any vertex not in H^{1q} that is <u>not</u> adjacent to any of the vertices belonging to one of these independent sets must have a lifting coefficient equal to zero (in any sequential or simultaneous lifting of the 1-hole-induced inequality (5.3)). In particular,

(a) for all y-nodes [ij] that do not belong to $Y(H^{1q})$, the lifting coefficient b_{ij} (in the lifted facet (5.4)) must be zero, since [ij] is not adjacent, in G_I, to any other y-node and, hence, to any node of the maximum independent set $Y(H^{1q})$. (Alternatively, recall from Proposition 5, that the coefficient b_{ij} must be less than or equal to $(|K''| - 1)/2$ in any sequentially lifted facet; since, $|K''| = 1$ for 1-holes, b_{ij} must be zero. However, we cannot establish this property for simultaneously lifted facets using Proposition 5.), and

(b) for all commodities k, none of whose f-nodes belong to H^{1q} (i.e., for all $k \notin K'$), the corresponding lifting coefficients $a_{p(k)}$, for every path $p(k) \in P(k)$, must be zero; again, this result follows from the fact that none of these f-nodes is adjacent in G_I to any of the vertices belonging to the maximum independent set $F(H^{1q})\setminus\{[p'(k_1)]\}$ of the 1-hole H^{1q}.

These two properties imply that we need to evaluate only the lifting coefficients $a_{p(k)}$ for commodities $k \in K'$ and $[p(k)] \notin F(H^{1q})$; all other coefficients corresponding to vertices not in $V(H^{1q})$ must be zero. From our earlier discussions, we know that some of the lifting coefficients even in this restricted subset of indices can be fixed to zero. For example, if

none of the y-nodes $[ij] \epsilon Y(H^{1q})$ are adjacent in G_I to some f-node $[p(k)] \notin F(H^{1q})$ with $k \epsilon K'$, then $a_{p(k)}$ must be zero. For the special case of 1-holes, the following stronger result holds.

For $2 \leq t \leq q$, define the subsets Y_t^- and Y_t^+ of y-nodes in the 1-hole H^{1q} as follows:

$$Y_t^- = \{[i_1,j_1],\ldots,[i_{t-1},j_{t-1}]\} \text{ and}$$

$$Y_t^+ = \{[i_t,j_t],\ldots,[i_q,j_q]\}.$$

Lemma 6:

Let H^{1q} be a 1-hole in the Intersection graph G_I and consider any f-node $[p''([k_t)]$ not in H^{1q} that corresponds to some commodity k_t with $2 \leq t \leq q$. If this node is not adjacent to any vertex of either the set Y_t^- or the set Y_t^+, then the corresponding lifting coefficient $a_{p''(kt)}$ must be zero in any lifting of the 1-hole induced inequality (5.3). (Note: This adjacency condition holds if and only if path $p''(k_t)$ does not contain any of the arcs whose corresponding y-nodes belong either to Y_t^- or to Y_t^+).

Proof:

Suppose $[p''(k_t)]$ is not adjacent to any vertex of Y_t^-. Then, the subset of vertices

$$Y_t^- \cup \{[p''(k_t)\} \cup \{[p(k_{t+1})],\ldots,[p(k_q)]\}$$

is a maximum independent set of the subgraph induced by $V(H^{1q}) \cup \{[p''(k_t)]\}$. The cardinality of this set is $(q + 1)$ $(= IND(H^{1q} + 1)$; hence, by Lemma 4, $a_{p''(kt)}$ must be zero. Similarly, we can show that, if $[p''(k_t)]$ is not adjacent to any node of Y_t^+, the independence number of $V(H^{1q}) \cup \{[p''(k_t)]\}$ is $(q + 1)$ and the

corresponding lifting coefficient is zero.

Thus, the only remaining f-variables that can have positive coefficients in any lifting (5.4) of the inequality (5.3) induced by a 1-hole H^{1q} are those corresponding to

(1) f-nodes $[p''(k_1)]$ for commodity k_1, such that the path $p''(k_1)$ contains at least one of the arcs in $Y(H)$, and

(2) f-nodes $[p''(k_t)]$ for $2 \leq t \leq q$, such that the path $p''(k_t)$ contains at least one arc belonging to each of the sets Y_t^- and Y_t^+.

Let F_1 and F_2 denote these two vertex sets. In the following proposition, we show that in any sequential lifting (5.4) of the 1-hole inequality, the coefficients $a_{p(k)}$ for both these types of f-nodes must be either 0 or 1.

PROPOSITION 6:

Every sequentially lifted facet of P_I, that is derived from the inequality (5.3) induced by a 1-hole H^{1q}, is of the form:

$$\sum_{p(k) \in F(H)} f_{p(k)} + \sum_{ij \in Y(H)} \bar{y}_{ij} + \sum_{p(k) \in F''} f_{p(k)} \leq q, \qquad (5.7)$$

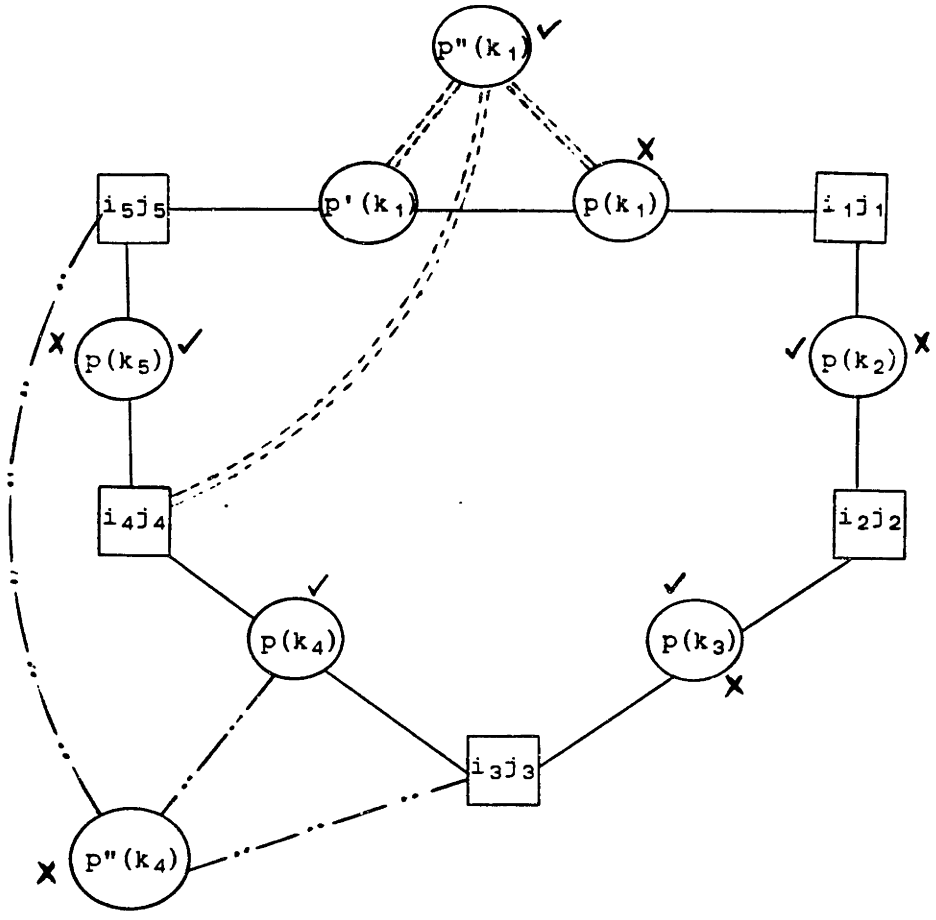
where F'' is a subset of $F_1 \cup F_2$, and the vertex sets F_1 and F_2 are as defined before.

Proof:

We have already shown that all the lifting coefficients except those corresponding to f-nodes belonging to the two sets F_1 and F_2 must be zero in any lifting of a 1-hole inequality. Therefore, we only have to show that the lifting coefficients corresponding to f-nodes belonging to

$F_1 \cup F_2$ are less than or equal to 1 in any sequential lifting of the 1-hole induced inequality (5.3). (Recall that all the lifting coefficients derived by the Sequential lifting procedure are integral; hence, $a_{p(k)} \leq 1$ implies that $a_{p(k)}$ is either 0 or 1). To prove this fact, we will demonstrate that the Vertex packing subproblem $[VP_t]$ of the Sequential lifting procedure has an optimal value z_t greater than or equal to $(q - 1)$ at every stage; then, the t^{th} lifting coefficient, which is determined as $(q - z_t)$, must be less than or equal to 1. Consider first any f-node $[p''(k_1)]$ belonging to the set F_1 and let $G_{p''(k1)}$ be the subgraph of G_I induced by the set of vertices $V(H^{1q}) \cup \{[p''(k_1)]\}$. The set of f-nodes $\{[p(k_t)] : t = 2,\ldots,q\} \cup \{[p''(k_1)]\}$ is an independent set (containing the node $[p''(k_1)]$) of cardinality q in this subgraph. (Figure 10 (Case 1) illustrates the structure of this independent set.) If the f-node $[p''(k_1)]$ is considered at the t^{th} stage of the sequential procedure, then $G_{p''(k1)}$ is a node-induced subgraph of the graph over which the Vertex packing problem $[VP_t]$ is defined. Therefore, the independent set of $G_{p''(k1)}$ just identified must be a feasible solution of $[VP_t]$, and its weight is $(q - 1)$. (In $[VP_t]$, all the vertices of H^{1q} are assigned unit weights, while the vertex that is being currently considered has zero weight.) Hence, z_t must be greater than or equal to (q-1), implying that the lifting coefficient $a_{p''(k1)}$ $(= z_t - q)$ must be less than or equal to 1. Similarly, for the f-node $[p''(k_{t'})]$ belonging to the set F_2, the set of f-nodes $\{[p(k_t)] : t = 1,2,\ldots,q,\ t \neq t'\} \cup \{[p''(k_{t'})]\}$ is an independent set with cardinality q of the subgraph $G_{p''(kt')}$ induced by $V(H^{1q}) \cup \{[p''(k_{t'})]\}$. (See Figure 10, Case 2). Therefore, the corresponding lifting coefficient $a_{p''(kt')}$ must be either 0 or 1 in

Figure 10: Illustration of maximum independent sets in 1-holes



Case 1: Path $p''(k_1)$ for commodity k_1 contains some arc $(i,j) \in A(H)$.
Nodes marked with '✓' are members of a maximum independent set.

Case 2: Path $p''(k_t)$, for some $t > 1$, contains at least one arc of Y_t^+ and one arc of Y_t^-.
Nodes marked with 'x' are members of a maximum independent set.

any sequential lifting of the 1-hole inequality. (Note : It is easy to see that for all vertices $[p''(k_t)] \epsilon F_1 \cup F_2$, for $1 \leq t \leq q$, the subgraph $G_{p''(kt)}$ of G_I induced by the vertex subset $V(H^{1q}) \cup \{[p''(k_t)\}$ does not contain any independent set of cardinality $(q + 1)$. Therefore, the coefficient $a_{p''(kt)}$ cannot be fixed at zero a priori (using Lemma 4).)

This result concludes our discussion of facets induced by 1-holes of the Intersection graph. We have shown that 1-holes can be identified using the Auxiliary graph G_a and the procedure of Section 2.2. In addition, we have proved that all the lifting coefficients in any sequentially lifted facet derived from 1-hole inequalities are zero or one; the variables that can have a lifting coefficient of 1 are easily identified using Proposition 6. We next examine another special type of odd hole.

3.3.7 Special Case 2: 2-holes of G_I:

Consider the special type of odd hole in which $K'' = K'$, i.e., all commodities $k \epsilon K'$ have exactly two corresponding f-nodes in the set $F(H)$. We will refer to such odd holes as 2-HOLES and denote them as H^{2q}, where $|K'| = |K''| = |Y(H^{2q})| = q$. Thus, in this case

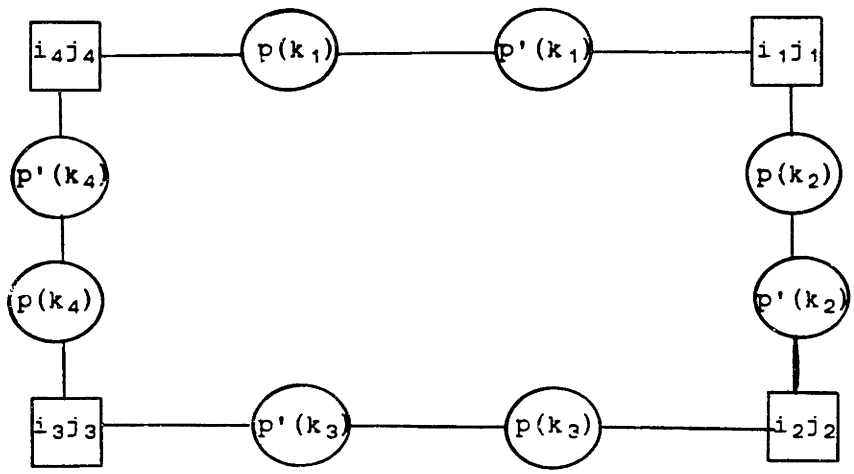
$$K' = K'' = \{k_t : 1 \leq t \leq q\},$$

$$F(H^{2q}) = \{[p(k_t)], [p'(k_t)] : 1 \leq t \leq q\}, \text{ and}$$

$$Y(H^{2q}) = \{[i_t, j_t] : 1 \leq t \leq q\}.$$

As before we assume that the y-node $[i_t, j_t]$ is adjacent in H^{2q} to the two f-nodes $[p'(k_t)]$ and $[p(k_{t+1})]$, for $1 \leq t \leq q$. Figure 11 is an example of a 2-hole with $q = 4$. Notice that the number of nodes in $H^{2q} = 3q$; hence,

Figure 11: Example of a 2-hole



$IND(H^{2q}) = (3q - 1)/2$. Also, every third node of the 2-hole is y-node.

Let us first describe a problem instance in which the Intersection graph contains a 2-hole. Consider the Network Design problem which has q commodities k_t, for $1 \leq t \leq q \leq |K|$, such that

- each of these q commodities has the same origin, designated as node 0,
- their destinations, denoted as D_t, for $1 \leq t \leq q$, are distinct, and
- the given network contains q distinct 'intermediate' nodes, denoted as nodes 1,2,...q, that are adjacent to the source node 0; besides, for every commodity k_t, for $1 \leq t \leq q$, the destination D_t is adjacent to the two intermediate nodes t and t+1 (where the sum t+1 is taken modulo q). Thus, the given network contains the two paths $p(k_t) = 0\text{-}t\text{-}D_t$ and $p'(k_t) = 0\text{-}t\text{+}1\text{-}D_t$ for each commodity k_t.

  (Note: Our subsequent discussions are also applicable to the case in which some or all of the intermediate nodes also serve as destinations for the q commodities. For convenience, however, we assume that the node subsets $\{1,2,\ldots,q\}$ and $\{D_1,\ldots,D_q\}$ are mutually exclusive.)

We will refer to this structure as a STAR configuration. Figure 12a is an example of such a network. We will be interested in Network Design instances in which this configuration arises as a subnetwork of a larger problem. The network over which the given problem is defined may contain other nodes, arcs, and commodities besides those corresponding to the Star configuration, and each of the q commodities $k_1,\ldots,k_q$ may have many more alternative routes from node 0 to their respective destinations. For instance, a Network Design problem that is defined over a complete graph with "complete" demand (i.e., demand from every node to every other node) contains many Star subnetworks.

Because of the special structure of the Star network, the subgraph of the Intersection graph G_I induced by the set of nodes

$$V(H^{2q}) = F(H^{2q}) \cup Y(H^{2q}),$$

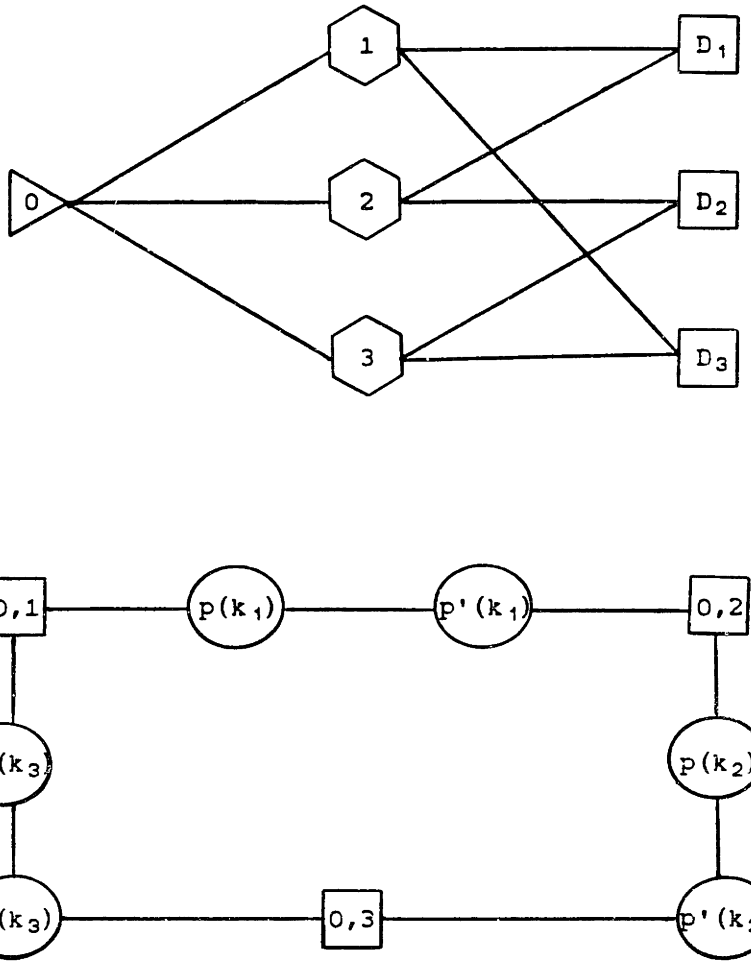
where $F(H^{2q}) = \{[p(k_t)], [p'(k_t] : 1 \leq t \leq q\}$, and

$$Y(H^{2q}) = \{[0,t], 1 \leq t \leq q\},$$

is a 2-hole. Figure 12b depicts the 2-hole of G_I corresponding to the Star configuration shown in Figure 12a. In general, the subgraph of G_I corresponding to the arcs and paths in a Star network bears a strong resemblance to the Intersection graph for the Plant location problem. In both cases, each f-node is adjacent to exactly one y-node (unlike general subgraphs of G_I, where each f-node may be adjacent to more than one y-node). We will exploit this fact later when we extend to the Network Design problem two classes of facial inequalities discussed by Cornuejols and Thizy [1982].

So far, we have described the structure of 2-holes in the Intersection

Figure 12: Star configuration and associated 2-hole



graph G_I and discussed one class of problem instances for which G_I contains 2-holes. Let us now specialize the sequential facet lifting procedure to inequalities (5.3) induced by this special class of 2-holes in G_I. We will let S^{2q} denote the 2-hole derived from any Star configuration. In order to identify lifting coefficients that must necessarily be zero, we will use the following fact about independent sets in general 2-holes:

> Given any two <u>consecutive</u> vertices u' and u" in H^{2q}, it is possible to find a maximum vertex packing of H^{2q} that does not contain both vertices u' and u".
>
> (In fact, this maximum independent set is unique; it consists of every alternate vertex of H^{2q} starting with the vertex adjacent to u'. (Refer Figure 13)).

Thus, if any vertex u_t not in the 2-hole S^{2q} is adjacent to at most 2 consecutive vertices of S^{2q}, the Independence number of the subgraph induced by $V(S^{2q}) \cup \{u_t\}$ is $(IND(S^{2q} + 1)$. Therefore, by Lemma 4, the lifting coefficients for all such vertices must be zero. In particular,

(a) the lifting coefficient b_{ij} corresponding to every arc (i,j) that is not in the Star configuration must be zero, since the corresponding y-node [ij] is not adjacent to any node of S^{2q}. The only remaining y-nodes of the Intersection graph G_I are those corresponding to the arcs (t,D_t) and $(t+1,D_t)$, for $1 \leq t \leq q$,

(b) the lifting coefficients corresponding to all the arcs (t,D_t) and $(t+1,D_t)$, for $1 \leq t \leq q$, must be zero. Each of these arcs belongs to exactly one path of the Star configuration, namely the path $p(k_t) = 0\text{-}t\text{-}D_t$ and $p'(k_t) = 0\text{-}t\text{+}1\text{-}D_t$, respectively; hence, the corresponding y-nodes are adjacent to exactly one node of S^{2q}, implying that the lifting coefficients are zero, and

(c) if $k \neq k_1,\ldots,k_q$, the lifting coefficient corresponding to every path $p(k) \epsilon P(k)$ for this commodity must be zero. Since every path that we consider is simple, p(k) can contain at most one of the q arcs (0,t), $1 \leq t \leq q$. Hence, [p(k)] is adjacent in G_I to at most one vertex of S^{2q}; therefore, the lifting coefficient $a_{p(k)}$ must be zero,

(d) for all k_t, $1 \leq t \leq q$, if any path $p''(k_t)$ does not contain any of the arcs (0,t), $1 \leq t \leq q$, then the lifting coefficient $a_{p''(kt)}$ must be zero. Since $p''(k_t)$ does not contain any of the q arcs incident from node 0 in the Star configuration, the node $[p''(k_t)]$ is adjacent only to the two consecutive nodes $[p(k_t)]$ and $[p'(k_t)]$ of S^{2q}. Hence, its lifting coefficient $a_{p''(kt)}$ must be zero.

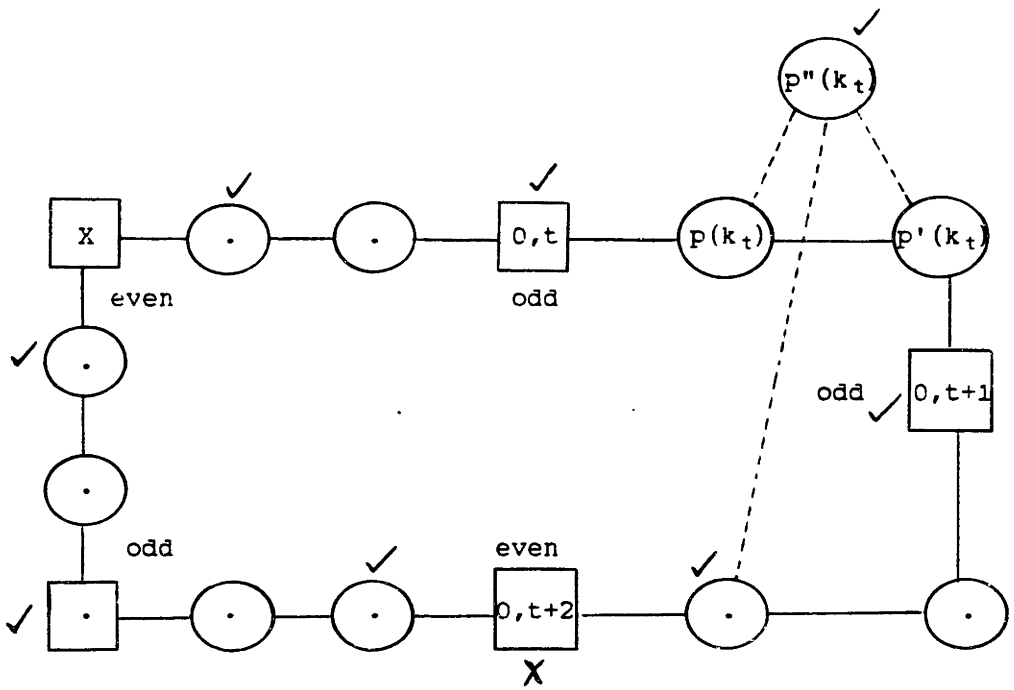
The only remaining lifting coefficients to be considered are those corresponding to paths $p''(k_t)$, for $1 \leq t \leq q$, that contain one of the arcs (0,t), $1 \leq t \leq q$. (Since we consider only simple paths, any path p(k) can contain at most one of these q arcs.). The f-nodes corresponding to these paths are adjacent to three vertices of S^{2q}. We will classify all such paths into two categories - Type A and Type B - and show that the lifting coefficients for Type A paths must be zero, while those for Type B paths must be 1 in any sequential lifting of the S^{2q}-induced inequality. For each index t, $1 \leq t \leq q$, we refer to the set of arcs $\{(0,t+2),(0,t+4),\ldots\ldots,(0,t+q-1)\}$ (with all sums taken modulo q) as the set of EVEN arcs E_t; correspondingly, $\{(0,t),(0,t+1),(0,t+3),\ldots.,(0,t+q-2)\}$ is the set of ODD arcs O_t. Then,

Type A paths are those paths $p''(k_t)$, for $1 \leq t \leq q$, that contain an Even arc belonging to the set E_t, and

Type B paths are those paths $p''(k_t)$, for $1 \leq t \leq q$, that contain an Odd arc belonging to the set O_t.

The f-node $[p''(k_t)]$ corresponding to any Type A path is adjacent to the three vertices - $[p(k_t)]$, $[p'(k_t)]$, and $[0,t]$ - where $(0,t)$ is an Even arc belonging to the set E_t.

Figure 13: Illustration for Type A nodes in 2-holes



Vertices belonging to E_t marked with x.
Vertices marked with '✓' constitute a maximum independent set.

As Figure 13 illustrates, the 2-hole S^{2q} contains a unique maximum vertex packing of cardinality $IND(S^{2q})$ that contains none of these three vertices. (This vertex packing consists of all alternate vertices starting with the y-node that is adjacent to $[p(k_t)]$.) Hence, the independence number of the subgraph induced by $V(S^{2q}) \cup \{[p''(k_t)\}$ is $(IND(S^{2q}) + 1)$, and by Lemma 4 the lifting coefficient $a_{p''(kt)}$ must be zero. We are thus left only with f variables corresponding to Type B paths. It is easy to see that, for all

Type B paths $p''(k_t)$, the subgraph induced by $V(S^{2q}) \cup \{[p''(k_t)\}$ does not contain any vertex packing of size $(IND(S^{2q}) + 1)$; therefore, the corresponding lifting coefficients $a_{p''(kt)}$ are not necessarily zero. In fact, as the following result shows, all these coefficients must be equal to 1 in any sequential lifting of the inequality (5.3) induced by the 2-hole S^{2q}.

Claim:

In any sequential lifting of the inequality (5.3) induced by S^{2q}, the coefficients in the lifted facet (5.4) for all Type B paths must be equal to 1.

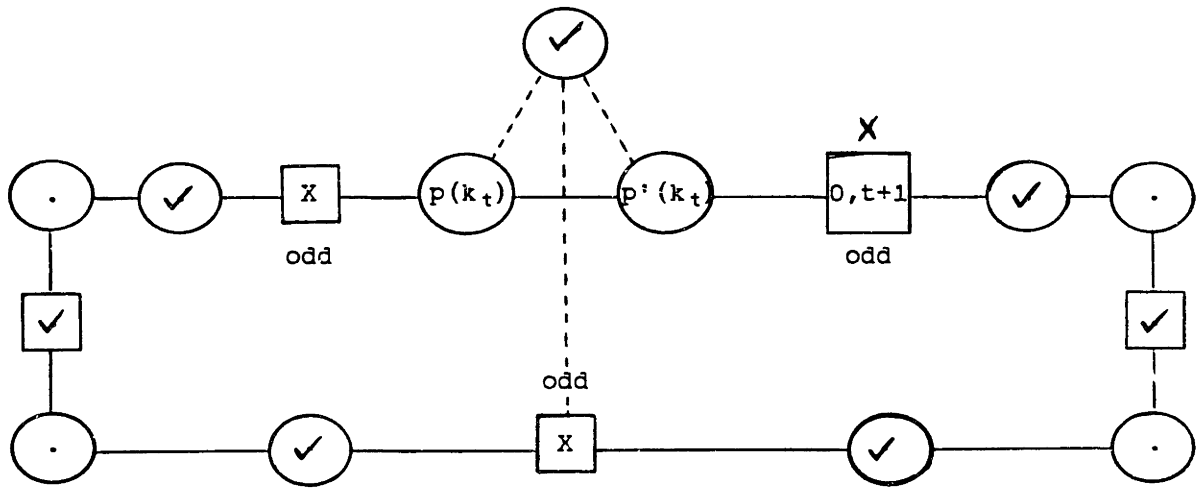
Proof:

Consider the subgraph of G_I induced by $V(S^{2q}) \cup \{[p''(k_t]\}$ for any Type B path $p''(k_t)$. In this subgraph, the set of vertices

$$\{[0,i],\ (0,i) \epsilon E_t\} \cup \{[p''(k_t)]\}$$

is contained in a maximum vertex packing that has cardinality $IND(S^{2q})$. And, this vertex packing is feasible in the Vertex packing subproblem $[VP_r]$ of the sequential lifting procedure, regardless of the step r at which the vertex $[p''(k_t)]$ is considered. (This is because the subgraph that we have considered is always a node-induced subgraph of the graph over which the Vertex packing problem $[VP_r]$ is defined.) Figure 14 gives an example of such an independent set. The 'objective function value' (for the subproblem $[VP_r]$) of this solution is $(IND(S^{2q} - 1)$. On the other hand, since we are adding only Type B nodes at each stage of the sequential procedure, it is possible to show, using an inductive argument, that $[VP_r]$ does not have a better vertex packing. Thus, the optimal value of $[VP_r]$ at each stage r of the sequential lifting

Figure 14: Illustration for Type B nodes in 2-holes



Vertices belonging to O_t marked with 'x'.
Vertices marked with '✓' constitute a vertex packing of size $IND(S^{2q})$

procedure (with each stage corresponding to the addition of one more Type B node) is $(IND(S^{2q} - 1)$. Therefore, the lifting coefficient for each Type B node must be 1.

We have thus been able to determine exactly all the lifting coefficients in sequentially lifted facets that are derived from Star configurations of the NDP. Notice that since $|K''| = q$ for 2-holes, the upper bound on the lifting coefficients provided by Proposition 6 is not very useful in this case. The following proposition summarizes the discussions of this section.

PROPOSITION 7:

Let S^{2q} be the 2-hole of the Intersection graph G_I corresponding to some Star configuration in the original graph. Then, the only sequentially lifted facet (5.4) of P_I that is derived from the inequality (5.3) induced by S^{2q} is

$$\sum_{t=1}^{q} (f_{p(kt)} + f_{p'(kt)}) + \sum_{p(k)\epsilon F_B} f_{p(k)} + \sum_{i=1}^{q} \bar{y}_{Oi} \leq (3q-1)/2 \qquad (5.8)$$

where F_B is the set of all Type B nodes corresponding to the given Star configuration.

A valid inequality for the Arc-flow formulation:

Inequality (5.8) in Proposition 7 is expressed in terms of Path-flow and design variables; since, it is valid for [SP], it must also be valid for the original strong disaggregated Path-flow formulation [UNDSPF]. We will now show how (5.8) can be translated into an equivalent valid inequality for the Arc-flow formulation as well. Recall that, for all commodities k_t, for $1 \leq t \leq q$, the set F_B includes all vertices (or path indices) $[p''(k_t)]$ such that the corresponding path $p''(k_t)$ contains an Odd arc arc $(0,i)$ belonging to the set O_t. For all $(0,i) \epsilon O_t$, let Q_{it} be the set of paths for commodity k_t that contain arc $(0,i)$ (including the two original paths $p(k_t) = 0\text{-}t\text{-}D_t$ and $p'(k_t) = 0\text{-}t\text{+}1\text{-}D_t$). Then,

$$\sum_{p''(kt)\epsilon Qit} f_{p''(kt)} = x_{Oi}^{kt} \qquad \text{for all } (0,i)\epsilon O_t \text{ and } 1 \leq t \leq q,$$

where x_{Oi}^{kt} is the total flow of commodity kt on arc $(0,i)$. Also,

$$\sum_{t=1}^{q} \sum_{(0,i)\in Q^t} \sum_{p''(kt)\in Q^{it}} f_{p(kt)} = \sum_{p(k)\in F_B} f_{p(k)} + \sum_{t=1}^{q} (f_{p(kt)} + f_{p'(kt)}).$$

Substituting these in inequality (5.8), we obtain

$$\sum_{t=1}^{q} \sum_{(0,i)\in Q^t} x_{0i}^{kt} + \sum_{i=1} \bar{y}_{0i} \leq (3q-1)/2 \qquad (5.9)$$

Inequality (5.9) is a valid cut for the Arc-flow formulation of the UNDP, [UNDAF], since this formulation is equivalent to the strong Path-flow formulation [UNDSPF] for which (5.8) is valid. Thus, by examining the structure of the Path-flow formulation, we have been able to obtain, in the special case when the given network G contains a Star configuration, a valid inequality for the Arc-flow formulation. These cuts are useful for the algorithms based on the Arc-flow formulation that we discuss in the next chapter.

Generating the 2-hole inequality by constraint aggregation:

Finally, we will demonstrate how the inequality (5.3) induced by the 2-hole S^{2q} (i.e., the facet of P_I^{S2q}) can also be obtained via constraint aggregation. Consider the following valid inequalities for the Path-flow formulation of an NDP that contains a Star configuration:

(a) $f_{p(kt)} + f_{p'(kt)} \leq 1$, $\quad 1 \leq t \leq q$,

(b) $f_{p(kt)} + \bar{y}_{0t} \leq 1$, $\quad 1 \leq t \leq q$, and

(c) $f_{p'(kt)} + \bar{y}_{0,t+1} \leq 1$, $\quad 1 \leq t \leq q$.

Inequalities (a) are implied by the demand constraints (2.2) of [UNDPF], while inequalities (b) and (c) belong to the set of forcing constraints (2.3) (which are relaxed versions of the strong forcing constraints

(2.3a)). Adding these 3q constraints, and dividing the composite constraint by 2, we obtain

$$\sum_{t=1}^{q} (f_{p(kt)} + f_{p'(kt)}) + \sum_{t=1}^{q} \tilde{y}_{Ot} \leq 3q/2 .$$

The right-hand side of this constraint can be rounded down to (3q-1)/2 (since q is odd, and there is an optimal integer solution to UNDP), resulting in the constraint (5.3) induced by the 2-hole S^{2q}.

This last observation concludes our discussion of inequalities (and the corresponding lifted facets) induced by odd holes of the Intersection graph G_I. Trotter [1975] has identified a much larger class of subgraphs, called webs, of the Intersection graph that induce facial inequalities for the Set packing polytope. The two types of subgraphs that we have considered so far - cliques and odd holes - are special types of webs. Next, we will show that, for the Set packing problem [SP] that is of interest to us, the corresponding Intersection graph G_I contains only these two types of webs.

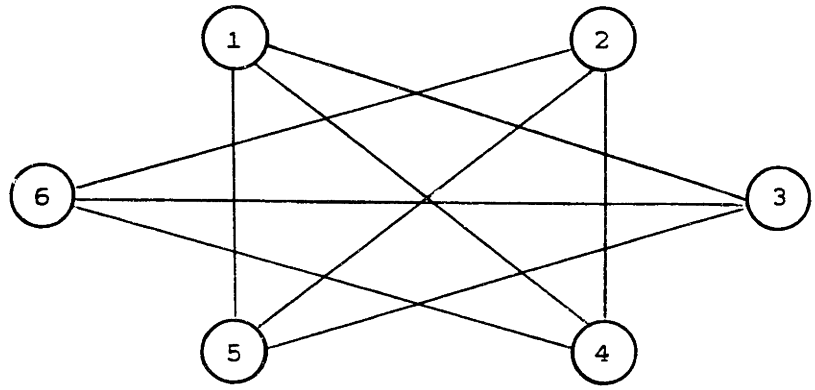
3.4 Webs of the Intersection Graph

Definition 3.8:

> Consider a node-induced subgraph G':(V',E') with $q \geq 2$ vertices labeled 1,2,...,q. G' is called a WEB, denoted as W(q,t), if for all vertices $i,j \in V'$, the edge (i,j) is included in E' if and only if $j = i+t, i+t+1, \ldots, i+q-t$, (where all the sums are taken modulo q), with $1 \leq t \leq \lfloor q/2 \rfloor$.

Figure 15 shows the web W(6,2). Note that the degree of each node in the

Figure 15: Example of a web



Web W(6,2): q = 6, t = 2

web W(q,t) is (q-2t+1). Also, W(q,t) has exactly q vertex packings, each of cardinality t. W(q,1) is a clique on q nodes, while W(2t+1,t), for $t \geq 2$, is an odd hole with (2t+1) vertices. Further, odd holes are the only webs with degree 2 for all nodes. We will now show that, apart from cliques and odd holes, the Intersection graph G_I corresponding to [SP] does not contain any other type of connected webs.

Claim:

The Intersection graph G_I does not contain any 'connected' webs W(q,t) other than odd holes and cliques.

Proof:

Suppose W(q,t) is a connected web of G_I that is neither a clique nor an

odd hole.

(a) Since W(q,t) is not a clique,

$$\text{the degree of each node} < q - 1$$

$$\Longrightarrow \quad q - 2t + 1 < q - 1$$

$$\Longrightarrow \quad t \geq 2.$$

(b) Since W(q,t) is connected and not an odd hole,

$$\text{the degree of each node} \geq 3$$

$$\Longrightarrow \quad q - 2t + 1 \geq 3$$

$$\Longrightarrow \quad t \leq \lfloor (q-2)/2 \rfloor .$$

(c) Since W(q,t) is connected and the degree of each node is less than (q-1), at least one of the nodes of V' must be a y-node. (Since W(q,t) is not a clique, all the nodes of V' cannot be f-nodes corresponding to the same commodity. If V' consists of only f-nodes that correspond to more than one commodity, then the subgraph induced by V' must necessarily be disconnected).

Without loss of generality, assume that node 1 is a y-node. Then, the nodes (t+1),(t+2),...,(q-t+1) must all be f-nodes (since no two y-nodes are adjacent to each other in G_I). From (a), $t \geq 2$; hence, node (t+1) is not connected to node (t+2) in W(q,t). However, both these nodes are f-nodes; therefore, they must correspond to different commodities. Similarly, nodes (t+2) and (t+3) must correspond to different commodities (though, (t+1) and (t+3) may correspond to the same commodity) and so on; nodes (q-t) and (q-t+1) must be f-nodes corresponding to different commodities. Now, since the degree of each node is greater than or equal to 3, node (q-t) must be adjacent at least to the nodes (q-t+t), (q-t+t+1)mod q, and (q-t+t+2)mod q, i.e., to nodes q, 1, and 2. Similarly, node (q-t+1) must be adjacent to nodes 1, 2,

and 3. Notice that (q-t) and (q-t+1) are two f-nodes corresponding to different commodities that are both connected to node 2. Hence, node 2 must be a y-node. Thus, starting with the hypothesis that node 1 is a y-node, we have shown that node 2 must also be a y-node. We can repeat this argument, starting next with node 2, to prove that node 3 is a y-node, and so on till we arrive at the contradiction that node (t+1) is a y-node. Hence, the only connected webs that the Intersection graph G_I contains are cliques and odd holes.

This result implies that the other types of web-induced facial inequalities for general Set packing problems that Trotter [1975] developed are not applicable to the current context. Next, we consider inequalities (5.1) induced by other classes of subgraphs of G_I.

3.5 Inequalities Induced by Other Subgraphs of the Intersection Graph G_I

For the Plant Location problem, Cornuejols and Thizy [1982] have developed two other classes of inequalities that are induced by non-web subgraphs of the Intersection graph. They have also shown that these inequalities can be lifted (using the sequential procedure, for instance) into facets of the Plant Location polytope. Similar subgraphs arise in the Intersection graph G_I corresponding to the NDP, if the original graph G contains the Star configuration discussed in Section 3.2. For these instances, we will extend the PLP inequalities to the Path-flow formulation of the UNDP; while the subgraph-induced inequalities are the same for both

the PLP and the NDP, the lifted facets have slightly different forms for the two problems.

Consider the following modified version of the Star configuration that was introduced earlier. There are m commodities $k_1,\ldots,k_m$, each with node 0 as the origin, and with distinct destinations $D_1,\ldots,D_m$. Node 0 is adjacent, in the given graph G, to q nodes, denoted as $1,2,\ldots,q$. Any or all these nodes might also be destinations of the m commodities. For $1\leq i\leq p$ and $1\leq r\leq m$, we will let $i(k_r)$ denote the path $0\text{-}i\text{-}D_r$ for commodity k_r. For any given m x q matrix C, let G(C) be the subgraph of the Intersection graph G_I induced by the set of vertices V(C), where

$$V(C) = F(C) \cup Y(C),$$

$$F(C) = \{[i(k_r)] : 1\leq r\leq m,\ 1\leq i\leq m,\ c_{ti} = 1\}, \text{ and}$$

$$Y(C) = \{[0i],\ 1\leq i\leq m\}\ .$$

Thus, the rows of C correspond to commodities and the columns correspond to the q nodes $1,2,\ldots,q$. F(C) is the set of f-nodes corresponding to the nonzero elements of C; Y(C) contains all the y-nodes of G_I corresponding to the arcs (0,i) of G. Observe that in the node-induced subgraph G(C), each f-node is adjacent to exactly one y-node (in particular, the vertex $[i(k_r)]$ of F(C) is adjacent to the y-node $[0i] \in Y(C)$). The Intersection graph corresponding to the Plant Location problem also possesses this feature. (In contrast, f-nodes are normally adjacent to more than one y-node in G_I, the Intersection graph corresponding to the NDP). This similarity allows us to extend Cornuejols and Thizy's [1982] cuts for the Plant Location problem to the NDP.

Consider now the C matrix whose rows are all the possible distinct

zero-one vectors with $t < q$ 1's and $(q-t)$ 0's; thus, $m = \binom{q}{t}$. We will let E^{qt} denote such a matrix. $E^{4,3}$, for instance, has the following form:

$$E^{4,3} = \begin{vmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{vmatrix} .$$

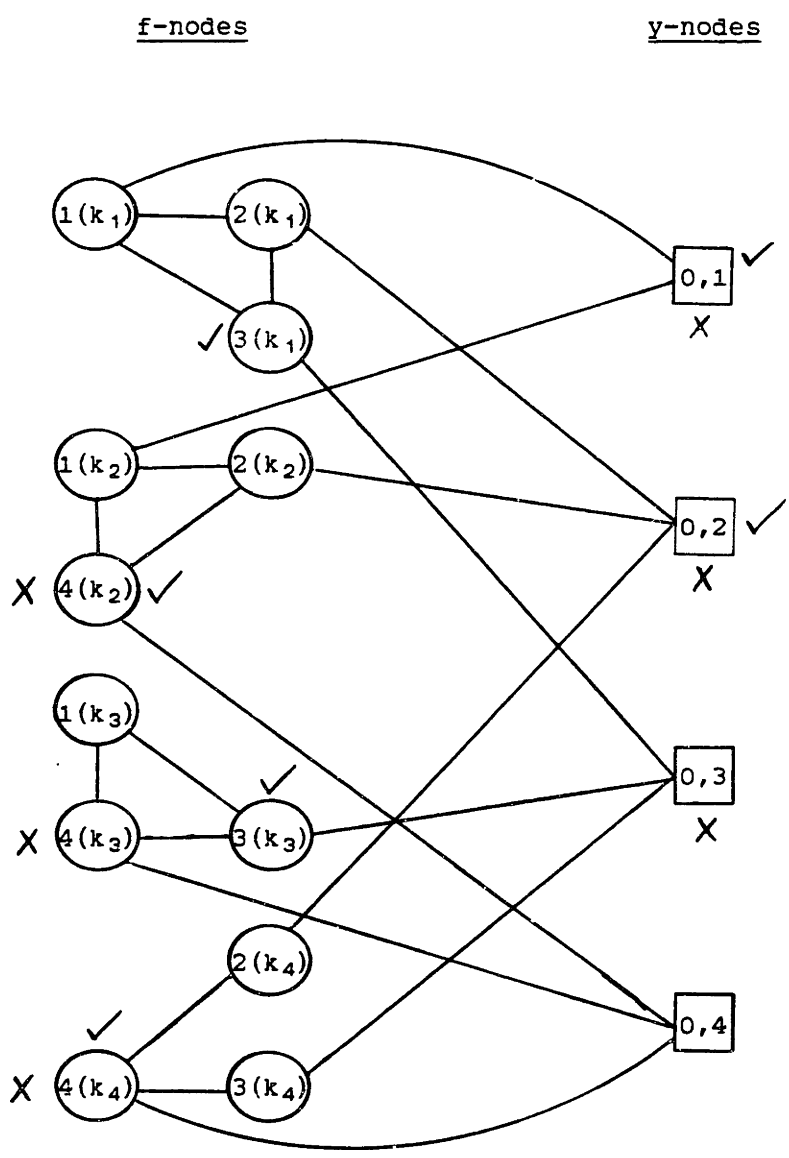
Figure 16 shows the subgraph $G(E^{4,3})$ of G_I corresponding to this matrix.

Cornuejols and Thizy [1982] have shown that the Independence number of the subgraph of the Plant Location Intersection graph induced by the matrix E^{qt} is $(\binom{q}{t} + t - 1)$; since, in the presence of a Star configuration, the corresponding subgraph of G_I is the same as that for the PLP, the independence number of the subgraph that we are considering is also $(\binom{q}{t} + t - 1)$. Hence, the inequality (5.1) induced by the node-induced subgraph $G(E^{qt})$ of G_I is

$$\sum_{r=1}^{m} \sum_{i=1}^{q} e^{qt}_{ri} f_{i(kr)} + \sum_{i=1}^{q} \bar{y}_{0i} \leq \binom{q}{t} + t - 1 \qquad (5.10)$$

where e^{qt}_{ri} is the element in the r^{th} row and i^{th} column of the matrix E^{qt}. As before, $P_I^{E(qt)}$ denotes the lower dimensional polytope that is the convex hull of all feasible [SP] solutions with variables corresponding to vertices not in $V(E^{qt})$ equal to zero. The inequalities (5.1) corresponding to the node-induced cliques and odd holes of G_I, that we examined earlier, were shown to be facets for the corresponding lower-dimensional polytopes. Similarly, we are now interested in determining whether the inequality (5.10) induced by $G(E^{qt})$ is a facet of $P_I^{E(qt)}$. Cornuejols and Thizy [1982] have established this fact in the context of the Plant Location problem. In Lemma 7, we express this result in terms of the Star subnetwork of the

Figure 16: The subgraph $G(E^{4,3})$ of the Intersection Graph G_I



Two types of maximum independent sets marked with 'x' and '✓'.

NDP.

Lemma 7: (Cornuejols and Thizy [1982])

Suppose the given Network Design problem contains a Star subnetwork. Then, the inequality (5.10) induced by the subgraph $G(E^{qt})$, for any given $\binom{q}{t}$ x q zero-one matrix E^{qt} with t 1's in each row, is a facet of $P_I^{E(qt)}$.

Cornuejols and Thizy [1982] have shown that a sufficient condition, due to Balas and Zemel [1977], for the inequality (5.1) corresponding to any node-induced subgraph G' of the Intersection graph to be a facet of $P_I^{G'}$, holds in this case; this observation establishes the result. Furthermore, for the PLP, Cornuejols and Thizy have shown that all sequential liftings of (5.10) must satisfy certain properties. These properties are, however, not directly applicable in the context of the NDP. We will now examine the form of sequential liftings of (5.10), given the structure of the Intersection graph G_I corresponding to [SP].

When we apply the Sequential lifting procedure described in Section 3.3.3 to the subgraph-induced inequality (5.10), we obtain a lifted facet of the following form:

$$\sum_{r=1}^{m}\sum_{i=1}^{q} e_{ri}^{qt} f_{i(kr)} + \sum_{i=1}^{q} \bar{y}_{0i} + \sum_{p(k)\in F'} a_{p(k)} f_{p(k)} + \sum_{i,j\in Y'} b_{ij}\bar{y}_{ij} \leq \binom{q}{t} + t - 1, \tag{5.11}$$

where F' and Y' are the set of path and arc indices that are not included in the first two summations.

As before, we will use Lemma 4 and Corollary 4.1 to identify f variables $f_{p(k)}$ and y variables $\bar{y}_{ij}$ for which the lifting coefficients $a_{p(k)}$ and b_{ij}

must be zero. For this purpose, we will first describe the structure of maximum independent sets in $G(E^{qt})$. Suppose s y-nodes, say $[0,1],\ldots,[0,s]$, belong to an independent set. This set can be 'completed' maximally by adding, for each commodity k_r, for $1 \leq r \leq m$, with $e_{ri}^{qt} = 1$ for some $i > s$, one of the corresponding f-nodes $[i(k_r)]$, for $i > s$ with $e_{ri}^{qt} = 1$. It is possible to show that independent sets such as these with $s = t$ or $s = t+1$ have the maximum cardinality (of $\binom{p}{t}+t-1$). The two types of maximum independent sets in the subgraph $G(E^{43})$ are indicated in Figure 16. Observe that, in general, given any set of (p-t+1) or fewer y-nodes of $G(E^{qt})$, it is always possible to identify a maximum vertex packing that does not contain any of these y-nodes. Similarly, given any particular commodity k_r, for $1 \leq r \leq m$, we can always find a maximum independent set that does not contain any of the f-nodes of $G(E^{qt})$ corresponding to this commodity.

For convenience, we will denote the set of all f-nodes not in $V(E^{qt})$ as F', the set of all y-nodes not in $V(E^{qt})$ as Y', and the set of all commodities k, none of whose f-nodes belong to $V(E^{qt})$, as K'. Then, using Lemma 4 and Corollary 4.1, we can set the following lifting coefficients to zero.

(a) Consider f-nodes $[p(k)] \in F'$ corresponding to commodities $k \in K'$. Since all paths in the Path-flow formulation are simple paths, p(k) contains at most one of the arcs (0,i), for $1 \leq i \leq q$. Furthermore, since $k = k_r$, for $1 \leq r \leq m$, the vertex $[p(k)]$ is not adjacent in G_I to any of the f-nodes of $F(E^{qt})$. Thus, $[p(k)]$ is adjacent to at most one vertex of $V(E^{qt})$, implying that a vertex packing with cardinality $\binom{q}{t} + t = IND(G(E^{qt})) + 1$ can be found in the subgraph of G_I induced

by the set of vertices $V(E^{qt}) \cup \{[p(k)]\}$. Therefore, by Lemma 4, the coefficient of $f_{p(k)}$ must be zero in any lifting of (5.10).

(2) Consider the f-nodes $[p(k_r)] \in F'$, for $1 \leq r \leq m$, such that the corresponding path $p(k_r)$ does not contain any of the arcs $(0,i)$, for $1 \leq i \leq q$. In this case, $[p(k_r)]$ is adjacent to only the f-nodes corresponding to commodity k_r of $G(E^{qt})$. The observation about independent sets of $G(e^{qt})$ made earlier implies that the subgraph induced by $V(E^{qt}) \cup \{[p(k_r)]\}$ must contain a vertex packing of size $\binom{q}{t} + t$; consequently, the coefficient $a_{p(kr)}$ in (5.11) must be zero.

(3) Consider the f-nodes $[p(k_r)] \in F'$, for $1 \leq r \leq m$, such that the corresponding path $p(k_r)$ contains arc $(0,i)$, for some $1 \leq i \leq q$ with $e_{ri}^{qt} = 0$. (As mentioned earlier, since $p(k_r)$ is a simple path, it cannot contain more than one arc from the set $\{(0,i): 1 \leq i \leq q\}$). In this case, the optimal completion of the set $\{[0,i]: 1 \leq i \leq q$, and $e_{ri}^{qt} = 1\} \cup \{[p(k_r)]\}$ is an independent set with $\binom{q}{t} + t$ vertices of the subgraph induced by $V(E^{qt}) \cup \{[p(k_r)]\}$; hence, $a_{p(kr)}$ must be zero in the lifted facet (5.11).

(4) Consider y-nodes $[ij] \in Y'$. All y-nodes of Y' other than those in the set $Y'' = \{[i,D_r]: 1 \leq i \leq q, 1 \leq r \leq m$, and $e_{ri}^{qt} = 1\}$ correspond to arcs of the original graph that are not contained in any of the paths $i(k_r)$, for $1 \leq i \leq q$ and $1 \leq r \leq m$. Thus, none of the vertices $[ij] \in Y' \setminus Y''$ are adjacent to any node of $G(E^{qt})$, implying that the lifting coefficients b_{ij} for all the corresponding $\bar{y}$ variables in (5.11) must be zero. .

Now, consider y-nodes $[i,D_r]$ belonging to the set Y". Each such vertex is adjacent to exactly one vertex of $G(E^{qt})$, namely the f-node $[i(k_r)]$ corresponding to the path $i(k_r) = 0\text{-}i\text{-}D_r$. Again, the coefficient b_{ij} corresponding to such $\bar{y}$ variables must be zero, since

the subgraph induced by $V(E^{qt})$ U $\{[ij]\}$ has an independent set of cardinality $IND(G(E^{qt})) + 1$.

Properties (1) - (4) imply that the only vertices not belonging to $V(E^{qt})$ that can have nonzero coefficients in any lifting of (5.10) are f-nodes $[p(k_r)]$, for $1 \leq r \leq m$, with arc $(0,i) \epsilon p(k_r)$ for some i such that $e^{qt}_{ri} = 1$, for $1 \leq i \leq q$. Such vertices are adjacent in G_I to (a) all the f-nodes of $F(E^{qt})$ corresponding to commodity k_r, and (b) the y-node [0i] for which $(0,i) \epsilon p(k_r)$, and $e^{qt}_{ri} = 1$. We next show that the coefficient $a_{p(kr)}$ corresponding to all such vertices has to be 1 in every sequential lifting (5.11) of (5.10). Consider such a f-node $[p(k_r)]$, and without loss of generality assume that $e^{qt}_{ri} = 1$ for i=1,2,...,t, 0 otherwise and that the arc (0,1) belongs to path $p(k_r)$. Then, the optimal completion of the set $\{[0,i],\ i = 2,...,t\}$ U $\{[p(k_r)]\}$ is a maximum independent set of cardinality $IND(G(E^{qt})$ of the subgraph induced by $V(E^{qt})$ U $\{[p(k_r)]\}$; moreover, this vertex packing contains the node $[p(k_r)]$ and is the optimal solution of the subproblem $[VP_s]$ as the s^{th} stage of the Sequential procedure, regardless of the order s in which the vertex $[p(k_r)]$ is considered. This fact implies that the optimal value z_s of $[VP_s]$ is $(IND(G(E^{qt}) - 1)$, and thus the coefficient $a_{p(kr)}$ is 1 in any sequentially lifted facet (5.11).

The foregoing discussions lead to Proposition 8.

PROPOSITION 8:

Suppose the given Network Design problem has an embedded Star configuration. For any zero-one matrix E^{qt} with t 1's in each row,

and $\binom{q}{t}$ rows, the unique sequential lifting of the valid inequality (5.10) induced by the corresponding subgraph $G(E^{qt})$ is

$$\sum_{i=1}^{p} \sum_{r=1}^{m} e_{ri}^{qt} f_{i(kr)} + \sum_{r=1}^{m} \bar{y}_{0i} + \sum_{r=1}^{m} \sum_{p(kr)\in Q'} f_{p(kr)} \leq \binom{q}{t} + t - 1 \qquad (5.12)$$

where $Q'_r = \{p(k_r) \in F(E^{qt})$, and $(0,i) \in p(k_r)$ for some $1 \leq i \leq q$, with $e_{ri}^{qt} = 1\}$

Again the facet (5.12) can be expressed in terms of the Arc-flow variables x_{0i}^{kr} rather than the Path-flow variables $f_{p(kr)}$. This is possible because (5.12) involves adding up the flows for a given commodity, say k_r, over <u>all</u> the paths of the original graph G that use certain arcs, namely the arcs (0,i) with $1 \leq i \leq q$ and $e_{ri}^{qt} = 1$. Thus, (5.12) can equivalently be written as

$$\sum_{r=1}^{m} \sum_{i=1}^{q} e_{ri}^{qt} x_{0i}^{kr} + \sum_{i=1}^{q} \bar{y}_{0i} \leq \binom{q}{t} + t - 1 ,$$

or, alternatively, in terms of the original y-variables as

$$\sum_{r=1}^{m} \sum_{i=1}^{q} e_{ri}^{qt} x_{0i}^{kr} - \sum_{i=1}^{q} y_{0i} \leq \binom{q}{t} + t - q - 1, \qquad (5.13)$$

where x_{0i}^{kr} is the flow of commodity k_r on arc (0,i) .

Recall that, whenever the original Network Design problem has an embedded Star configuration, we can describe several node-induced subgraphs G(C) of the corresponding Intersection graph G_I using the zero-one matrix C. We have just examined the nature of the induced inequalities when the matrix C has the special structure described as E^{qt}. One approach for generating additional classes of such valid inequalities would be to examine other special types of matrices. We would have to determine if the

inequality (5.1) induced by the corresponding subgraphs G(C) of G_I is a facet for the lower dimensional polytope $P_I^{G(C)}$, and then identify the form of the lifting coefficients. We next describe briefly one other such matrix used by Cornuejols and Thizy [1982] for generating facets of the Plant Location polytope.

Consider the q x q cyclic matrix C^{qt} whose rows consist of t consecutive ones successively shifted one position to the right. We first considered this matrix in Section 2.2, where we showed that the inequalities derived by Cornuejols et al. [1977] for the Plant Location problem can also be obtained using the Auxiliary graph G_a and the related dual solution (u^*, z_0^*). In Section 3.3.7, we considered 2-holes in the Intersection graph G_I that arise when the original network has a Star configuration S^{2q}. The corresponding matrix C in that case was a cyclic matrix with 2 consecutive 1's in each row. We showed that the Independence number of the 2-hole was (3q-1)/2 and examined the structure of the sequentially lifted facet derived from the inequality induced by S^{2q}. We will now consider the subgraphs $G(C^{qt})$ corresponding to general cyclic matrices in which q and t are relatively prime. Cornuejols and Thizy [1982] have shown, in the context of the Plant Location problem, that the Independence number of this subgraph is $2q - \lfloor q/t \rfloor$, where $\lfloor a \rfloor$ is the smallest integer greater than or equal to a. They have also shown that the inequality (5.1) induced by this subgraph is a facet for the lower-dimensional polytope $P_I^{C(q,t)}$ (and hence, can be lifted into a facet for the original polytope P_I) if and only if the parameters q and t are such that $q = st + 1$ for some integer s. Since, the subgraph of G_I corresponding to the cyclic matrix C^{qt} is the same for the UNDP with an

embedded Star configuration as it is for the Plant Location problem, Cornuejols and Thizy's results are applicable in the current context as well. In other words,

Lemma 8: (Cornuejols and Thizy [1982])

Suppose the given Network Design problem has an embedded Star configuration. Then, for any cyclic matrix C^{qt}, $q = st + 1$ for some integer s, the inequality induced by the subgraph $G(C^{qt})$ of G_I:

$$\sum_{r=1}^{q} \sum_{i=1}^{q} c^{qt}_{ri} f_{i(kr)} + \sum_{i=1}^{q} \bar{y}_{Oi} \leq 2q - |q/t| \qquad (5.14)$$

is a facet of $P_I^{C(q,t)}$ that can be lifted into a facet of P_I.

As before, we can set some of the lifting coefficients to zero a priori using information about the structure of maximum independent sets in the subgraph $G(C^{qt})$; similarly, we can bound the nonzero coefficients. We do not pursue this further here.

We now conclude our discussion of inequalities that are derived using node-induced subgraphs of G_I. In each case that we considered, the inequality (5.1) induced by the subgraph G' of G_I was known to be a face of P_I that could be lifted into a facet of P_I using, for instance, the Sequential lifting procedure. We identified several variables whose lifting coefficients must necessarily be zero and determined, in some cases, the unique sequentially lifted facet completely. Finally, we were able to express two of the facets in terms of Arc-flow variables, thus providing valid inequalities for the Arc-flow formulation of the UNDP. We

will now describe a different approach for generating valid inequalities that is applicable specifically to the Capacitated Network Design problem in Arc-flow form.

3.6 Cutset Inequalities for the Capacitated Network Design Problem

Consider the Arc-flow formulation [NDAF] of the Capacitated Network Design problem outlined in Section 1.3. We will now describe a method for generating valid inequalities that strengthen the LP relaxation of this problem. The method involves identifying cutsets of the given network and ensuring that the total capacity of the arcs belonging to this cutset that are included in the final network design is enough to meet all the flow requirements across the cut.

Definition 3.9:

> Given a partition (U,U') of the set of nodes N of the given network G:(N,A), the corresponding CUTSET, denoted as CS(U), is the set of all arcs $(i,j) \epsilon A$, such that $i \epsilon U$ and $j \epsilon U'$.

Recall that, in the general Network Design problem, each arc $(i,j) \epsilon A$ has a Capacity of B_{ij} units, and R_k units of every commodity $k \epsilon K$ have to transported from the Origin O(k) to Destination D(k) along the arcs that are included in network design. Given the partition (U,U') of N, let

$$K(U) = \{k \epsilon K: O(k) \epsilon U, D(K) \epsilon U'\}, \text{ and}$$

$$R(U) = \sum_{k \epsilon K(U)} R_k.$$

Thus, K(U) is the set of all commodities that must flow across the cutset CS(U), and R(U) is the total demand requirements of these commodities.

Clearly, the capacity of all the arcs belonging to CS(U) that are included in the final network design must be greater than or equal to R(U). This constraint can be expressed as

$$\sum_{(i,j)\in CS(U)} B_{ij}y_{ij} \geq R(U). \qquad (6.1)$$

Constraint (6.1) is a valid inequality for the CNDP for all partitions (U,U') of the node set N. However, (6.1) is redundant even in the LP relaxation of the formulation [NDAF] as the following argument demonstrates. Let [L3] be the LP relaxation of [NDAF].

Claim:

Inequality (6.1) is redundant in [L3] for all partitions (U,U') of N.

Proof:

Let (x^*,y^*) be any feasible solution of the LP relaxaton [L3]. Then,

$$\sum_{(i,j)\in CS(U)} B_{ij}y_{ij} \geq \sum_{(i,j)\in CS(U)} \sum_{k\in K(U)} x_{ij}^{k*}$$

$$\geq \sum_{k\in K(U)} R_k = R(U),$$

where the first inequality holds because the flow vector x^* satisfies the capacity constraints (1.4) of [NDAF] and the second inequality follows from the fact that x^* satisfies the flow conservation equations (1.2) of [NDAF]. Thus, the LP solution (x^*,y^*) satisfies the inequality (6.1), implying that (6.1) is redundant in [L3].

While constraints (6.1) by themselves are redundant in the LP relaxation of [NDAF], logical constraints (on the design variables) derived from (6.1) and inequalities obtained by reducing the coefficients of (6.1) may not be

redundant in [L3]. We next describe how such nonredundant constraints can be obtained.

3.6.1 Coefficient reduction:

Since the design variables y_{ij} are required to be binary, we can replace the coefficients B_{ij} in the left-hand side of inequality (6.1) by Min $\{B_{ij},R(U)\}$ to obtain the following valid constraint for [NDAF]:

$$\sum_{i,j\in CS(U)} \text{Min } \{B_{ij},R(U)\}y_{ij} \geq R(U). \qquad (6.2)$$

Constraint (6.2) is tighter than (6.1) if there is at least one arc (i,j) in the given cutset CS(U) for which $B_{ij} > R(U)$. In this case, (6.2) is not redundant in [L3]. Martin and Schrage [1983] have outlined a general procedure that uses constraint aggregation coupled with a similar coefficient reduction procedure to systematically generate valid inequalities for mixed zero-one problems.

In general, given a LP solution (x^*,y^*) it might be difficult to identify all the cutsets CS(U) for which the corresponding inequality (6.2) is violated. Therefore, one might consider only special types of node sets U such as the following: Suppose the set U consists of a single node, say node i. Then, inequality (6.2) reduces to

$$\sum_{j\in J(i)} \text{Min } \{B_{ij}R(i)\}y_{ij} \geq R(i), \qquad (6.3)$$

where R(i) is the sum of the demands for all commodities originating at node i, and

$$J(i) = \{j\in N: (i,j)\in A\} .$$

Given a solution (x^*,y^*) of [L3], it is easy to determine if constraint

(6.3) is violated for any of the nodes $i \epsilon N$; if it is, the LP relaxation can be strengthened by adding this constraint.

Finally, a valid constraint similar to (6.2) can also be formulated by considering a subset K'(U) of commodities in K(U). If R'(U) is the sum of the demands for the commodities in the set K'(U), the inequality (6.2) with coefficients Min $\{B_{ij}, R'(U)\}$ in the right-hand side and R'(U) in the left-hand side is also valid for [NDAF]. However, it can be easily shown that this new constraint is not as tight as the original constraint (6.2) that is formulated with respect to the complete set of commodities K(U).

3.6.2 Minimal Cover cuts:

Logical constraints on the design variables y_{ij} can be derived from the cutset inequality (6.1) using the concept of Minimal cover cuts proposed by Crowder, Johnson and Padberg [1983]. Such constraints are constructed as follows: Suppose, for a given partition (U,U') of N, there is a subset of arcs CS'(U) belonging to the cutset CS(U) and satisfying

$$\sum_{i,j \epsilon CS'(U)} B_{ij} < R(U) \quad \text{and} \tag{6.4a}$$

$$\sum_{i,j \epsilon CS'(U)} B_{ij} + B_{i'j'} \geq R(U) \quad \text{for all } (i',j') \epsilon CS(U) \backslash CS'(U). \tag{6.4b}$$

Then, the logical inequality

$$\sum_{i,j \epsilon CS''(U)} y_{ij} \geq 1, \tag{6.5}$$

where $CS''(U) = CS(U) \backslash CS'(U)$,

is a valid cut for [NDAF]. Since the sum of the capacities of all the arcs

belonging to the subset CS'(U) is not sufficient to "cover" the flow requirements across the cut (condition (6.4a)), at least one arc from the set CS"(U) must be included in the network design; hence, the validity of constraint (6.5). Notice that (6.5) is valid even if the given subset CS'(U) does not satisfy condition (6.4b); however, choosing subsets CS'(U) that satisfy (6.4b) ensures that (6.5) is as tight as possible. The subset CS"(U) = CS(U)\CS'(U) is often referred to as the MINIMAL COVER (with respect to the given cutset CS(U)). Each cutset CS(U) might have several Minimal covers. The Minimal cover with the least number of elements can be identified as follows: Arrange the arcs of CS(U) in order of non-decreasing capacity. Let CAP(s) be the cumulative capacity of the first s arcs (after reodering) of CS(U). If s^* is the largest integer such that CAP(s^*) is less than R(U), then the last $|CS(U)| - s^*$ arcs of CS(U) constitute a Minimal cover of CS(U) containing the least number of arcs.

Given any LP solution (x^*,y^*) of the LP relaxation of [NDAF], and a cutset CS(U), the problem of finding the most violated Minimal cover cut (6.5) can be posed as a Binary Knapsack problem. Define the binary variable s_{ij} as follows:

$$s_{ij} = \begin{cases} 1 \text{ if arc } (i,j) \text{ is included in the Minimal cover CS"(U)} \\ 0 \text{ otherwise} \end{cases}$$

Let $z^*(U)$ be the optimal value of the following Knapsack problem:

$$z^*(U) = \text{Min} \sum_{i,j \in CS(U)} \bar{y}_{ij} s_{ij}$$

subject to

$$\sum_{i,j \in CS(U)} B_{ij} (1 - s_{ij}) < R(U)$$

$$s_{ij} = 0 \text{ or } 1 \quad \text{for all } (i,j) \in CS(U) .$$

Then, the given solution violates one of the Minimal cover cuts (6.5)

formulated with respect to the cutset CS(U) if and only if $z^*(U) < 1$; and, if $z^*(U) < 1$, the most violated Minimal cover cut is the one formulated using the cover $CS''(U) = \{(i,j) \epsilon CS(U): \ s_{ij} = 1\}$.

Finally, let us once again examine the form of (6.5) when the set U that defines the cutset CS(U) consists of a single node $i \epsilon N$, i.e., $U = \{i\}$. In this case, we have to identify a set $J'(i) \subseteq J(i) = \{j \epsilon N: \ (i,j) \epsilon A\}$ such that

$$\sum_{j \epsilon J'(i)} B_{ij} < R(i), \text{ the sum of the demands of commodities originating at node i,}$$

and the corresponding valid inequality is

$$\sum_{j \epsilon J''(i)} y_{ij} \geq 1 \ ,$$

where $J''(i) = J(i) \backslash J'(i)$.

Notice that if the capacity of all the arcs (i,j), $j \epsilon J(i)$ is greater than or equal to R(i), then this inequality becomes

$$\sum_{j \epsilon J(i)} y_{ij} \geq 1, \text{ assuming } R(i) > 0,$$

which is an obvious constraint (since $R(i) > 0$, node i is a source for at least one commodity; hence, at least one of the arcs incident from node i must be included in the network design).

Van Roy and Wolsey [1984] have discussed a different class of inequalities for the Uncapacitated Network Design problem that is also derived by examining cutsets of the given network. While they consider a single commodity problem, their cuts can be readily extended to the multicommodity case that we are interested in. They have also discussed several special cases, including the lot-sizing and multilevel distribution problem, for which the problem of identifying a violated inequality for a

given fractional solution is solvable in polynomial time.

3.7 Concluding Remarks

In this chapter, we have examined various valid inequalities that are not redundant in the LP relaxation of the Network Design problem. The major part of this chapter was devoted to constructing such inequalities for the strong disaggregated Path-flow formulation [UNDSPF] of the Uncapacitated Network Design problem. By converting this formulation into Set packing form, we were able to specialize the facial inequalities derived by Trotter [1975] and Padberg [1977] for the general Set packing problem. The Intersection graph G_I was used throughout as a tool for identifying such inequalities, and because of the special structure of this graph for the NDP, we were able to narrow down considerably the possible values of the lifting coefficients in facets derived from these inequalities. We introduced a structure called the Star configuration; Network design problems defined over complete networks, with several commodities originating at a single node, contain such subnetworks, for example. We exploited the similarity between this configuration and the Plant Location problem to extend some of the inequalities proposed by Cornuejols and Thizy [1982] for the PLP. Surprisingly, two of these inequalities could be expressed in terms of Arc-flow variables as well, leading to valid cuts for the Arc-flow formulation [NDAF] of the UNDP.

All but the last class of (cutset) inequalities discussed in this chapter are known to be facets of the Set packing polytope P_I. However, we

have not been able to prove that these cuts are facets of the integer polytope corresponding to the original UNDP formulation [UNDSPF]. Also, the inequalities that we discussed do not constitute a complete facial description of the Set packing polytope.

We were able to relate the subgraph-induced families of inequalities to those derived using constraint aggregation procedures and the constraint generation methods of Chapter 2. The emphasis throughout this chapter has, however, been on the description of valid inequalities, rather than on algorithmic procedures for identifying, given a particular LP solution, violated, valid inequalities. Furthermore, as mentioned at the outset, we have concerned ourselves solely with inequalities that can be generated by examining the constraint set alone. We indicate, in the next chapter, how objective function and dual information can be used to identify additional violated cuts.

Beginning with Chapter 4, we will focus on the development and evaluation of algorithms for solving the Network Design problem. We will work with the Arc-flow formulation rather than the Path-flow formulation. In Chapter 4, we present a general framework for studying Dual Ascent algorithms for the NDP. We discuss specific implementations of this algorithm, extensions to the Capacitated case, ways in which some of the valid inequalities discussed in the last two chapters can be incorporated in the solution procedure, and some computational results for the UNDP. In Chapter 5, we describe a composite Ascent/Subgradient procedure for solving a special type of Network Design problem called the LTL Consolidation problem.

VALID INEQUALITIES AND ALGORITHMS FOR THE NETWORK DESIGN PROBLEM WITH AN APPLICATION TO LTL CONSOLIDATION

by

ANANTARAM BALAKRISHNAN

Bachelor of Technology,
Indian Institute of Technology, Madras, India
(1976)

Post-graduate Diploma in Management,
Indian Institute of Management, Ahmedabad, India
(1978)

Submitted to the
Sloan School of Management
in Partial Fulfillment of the
Requirements of the Degree of

DOCTOR OF PHILOSOPHY

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

December 1984

Signature of Author: Anantaram Balakrishnan
Sloan School of Management, 27 December 1984

Certified by: Thomas L. Magnanti
Thomas L. Magnanti, Thesis Supervisor

Accepted by: Richard Schmalensee
Richard Schmalensee, Chairman, PhD. program

CHAPTER 4

DUAL ASCENT ALGORITHMS FOR THE NETWORK DESIGN PROBLEM

In this chapter, we examine algorithmic approaches for solving the Network Design problem. Our discussion focuses mainly on dual ascent algorithms for the uncapacitated and capacitated versions of the problem. We also discuss a few problem reduction methods and present some computational results.

Even though some of its special cases, such as the Shortest path and Minimal Spanning tree problems, can be solved in polynomial time, the Network Design problem, in its general form, is known to be NP-hard (Johnson et al. [1978]). Consequently, it is unlikely that a polynomial time algorithm can be devised for solving the general problem. Moreover, general purpose MIP codes are unable to solve even medium-sized Network Design problems, because the problem formulation grows very rapidly with the number of arcs and commodities in the network. For instance, the Strong Arc-flow formulation [NDAF] for a problem with 20 nodes, 100 arcs, and 100 commodities contains 100 binary variables, 10,000 continuous variables and over 12,000 constraints. Problems of a much larger size are frequently encountered in the design of transportation and communication networks. As a consequence, for solving the Network Design problem, it is necessary to specialize some of the general purpose algorithms - such as branch-and-bound and implicit enumeration schemes, cutting plane and group theoretic methods, and dynamic programming - that have been proposed for

solving general mixed-integer programs. Indeed, many of the major computational successes in the area of large-scale optimization have been achieved by such specialization. In most of these instances, the key factor has been the ability to exploit special structures in the problem in order to efficiently obtain good lower and upper bounds on the optimal objective function value. For this purpose, researchers have employed various devices including

(1) decomposition of the problem into smaller subproblems that can be solved efficiently,

(2) use of good optimization-based heuristic procedures, and

(3) addition of strong valid inequalities to obtain tight formulations.

Hitherto, Lagrangian relaxation, Benders' decomposition, and dual ascent, have been the three most successful and widely-used lower bound generation strategies. They have been applied to numerous Network design-related problems such as the Facilities Location problem, the Traveling Salesman problem, the Multicommodity Distribution problem, and the Steiner tree problem. Magnanti [1976] has shown that these three techniques are in fact closely related within the framework of mathematical programming duality.

Because of its underlying network structure, the Network Design problem is an especially attractive context for applying these decomposition and lower-bounding methods. In Sections 4.1 and 4.2, we briefly describe how to exploit this structure using Lagrangian relaxation and Benders' decomposition; however, our main focus is on dual ascent schemes. In Section 4.3, we present a general framework for devising various dual ascent schemes for the UNDP; we also discuss some alternative implementations that are possible within this framework. We then extend the algorithm to the capacitated case and discuss how the method can be

modified to handle additional side constraints. In Section 4.4, we discuss techniques for reducing the problem (i.e., fixing some of the binary variables to zero or one) based on objective function and dual information. Finally, in Section 4.6, we present some computational experience with a particular implementation of the dual ascent algorithm. Throughout this chapter, we use the Strong Arc-flow formulation [NDAF] of the NDP.

4.1 Lagrangian Relaxation for the Network Design Problem

Lagrangian Relaxation or Price directive decomposition (Geoffrion [1974], Fisher [1981], Shapiro [1979a]) is most effective for problems that have coupling or "complicating" constraints; without these constraints, the residual problems can usually be solved quite efficiently. In this scheme, a relaxation of the original problem, called the Lagrangian subproblem, is obtained by dualizing the complicating constraints, i.e., by removing them from the constraint set and adding a linear combination of these constraints to the objective function. The multipliers that are used to aggregate the constraints are referred to as Dual or Lagrangian multipliers (or prices). For any given set of multipliers, the objective function value of the Lagrangian subproblem is a lower bound on the optimal objective function value of the original problem. The Lagrangian master problem seeks the multiplier values that yield the best lower bound; subgradient optimization and column-generation techniques can be used to solve this master problem. The lower bound that is obtained in this manner can then be used, for instance, to fathom the nodes in a branch-and-bound tree. Alternatively, solutions of the Lagrangian subproblem can be used to

construct feasible solutions for the original problem; such a method is necessarily a heuristic procedure and has sometimes been referred to as a Lagrangian heuristic (Fisher [1981]).

The impressive computational results reported by Held and Karp ([1970],[1971]) for the Traveling Salesman problem using a Lagrangian relaxation algorithm prompted widespread use of this technique for solving other hard problems related to the NDP. Lagrangian relaxation has subsequently been used, with considerable success, to solve problems in areas such as facility location (Geoffrion and McBride [1978]), computer data base design (Fisher and Hochbaum [1980]), vehicle fleet planning (Fisher et al. [1982]), and topological design of computer networks (Gavish [1982]).

For the same formulation, it is possible to obtain different relaxations by dualizing different constraints. These competing relaxations might differ both in terms of the sharpness and the time required to compute the respective bounds. Fisher [1981] highlights the importance of choosing the appropriate relaxation. We next examine two alternative relaxation schemes for the NDP.

In the formulation [NDAF], the forcing and capacity constraints (1.3) and (1.4) link the design variables and the flow variables. Hence, a natural relaxation of [NDAF] is obtained by dualizing these two sets of constraints, using non-negative multipliers w^k_{ij}, for all $(i,j) \in A$ and $k \in K$, and t_{ij}, for all $(i,j) \in A$, respectively. The corresponding Lagrangian subproblem $LR_1(w,t)$ is

$[LR_1(w,t)]$

$$\text{Minimize} \quad \sum_k \sum_{(i,j)} (c_{ij}^k + w_{ij}^k + t_{ij}) x_{ij}^k + \sum_{(i,j)} (F_{ij} - \{ \sum_k d_{ij}^k w_{ij} \})$$

subject to (1.2) and (1.5),

where $d_{ij}^k = \text{Min } \{B_{ij}, R_k\}$.

Notice that the x and y variables are no longer coupled in this subproblem and that it is always optimal to set

$$y_{ij} = \begin{cases} 1 & \text{if } (F_{ij} - \sum_k d_{ij}^k w_{ij}^k - B_{ij} t_{ij}) < 0 \\ 0 & \text{otherwise.} \end{cases}$$

Also, the subproblem $LR_1(w,t)$ seperates into $|K|$ shortest path problems, one for each commodity. Thus, in the optimal subproblem solution, x_{ij}^k is zero on all arcs except those that lie on the shortest path from O(k) to D(k), with respect to arc costs $(c_{ij}^k + w_{ij}^k + t_{ij})$; a flow of R_k units is routed on this shortest path. Since this relaxation satisfies the Integrality property (Geoffrion [1974]), the best lower bound that can be obtained using this scheme is equal to the optimal value of the LP relaxation of [NDAF].

The Lagrangian lower bound can be improved by

(a) relaxing just the capacity constraints (1.4);
In this case, the subproblem reduces to an Uncapacitated Network Design problem, which is itself NP-hard, or

(b) adding the valid constraints

$$\sum_k x_{ij}^k \leq B_{ij} \quad \text{for all } (i,j) \in A .$$

In the new subproblem, the design variables y_{ij} are again set equal to 1 or 0 depending on whether or not $(F_{ij} - \Sigma_k d_{ij}^k w_{ij}^k - B_{ij} t_{ij})$ is negative. However, in order to determine the optimal values of the flow variables x_{ij}^k we must now solve a Multicommodity flow problem. Even though this enhanced relaxation also satisfies the Integrality property, for any given set of Lagrange multipliers $\{w_{ij}^k\}$ and $\{t_{ij}\}$, the addition of these constraints ensures that the lower bound is at least as tight as that provided by $LR_1(w,t)$; if the arc capacities

B_{ij} are very restrictive in the original problem, we might expect the new bound to be strictly greater for almost all values of the dual multipliers.

These two variants illustrate the usual tradeoff between better lower bounds and more computational effort required to achieve these bounds that is characteristic of most Lagrangian relaxation schemes.

Let us now consider an alternative relaxation $[LR_2(v,t)]$ that is obtained by dualizing the flow conservation equations (1.2) and the capacity constraints (1.4) of [NDAF] using multipliers v_i^k, for all $i \epsilon N$ and $k \epsilon K$, and t_{ij}, for all $(i,j) \epsilon A$, respectively. The subproblem $[LR_2(v,t)]$ is

$[LR_2(v,t)]$

$$\text{Minimize} \quad \sum_k \sum_{(i,j)} (c_{ij}^k + v_i^k - v_j^k + t_{ij}) x_{ij}^k + \sum_{(i,j)} (F_{ij} - B_{ij} t_{ij}) u_{ij}$$

$$\text{subject to} \qquad (1.3) \text{ and } (1.5),$$

for any given set of non-negative multipliers $\{t_{ij}\}$ and multipliers $\{v_i^k\}$ that are unrestricted in sign. The subproblem $[LR_2(v,t)]$ can be solved by inspection in the following manner:

If $(c_{ij}^k + v_i^k - v_j^k + t_{ij})$ is > 0 (respectively, ≤ 0), then it is optimal to set $x_{ij}^k = 0$ (respectively, $= d_{ij}^k y_{ij}$). Hence, if we define

$$e_{ij}^k = \text{Min } \{0,\ c_{ij}^k + v_i^k - v_j^k + t_{ij}\},$$

$[LR_2(v,t)]$ can be rewritten as

$$\text{Minimize} \quad \sum_{(i,j)} (F_{ij} + \{ \sum_k e_{ij}^k d_{ij}^k \} - B_{ij} t_{ij})\ y_{ij}$$

$$\text{subject to} \qquad y_{ij} = 0 \text{ or } 1 \quad \text{for all } (i,j) \epsilon A\ .$$

Thus, the optimal solution of $[LR_2(v,t)]$ is

$$y_{ij} = \begin{cases} 1 & \text{if } (F_{ij} + \{ \sum_k e_{ij}^k d_{ij}^k \} - B_{ij} t_{ij} \leq 0 \\ 0 & \text{otherwise, and} \end{cases}$$

$$x^k_{ij} = \begin{cases} d^k_{ij} y_{ij} & \text{if } e^k_{ij} \leq 0 \\ 0 & \text{otherwise .} \end{cases}$$

It is possible to devise other Lagrangian relaxation schemes for the NDP. For instance, in Chapter 5, we use a relaxation obtained by dualizing only the flow conservation equations (1.2) to derive lower bounds for the LTL Consolidation problem. Gavish [1983] has used a similar relaxation to solve the Capacitated Minimal Spanning tree problem. In all these schemes, the lower bound can be improved by adding to the formulation side constraints that are valid for the original problem, but that are not redundant in the Lagrangian subproblem; of course, as we have already seen, the subproblems would then become more difficult to solve. The choice of the appropriate relaxation can usually be made only after experimenting with alternative schemes for a given problem. Finally, Lagrangian relaxation is very closely related to the dual ascent techniques that we describe later. For instance, the dual ascent procedure of Section 4.3 can also be viewed as a Multiplier Adjustment method, i.e., a heuristic for iteratively changing the dual multipliers (without using either subgradient optimization or column-generation techniques) to increase the Lagrangian subproblem objective function value monotonically. We will later discuss this interpretation in greater detail.

4.2 Benders' Decomposition for the Network Design Problem

Benders' or Resource directive decomposition (Benders [1962], Lasdon [1970]) is an algorithmic strategy that is particularly well-suited for

problems that have, except for a few 'complicating' variables, an underlying problem structure that can be solved efficiently. Algorithms based on Benders' decomposition have been applied successfully to distribution systems design (Geoffrion and Graves [1974]), aircraft routing (Richardson [1976]), railroad engine scheduling (Florian et al. [1976]) and more recently to Fixed-charge Network Design (Magnanti et al. [1984]). All these applications can be modeled as two-stage decision processes with decisions about the complicating variables being made in the first stage; these decisions lead to subproblems whose solution yields the values of the remaining variables. The Benders' decomposition procedure mechanizes this process; it proceeds by iteratively fixing values for the complicating variables in the master problem and then solving the residual subproblems in order to derive additional constraints for the master problem or to establish optimality. Benders' decomposition schemes are often favored not only because of their computational superiority in certain circumstances, but also because they provide feasible solutions at all intermediate stages; this feature permits early termination and enables the use of Benders' decomposition as the basis for good heuristic procedures.

For the Network Design problem, a natural decomposition scheme is to fix the design variables y_{ij} in the master problem; then, the optimal values of the routing or flow variables x^k_{ij} can be determined by solving a minimum-cost multicommodity flow problem over the chosen network. The optimal routing solution is then used to construct Benders' cuts, which when added to the master problem result in a redefinition of the network design. Thus, the basic steps of the Benders' algorithm at the t^{th} iteration are

Step 1: Choose the new tentative network configuration (i.e., set values for the design variables y_{ij}) by solving the current master problem.

Step 2: For the given design, find the optimal routing for all commodities by solving a minimum-cost multicommodity flow problem over the chosen network.

Step 3: Using the optimal dual solution of the multicommodity flow subproblem, construct a valid cut(s) for the master problem.

Step 4: If the current y-vector satisfies the new cut, STOP; the current solution is optimal. Otherwise, add the cut to the master problem, increment the iteration count t and return to Step 1.

(Magnanti et al. [1984] give a more detailed description of the Benders' algorithm as applied to the Uncapacitated Network Design problem; the only difference for the capacitated case is that the subproblems are multicommodity flow problems, rather than Shortest path problems.)

The Benders' cuts generated at each iteration are essentially lower bounds, expressed as a function of the y-variables, on the optimal value of the original problem. They have the following general form:

$$z \geq a_0^t + \sum_{(i,j)} b_{ij}^t y_{ij},$$

where the coefficients a_0^t and b_{ij}^t are determined using the optimal dual solution of the multicommodity subproblem at iteration t. The Benders' master problem after the t^{th} iteration is

$$\text{Minimize} \quad z$$

subject to

$$z \geq a_0^q + \sum_{(i,j)} b_{ij}^q y_{ij} \qquad q = 1,2,\ldots,t,$$

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

This problem is a mixed-integer program with a single continuous variable z and can be solved using either a branch-and-bound algorithm, an enumeration procedure, or a Lagrangian relaxation/ranking procedure similar to the one described by Granot and Zang [1983] for solving Min-Max problems.

Alternatively, by solving the master problem heuristically, we can generate valid lower bounds that might be useful for problem reduction or in a branch-and-bound algorithm. Moreover, we can perturb the feasible solutions that are generated by the algorithm to construct good heuristic solutions for the NDP.

The computational performance of the Benders' decomposition procedure depends, to a large extent, on the strength of the Benders' cuts. The multicommodity flow subproblems frequently have multiple dual solutions and each of these would yield a different Benders' cut. Magnanti and Wong [1981] have proposed a procedure for systematically choosing the dual solution that results in the best possible (or pareto-optimal) Benders' cut. In their experience (Magnanti et al. [1984]), this technique results in a substantial computational improvement while solving Uncapacitated Network Design problems. They have also shown how the different lower bounds for the UNDP proposed in the literature (Hoang [1973], Boyce et al. [1973], Dionne and Florian [1979], Los and Lardinios [1982]) can be interpreted as Benders' cuts that are derived from alternative dual solutions.

Finally, we note that it is not necessary to solve the Benders' subproblems to optimality, especially during the initial stages of the algorithm. Benders' cuts may be derived using any dual feasible solution; of course, the strength of the cuts would depend on how close the dual solutions are to optimality. This observation suggests the possiblility of using an approximate procedure, such as the dual ascent procedure outlined next, within the framework of Benders' decomposition.

4.3 Dual Ascent Procedures for the Network Design Problem

Solving linear programming relaxations of mixed integer programs is, perhaps, the most frequently used method for obtaining lower bounds in branch-and-bound and other enumeration algorithms; even when other bounding methods are employed, the LP relaxation often serves as a benchmark against which the quality of different bounds is compared. Often, the dual of the LP relaxation is easier to solve because of its special structure. Furthermore, for some problems, researchers have found it more advantageous to derive 'good' dual solutions 'fast' rather than solve the dual problems to optimality. (Recall that, in a minimization problem, the objective function value of any dual feasible solution is a lower bound on the optimal value of the LP relaxation; hence, it is also a lower bound on the optimal value of the original problem). Dual ascent procedures are used precisely for this purpose. They attempt to increase the dual objective function value monotonically by a heuristic manipulation of the dual variables that maintains dual feasiblity at all stages. Dual ascent procedures are normally not guaranteed to yield optimal dual solutions, but in most successful applications they provide near-optimal solutions extremely fast.

Dual ascent procedures have been used with considerable success to solve Uncapacitated Facility location problems (Erlenkotter [1978], Van Roy and Erlenkotter [1982]), Plant Location problems with additional side constraints (Guignard and Speilberg [1979]), Generalized Assignment

problems (Fisher et al. [1980]) and the Steiner tree problem (Wong [1983]). Recently, Magnanti et al. [1984] have solved to optimality comparatively large Uncapacitated Network Design problems using a composite algorithm in which a dual ascent procedure generates lower bounds; in their experience, the lower bounds obtained using dual ascent are extremely tight, and using these bounds, they were able to fix (to zero or one) 70% to 90% of the design variables before initiating the enumeration (Benders' decomposition) procedure. This experience suggests that dual ascent-based algorithms might be computationally attractive even for general Network Design problems with capacitated arcs and additional side constraints.

In the Network Design context, dual ascent solutions can be used for a variety of purposes besides generating lower bounds. For instance, it is possible to devise good heuristic procedures that use dual ascent solutions as their starting point. By using dual ascent solutions in conjunction with the LP complementary slackness conditions, we can often construct primal solutions that are almost feasible. When we apply a primal heuristic such as the Add/Drop procedure or the Exchange algorithm (see, for example, Billheimer and Gray [1973], Boorstyn and Frank [1977]) to these solutions, we obtain primal feasible solutions that are locally optimal. We elaborate on this possiblity later in this chapter. Dual ascent solutions can also be used to formulate Benders' cuts since they are feasible in the dual of the Benders' subproblem described earlier. Finally, as Magnanti and Wong [1984b] have shown, dual ascent solutions can be used for preprocessing and logical reduction of Network Design problems.

In this section, we first outline the general principles which form

the basis for various dual ascent procedures for the Network Design problem. For clarity and ease of exposition, we first describe the dual ascent framework for solving the Uncapacitated Network Design problem; later, we extend this framework to the capacitated case and to problems with additional side constraints. We illustrate the manner in which these basic ideas can be used to develop several alternative implementations of the Dual ascent procedure, highlighting the special structure of the NDP and methods for exploiting this structure. Next, we outline a few dual ascent-based heuristics and problem reduction procedures. Finally, we present limited computational results using one particular version of the dual ascent algorithm for the UNDP. The emphasis throughout this section is on generating good lower bounds for the NDP; we presume that the algorithms discussed here will be incorporated in a branch-and-bound or enumeration procedure if the problems must be solved to optimality.

4.3.1 A framework for Dual Ascent procedures for the UNDP:

Consider the Strong Arc-flow formulation [UNDSAF] (with scaled, unit demands for each commodity) of the Uncapacitated Network Design problem, which we repeat here for convenience.

[UNDSAF]

$$\text{Minimize} \quad \sum_{k} \sum_{(i,j)} c_{ij}^{k} x_{ij}^{k} + \sum_{(i,j)} F_{ij} y_{ij} \tag{7.1}$$

subject to

$$-\sum_{j} x_{ij}^{k} + \sum_{j} x_{ji}^{k} = \begin{cases} -1 & \text{if } i = O(k) \\ +1 & \text{if } i = D(k) \\ 0 & \text{otherwise} \end{cases} \qquad \text{all } k \in K \tag{7.2}$$

$$x_{ij}^{k} \leq y_{ij} \qquad \text{all } (i,j) \in A,\ k \in K \tag{7.3}$$

$$x_{ij}^{k},\ y_{ij} \geq 0 \qquad \text{all } (i,j) \in A,\ k \in K \tag{7.4a}$$

$$y_{ij} \text{ integer} \qquad \text{all } (i,j) \in A. \tag{7.4b}$$

Note that

(1) we have multiplied the flow conservation equations (1.2) of [NDAF] by -1; this convention will later enable us to interpret the dual problem conveniently, and

(2) assuming $F_{ij} \geq 0$ for all arcs $(i,j) \in A$, [UNDSAF] has an optimal solution in which all the design variables y_{ij} are less than or equal to 1. We have, therefore, omitted this constraint in the formulation [UNDAF].

The LP relaxation of [UNDSAF], denoted as [LUND], is obtained by relaxing the integer constraints (7.4b). Our final objective is to devise computationally efficient procedures for finding good solutions to the dual of this LP relaxation. Before doing so, however, we indicate how the LP relaxation of the weak formulation of the UNDP reduces to a series of Shortest path problems and can hence be solved very efficiently.

LP relaxation for the Weak Arc-flow formulation:

Recall that, in the weak formulation, the disaggregated forcing constraints (7.3) are replaced by the weaker constraints

$$\sum_k x_{ij}^k \leq |K| y_{ij} \qquad \text{for all } (i,j) \epsilon A. \tag{7.3a}$$

Let [LWUND] denote the LP relaxation of this weak Arc-flow formulation. Observe that, if $F_{ij} \geq 0$, for all $(i,j) \epsilon A$, [LWUND] has an optimal solution in which

$$y_{ij} = \sum_k x_{ij}^k / |K| \qquad \text{for all } (i,j) \epsilon A,$$

that is, constraints (7.3a) will be satisfied as equalities in the optimal LP solution. We can, therefore, eliminate the design variables y_{ij} from the formulation [LWUND] by substituting for y_{ij} in terms of x_{ij}^k (using this expression) in the objective function, and omitting the constraints (7.3a). Thus, we obtain the following equivalent formulation for [LWUND]:

$$\text{Minimize} \quad \sum_k \sum_{(i,j)} (c_{ij}^k + F_{ij}/|K|) x_{ij}^k$$

$$\text{subject to} \quad (7.2) \text{ and } (7.4a).$$

In other words, for each arc (i,j), the flow cost of each commodity is increased by the 'fixed cost per commodity'; the optimal solution of [LWUND] can, therefore, be obtained by solving $|K|$ shortest path problems, one for each commodity k, using the new coefficients of x_{ij}^k as arc lengths. Similarly, when the arcs are capacitated, the LP relaxation of the weak Arc-flow formulation of the CNDP reduces to a minimum-cost multicommodity flow problem in which each arc (i,j) has capacity B_{ij} and the cost per unit flow of commodity k on arc (i,j) is $(c_{ij}^k + F_{ij}/B_{ij})$.

The basic Dual Ascent strategy:

For the dual ascent procedures to be examined in this section, we use the Strong Arc-flow formulation since its LP relaxation gives much tighter lower bounds than [LWUND]. Let v_i^k, for $i \epsilon N$ and $k \epsilon K$, and w_{ij}^k, for $(i,j) \epsilon A$ and $k \epsilon K$, denote the dual variables corresponding to constraints (7.2) and (7.3) of [LUND], respectively. Then, the dual of [LUND], denoted as [DUND], is

[DUND]

$$\text{Minimize} \quad z_D = \sum_k v_{D(k)}^k \tag{8.1}$$

s.t.

$$v_j^k - v_i^k - w_{ij}^k \leq c_{ij}^k \qquad \text{all } (i,j) \epsilon A,\ k \epsilon K \tag{8.2}$$

$$\sum_k w_{ij}^k \leq F_{ij} \qquad \text{all } (i,j) \epsilon A \tag{8.3}$$

$$v_i^k \text{ unrestricted in sign} \qquad \text{all } i \epsilon N,\ k \epsilon K \tag{8.4a}$$

$$w_{ij}^k \geq 0 \qquad \text{all } (i,j) \epsilon A,\ k \epsilon K. \tag{8.4b}$$

Notice that, for each commodity $k \epsilon K$, one of the conservation equations (7.2) is redundant. Hence, one of the corresponding dual variables may be defined arbitrarily. In [DUND], we have set the variable $v_{D(k)}^k$ to zero for all commodities $k \epsilon K$. We let s_{ij} denote the slack in constraint (8.3) corresponding to arc (i,j).

The dual ascent strategy that we consider consists of alternately modifying the w_{ij}^k values (keeping the v values fixed) and the v_i^k values (keeping the w values fixed) in order to increase the dual objective function value monotonically. For this purpose, we must determine the best

assignment of v values for a given set of w values and vice versa. We next discuss two special features of the dual problem [DUND] that facilitate this assignment.

For any given set of values $\{w_{ij}^k\}$ that satisfy constraints (8.3), the optimal values of v_i^k (with respect to the given vector w) can be determined by solving the following subproblem [SP(w)]:

[SP(w)]

$$\text{Maximize} \quad \sum_k v_{D(k)}^k$$

subject to

$$v_j^k - v_i^k \leq c_{ij}^k + w_{ij}^k \qquad \text{for all } (i,j) \epsilon A \text{ and } k \epsilon K.$$

The optimal value of [SP(w)] is the best dual objective function value z_D that can be attained using the given values for w_{ij}^k. The problem [SP(w)] separates into $|K|$ independent subproblems, one for each commodity $k \epsilon K$; the k^{th} subproblem $[SP_k(w)]$ is just the dual of a Shortest path problem (from origin O(k) to destination D(k)) using $(c_{ij}^k + w_{ij}^k)$ as arc lengths. Thus, if we define the MODIFIED ARC LENGTHS e_{ij}^k as

$$e_{ij}^k = c_{ij}^k \qquad \text{for all } (i,j) \epsilon A \text{ and } k \epsilon K,$$

then

Property 1:

For any given set of values for w_{ij}^k satisfying (8.3), the 'best' value of z_D is the sum over all commodities $k \epsilon K$ of the shortest path length from origin O(k) to destination D(k), using e_{ij}^k as arc lengths.

Henceforth, unless otherwise stated, we will assume that all shortest paths (and lengths of paths) are determined using the Modified arc lengths e_{ij}^k. Observe that, if w_{ij}^k increases for some arc (i,j) and commodity k, the corresponding Modified arc length e_{ij}^k increases, thereby potentially increasing the shortest path length $v_{D(k)}^k$ and hence the dual objective

function value z_D. (z_D will necessarily increase if arc (i,j) belongs to all the shortest paths from O(k) to D(k)). Thus, Property 1 suggests the following method for increasing the dual objective function value monotonically.

Modify the values of one or more w_{ij}^k values iteratively so that:

(1) constraints (8.3) remain feasible, and

(2) the shortest path length $v_{D(k)}^k$ increases for at least one commodity $k \epsilon K$ at each stage, thus increasing z_D.

We can maintain feasibility in constraints (8.3) by ensuring that the slacks s_{ij}, for all arcs (i,j), are always non-negative. Thus, the w-modification step can be viewed as a process of 'using up' the current slack s_{ij} for increasing the w_{ij}^k values for one or more commodities $k \epsilon K$. All the dual ascent algorithms that we consider in this chapter use this procedure as the basis for the Ascent phase.

Let us now determine the best assignment of w values for any given set of 'feasible' v_i^k values. In this case, the dual solution can be 'completed' by setting

$$w_{ij}^k = \text{Max } \{0, v_j^k - v_i^k - c_{ij}^k\} \quad \text{for all } (i,j) \epsilon A \text{ and } k \epsilon K. \qquad (8.5)$$

In order to satisfy constraints (8.2) and the non-negativity constraints, w_{ij}^k must be greater than or equal to the expression on the right-hand side of equation (8.5). Therefore, the values of w_{ij}^k determined using (8.5) are the smallest possible, for the given set of v values. (Note: Since the given v values are feasible, these w values must be feasible in constraints (8.3).) Choosing w values in this manner ensures that the slacks s_{ij} are as large as possible (for the given v values), thereby providing greater flexibility for manipulating the w values in the next ascent step. Thus,

Property 2:

For any given set of 'feasible' values for v_i^k, the 'best' values of w_{ij}^k are determined using the equation:

$$w_{ij}^k = \text{Max } \{0,\ v_j^k - v_i^k - c_{ij}^k\} \quad \text{for all } (i,j) \epsilon A,\ k \epsilon K\ .$$

Just as we increase w values, in the ascent step, in order to increase the values of $v_{D(k)}^k$, we must select feasible v_i^k values for intermediate nodes i (i.e., nodes that are neither the origin nor destination for commodity k) so that $v_{D(k)}^k$ does not decrease and the w values, when updated using equation (8.5), are as small as possible.

It is possible to show that the LP dual [DUND] has an optimal solution that satisfies both properties 1 and 2. The two properties are complementary in the sense that they provide the 'best' set of w values for a given set of v values and vice versa. This observation suggests an ascent strategy that alternately modifies the w and v values in order to increase the dual objective function monotonically. The following procedure formalizes this idea.

The Basic procedure:

w-increasing step:

For one or more arcs $(i,j) \epsilon A$ with positive slack s_{ij}, use the slack to increase the w_{ij}^k values for one or more commodities $k \epsilon K$. Update the slacks s_{ij} correspondingly. If there are no arcs with positive slack, STOP.

v-updating step:

Update $v_{D(k)}^k$ (and z_D) as the length of the shortest path from O(k) to D(k), using the new values of the modified arc lengths $e_{ij}^k = c_{ij}^k + w_{ij}^k$. Set v_i^k, for intermediate nodes i, equal to any optimal dual solution of the SP problem $[SP_k(w)]$. If z_D does not increase in this step, STOP.

w-decreasing step:

Using the v values derived in Step 2, update
$w^k_{ij} \leftarrow \text{Max } \{0, v^k_j - v^k_i - c^k_{ij}\}$ for all $(i,j) \epsilon A$ and $k \epsilon K$,
and increase the slacks s_{ij} correspondingly. Return to Step 1.

The description of the first two steps in the basic procedure is quite ambiguous. For instance, the w-increasing step does not specify how to choose the arcs (i,j) and commodities k, for which the w^k_{ij} values must be increased. Similarly, the v-updating step does not specify which optimal dual (shortest path) solution to choose. We have deliberately left the specification incomplete so that this procedure can serve as a general framework for the various algorithms that we describe later. As we shall see next, Steps 1 and 2 can be implemented in several alternative ways, leading to significantly different algorithms. Also, this procedure does not include the possibility of increasing the dual objective function value by increasing w values for certain commodities and decreasing them for others. Later, while discussing the role of complementary slackness conditions in dual ascent procedures, we will give examples of situations in which such an ascent strategy is legitimate.

4.3.2 Alternative methods for the w-increasing step:

We now discuss several alternative ways to select the arcs (i,j) and commodities k for which w^k_{ij} must be increased, and to determine the amount of this increase, in each iteration of the basic procedure.

A naive approach is to increase w^k_{ij} by the same amount for all commodities, i.e., to update w^k_{ij} as

$$w^k_{ij} \leftarrow\!- \; w^k_{ij} + (s_{ij}/|K|) \quad \text{for all } (i,j)\epsilon A \text{ and } k\epsilon K \ ,$$

in each iteration of the basic procedure. This approach has two major disadvantages. First, for many arcs (i,j) (particularly those that do not lie on any shortest path for commodity k), increasing w^k_{ij} will not increase $v^k_{D(k)}$. Hence, the w^k_{ij} values corresponding to such arcs would necessarily be reduced to their original values in the w-decreasing step of the basic procedure, thus wasting considerable computational effort. Second, since this scheme does not preferentially allocate the slacks s_{ij} to increase those w^k_{ij}'s that might potentially increase $v^k_{D(k)}$, the rate of increase in the dual objective function could be slower than is otherwise possible. This procedure could be slow also because, as the number of iterations increases, some of the slacks s_{ij} might never become zero; rather, they may take on successively smaller, but positive values at each iteration. Correspondingly, the rate of increase of z_D would decrease at each iteration without becoming zero (and hence not stopping in either the w-increasing or v-updating step), and the procedure could, conceivably, continue for a very large number of iterations without stopping.

Therefore, our objective while devising methods for w-increasing step should be to use the slacks s_{ij} at each stage to selectively increase those w^k_{ij} values (and hence the corresponding modified arc lengths e^k_{ij}) that can potentially increase the shortest path lengths $v^k_{D(k)}$ (and hence z_D). For instance, if arc (i,j) lies on the current shortest path for commodity k, then w^k_{ij} is an obvious candidate for increase. (By <u>current shortest path for commodity k</u>, we mean the shortest path from O(k) to D(k), using e^k_{ij} as arc lengths, where modified arc lengths e^k_{ij} are computed using the <u>current</u> values of w^k_{ij}). The three w-increasing methods that we next discuss

attempt to achieve this objective in different ways.

The Next Shortest path method:

This method increases exactly one w^k_{ij} value in each iteration of the basic procedure. The method is based on the following observation. Consider any arc (i,j) with a positive slack s_{ij}. If arc (i,j) is not on any current shortest path for some commodity $k \epsilon K$, then increasing w^k_{ij} alone will not increase $v^k_{D(k)}$. On the other hand, if (i,j) belongs to one of the current shortest paths, say path P^{k*}, then, as w^k_{ij} increases, $v^k_{D(k)}$ will also increase as long as P^{k*} continues to be a shortest path for commodity k (using the higher value of w^k_{ij}). If w^k_{ij} is increased further (assuming that the slack s_{ij} is large enough to accomodate this increase), the NEXT PATH defined below becomes the new shortest path for commodity k, and $v^k_{D(k)}$ remains equal to the length of this Next path.

Definition 4.1:

> The NEXT PATH P^k_{ij} with respect to a given arc (i,j) and commodity k is the shortest path from O(k) to D(k) when arc (i,j) is removed from the network.

Let $l(P^k_{ij})$ denote the length of the Next path; notice that, since P^k_{ij} does not contain arc (i,j), $l(P^k_{ij})$ does not change as w^k_{ij} increases. Therefore, it is not worthwhile to increase w^k_{ij} beyond the point at which the current length of P^{k*} becomes equal to $l(P^k_{ij})$. If l^*_k denotes the value of $v^k_{D(k)}$ at the start of this iteration, then the maximum 'permissible' increase in w^k_{ij} is $(l(P^k_{ij}) - l^*_k)$. This difference is zero if, at the outset, commodity k has at least one current shortest path that does not contain arc (i,j)

(i.e., arc (i,j) does not belong to all the shortest paths for commodity k and P^k_{ij} is itself a current shortest path for commodity k that does not contain arc (i,j)); otherwise, this difference is strictly positive. Since w^k_{ij} cannot be increased by more than the current value of the slack s_{ij}, it should increase by at most

$$d^k_{ij} = \text{Min } \{ s_{ij}, [l(P^k_{ij}) - l^*_k] \}.$$

in this iteration. The strategy then is to select an arc (i,j) and commodity k for which this 'desirable' increase in w^k_{ij} is positive, and to increase w^k_{ij} by d^k_{ij}. With this strategy, the dual objective function value z_D strictly increases in each iteration. We can thus summarize this method for implementing the w-increasing step as follows:

If for some arc $(i,j) \epsilon A$, and commodity $k \epsilon K$, d^k_{ij} is positive, increase w^k_{ij} by d^k_{ij} and go to the v-updating step. Otherwise, Stop.

Discussion:

This procedure does not specify any particular order for examining the arcs and commodities. It is possible to devise several heuristic rules for specifying this order. For instance, we can adopt a 'greedy' approach, and increase the w^k_{ij} value corresponding to that arc (i,j) and commodity k with the maximum value of d^k_{ij}. Alternatively, we can scan the arc list in order of decreasing slacks s_{ij}, and choose the first arc for which d^k_{ij} is positive for some $k \epsilon K$. We can then use another rule for selecting the commodity k (for instance, choose the commodity with maximum d^k_{ij}). As is evident from these examples, there are multitude of 'reasonable' rules for selecting the arcs and commodities; none of the rules appear to be provably superior to the others and the choice must necessarily be based on computational experience.

Let us now examine the computational requirements for this method. Every time a value of d^k_{ij} is calculated, we must determine the length of the corresponding Next path P^k_{ij}. One way to determine this length is to re-solve the shortest path problem without arc (i,j). However, if all the current shortest path labels are stored from the previous execution of the v-updating step, it might be possible to devise a more efficient method, using these labels, for calculating $l(P^k_{ij})$. Also, we know that for arcs that do not belong to <u>all</u> the current shortest paths for commodity k, d^k_{ij} must be zero. Therefore, if we flag all such arcs in the v-updating step, then we need to consider only the remaining arcs; the number of such arcs would be considerably smaller, since, for each commodity, at most n arcs can belong to all the current shortest paths. In spite of these simplifications, the computational effort required by this method could be substantial, relative to other w-increasing methods, especially if the scanning rule at each iteration requires several evaluations of d^k_{ij} (in order to select the arc and commodity for which w^k_{ij} must be increased).

Another disadvantage of this method is that it increases just one value of w^k_{ij} in each iteration. As a result, many more iterations of the basic procedure would be required, compared to a strategy that increases several w^k_{ij} values simultaneously. More important, however, is the possibility that this method might not recognize certain opportunities for increasing the dual objective function value. Consider, for instance, the following scenario. For some commodity $k \epsilon K$, suppose there are two arcs, say (i1,j1) and (i2,j2), satisfying the following conditions:

- both (i1,j1) and (i2,j2) belong to one or more current shortest paths for commodity k,
- commodity k has several alternate current shortest paths, each of

which contains either arc (i1,j1) or arc (i2,j2), but not both; thus neither arc is on <u>all</u> the current shortest paths, so that $d^k_{i1,j1} = d^k_{i2,j2} = 0$, and

- both the slacks $s_{i1,j1}$ and $s_{i2,j2}$ are positive.

When we apply the Next Shortest path method in this situation, it will increase neither $w^k_{i1,j1}$ nor $w^k_{i2,j2}$, since $d^k_{i1,j1} = d^k_{i2,j2}$ neither arc belongs to <u>all</u> the shortest paths). However, since each current shortest path from O(k) to D(k) contains one of these two arcs, increasing both $w^k_{i1,j1}$ and $w^k_{i2,j2}$ <u>simultaneously</u> will increase $v^k_{D(k)}$ and hence z_D. It is possible to construct similar examples in which three or more 'parallel' arcs satisfy these three conditions. Such a situation will necessarily arise whenever any commodity has more than one current shortest path, a condition that is likely to occur very often. (Recall that every time w^k_{ij} increases by $d^k_{ij} = 1(P^k_{ij} - 1^*_k$, the Next path becomes an alternate shortest path.)

Another drawback of this method is that it ignores 'interaction' effects between dual variables since it increases just one w value at each iteration. For instance, increasing w^k_{ij} for some arc $(i,j)\epsilon A$ and $k\epsilon K$, decreases the values of $d^k_{i',j'}$ for all other arcs (i',j') that lie on the same current shortest path as (i,j). At the same time it reduces the slack s_{ij}, thereby decreasing $d^{k'}_{ij}$ for all other commodities k'. The Next shortest path method does not take into account such effects. Consequently, this method overlooks some opportunities for ascent.

Since the Next shortest path does not recognize instances in which simultaneous increases of w^k_{ij} values can increase the dual objective function value, it must be used in conjunction with another method that

identifies such possibilities. We later discuss how complementary slackness conditions can be used to identify w variables that must to be increased simultaneously for further ascent.

Interpreting a previous Dual Ascent procedure:

Before describing other w-increasing methods, we briefly indicate how the dual ascent algorithm for the UNDP proposed by Magnanti and Wong [1984b] can be interpreted as a particular implemetation of the Next shortest path method. (For consistency, we will consider a slightly modified version of their algorithm.) Magnanti and Wong's method also increases one w_{ij}^k value at a time. At each stage of the algorithm, for all commodities $k \epsilon K$, the method maintains a tree rooted at destination D(k). The arcs in this tree are those that lie on the current shortest path for commodity k and whose current slacks s_{ij} are zero. The algorithm initializes the dual solution and the data structures as follows:

- $w_{ij}^k = 0$, $s_{ij} = F_{ij}$ for all $(i,j) \epsilon A$ and $k \epsilon K$,
- v_i^k = length of the shortest path from O(k) to node i, using c_{ij}^k as arc lengths, for all $i \epsilon N$ and $k \epsilon K$, and
- the rooted tree for each commodity $k \epsilon K$, consists of only the destination node D(k).

At each iteration, the method chooses the next commodity k in the given list of commodities as the commodity for which some w_{ij}^k value must be increased. In other words, the algorithm sequentially scans the commodity list, selecting successive commodities in successive iterations. If the rooted tree (corresponding to commodity k) contains a path from O(k) to D(k), then the procedure does not increase any w value corresponding to this commodity; instead it considers the next commodity in the list. If, on the other hand, O(k) is not reachable from D(k) using the arcs of the

rooted tree, the procedure identifies the following subset A_k of arcs that do not belong to the rooted tree:

A_k = the set of all arcs that belong to the current shortest paths for commodity k and that are incident to one of the leaves of the rooted tree corresponding to commodity k.

A_k is used as the list of candidate arcs, one of whose w^k_{ij} value must be increased. From this subset, the algorithm identifies the arc (i',j') with the smallest value of d^k_{ij} (as defined before) and increases the corresponding w value $w^k_{i',j'}$ by $d^k_{i',j'}$. The values of v^k_i corresponding to nodes i that lie on the path from j' to D(k) (j' and D(k)) included) are then increased by $d^k_{i',j'}$, thus completing the v-updating step as well. Arc (i',j') is added to the rooted tree if, at the end of this iteration, the slack $s_{i',j'}$ reduces to zero. Because of the manner in which the method updates the v values and the rooted tree in each iteration, it can determine the values of d^k_{ij} for arcs (i,j) belonging to the set A_k very efficiently, without resorting to any shortest path calculations. The algorithm stops, when, for all commodities $k \epsilon K$, O(k) is reachable from D(k) along the arcs of the corresponding rooted tree. Observe that, in essence the algorithm attempts to 'grow' a tree rooted at D(k); all the arcs in this tree are shortest path arcs with zero slacks. At each iteration, by choosing an arc belonging to the subset A_k, the algorithm extends the tree further towards the origin O(k). Thus, the basic principles used in this procedure are the same as those for the Next shortest path method. By using appropriate data structures, and specifying the order in which w^k_{ij} values must be increased, Magnanti and Wong [1984b] have obtained an efficient implementation of the algorithm. They have also suggested how this procedure can be used in an iterative manner in conjunction with the complementary slackness conditions and a heuristic algorithm, in order to

obtain better dual solutions. We will later discuss some of these possibilities.

The Simultaneous w-increasing method:

Our objective in the w-increasing step of the basic procedure is to increase those w_{ij}^k values that can potentially increase $v_{D(k)}^k$. The Next shortest path method identifies, and increases, one such value at each iteration of the basic procedure. We have already discussed the drawbacks of increasing one w value at a time. The Simultaneous increase method that we consider now, increases several w values simultaneously at each iteration of the basic procedure. For every arc (i,j) of the network, this method attempts to identify a set of commodities for which increasing w_{ij}^k would increase the shortest path lengths $v_{D(k)}^k$. Since identifying this actual set of commodities might be computationally expensive, the algorithm instead finds a superset K_{ij} that contains this set. In other words, K_{ij} consists of commodities for which increasing w_{ij}^k has the potential to increase the shortest path lengths. The method then simultaneously allocates the slack s_{ij} corresponding to arc (i,j) equally to all the commodities belonging to the set K_{ij}, i.e., it increases w_{ij}^k by $(s_{ij}/|K_{ij}|)$ for all $k \epsilon K_{ij}$ and all arcs $(i,j) \epsilon A$.

First, note that, for any given arc (i,j), we can easily identify the commodities for which increasing w_{ij}^k will not increase their current shortest path distance. We use three conditions to make this identification:

(C1) All the arcs on some current shortest path from O(k) to D(k) (using

the modified arc lengths e^k_{ij}) have zero slack.

Since w values for arcs on this path cannot increase further (because the slacks are zero), $v^k_{D(k)}$ cannot increase in the w-increasing step.

(C2) All the arcs on some current shortest path from O(k) to node j have zero slack.

Again, the shortest path distance to node j cannot be increased in the w-increasing step; hence, increasing w^k_{ij} will not increase $v^k_{D(k)}$.

(C3) $v^k_j - v^k_i < c^k_{ij} + e^k_{ij}$.

In this case, arc (i,j) does not belong to any shortest path for commodity k; therefore, increasing w^k_{ij} (and hence the arc's cost e^k_{ij}) will not affect the shortest path distance $v^k_{D(k)}$.

We let K_{ij} be the set of commodities that do <u>not</u> satisfy (C1) - (C3). This set may contain commodities for which increasing w^k_{ij} does not increase $v^k_{D(k)}$, since criterion (C3) employs a necessary, but not sufficient, condition for arc (i,j) to lie on the the current shortest path. For instance, if for all nodes i the current value of v^k_i is the shortest path distance from O(k) to node i, then all arcs belonging to the shortest path tree rooted at O(k) violate condition (C3). However, many of these arcs may not belong to any current shortest path for commodity k.

The Simultaneous method then executes the w-increasing step of the basic procedure as follows:

For each arc (i,j) with $s_{ij} > 0$ and K_{ij} nonempty, set

$$w^k_{ij} \leftarrow w^k_{ij} + (s_{ij}/|K_{ij}) \quad \text{for all } k \epsilon K_{ij}.$$

This procedure has several noteworthy characteristics.

(1) Let K' be the set of commodities that violate condition (C1). It is possible to show that, after the w values are increased using this method, the value of $v^k_{D(k)}$ (when calculated in the v-updating step) must necessarily increase for <u>each</u> commodity k in the set K' (while $v^k_{D(k)}$ remains the same for all commodities not belonging to the set K').

(2) The Simultaneous method terminates only when <u>all</u> commodities satisfy condition (C1). Thus, at termination, each commodity $k \epsilon K$ has at least one shortest path (using the modified arc lengths e^k_{ij}) from O(k) to D(k) all of whose arcs have zero slack. As we shall see later, because of this property, we can use the dual solution derived by this method to construct a primal solution that satisfies two out of the three complementary slackness conditions. In contrast, the Next shortest path method (with the exception of Magnanti and Wong's [1984b] algorithm) does not ensure that (C1) is satisfied for each commodity k.

(3) Since this method increases several w values simultaneously, most of the instances that we discussed earlier, in which more than one w value must be changed to increase z_D, do not arise here. There are, however, other opportunities for ascent that this method (as well as the previous method) does not recognize. In particular, the method does not tradeoff increases in some w values against decreases in others.

The Simultaneous w-increasing method has several variants. For instance, if, in the v-updating step, we flag the arcs that belong to <u>all</u> the shortest paths for each commodity, then subset of 'potential' commodities K_{ij} can be restricted even further (and condition (C3) need not be applied). Also, for each arc (i,j), we have allocated the slack s_{ij} <u>equally</u> to all the commodities in K_{ij}. Other non-uniform allocation rules may be more appropriate and justifiable. At the end of this chapter we report on some computational experience for a dual ascent algorithm using this implementation of the w-increasing step. Next, we consider a more comprehensive and systematic method for increasing several w values

simultaneously in the first step of the basic procedure.

The Composite method:

In this method, we pose the problem of allocating the slacks s_{ij} to the commodities k for increasing w^k_{ij} as a series of optimization problems. We show how each of these problems can be transformed into an equivalent Minimum-cost flow problem defined over a network with a special structure, and can hence be solved very efficiently. We also relate this model to some optimization problems that arise in the context of network reliability and PERT/CPM networks.

Unlike the previous method, the Composite method considers the commodities sequentially, changing one or more w^k_{ij} values for one commodity k at each stage. The method is based on the following principle. Suppose, for the current commodity k, we have decided to 'allocate' at most $b^k_{ij} \geq 0$ of the slack s_{ij} for increasing w^k_{ij} from its current value. For instance, we may set b^k_{ij} equal to s_{ij}, for all arcs $(i,j) \epsilon A$; alternatively, since we know that it is not worthwhile to increase w^k_{ij} by more than d^k_{ij} (defined in the Next shortest path method), we can set $b^k_{ij} = d^k_{ij}$ for all $(i,j) \epsilon A$. Then, the maximum possible value of $v^k_{D(k)}$, subject to this allocation limit is $l^*_k(b^k)$, the length of the shortest path from O(k) to D(k) using $(e^k_{ij} + b^k_{ij})$ as arc lengths. Since we are interested in maximizing z_D, we might wish to increase the w^k_{ij}'s so that $v^k_{D(k)}$ becomes equal to this upper limit $l^*_k(b^k)$; alternatively, we could choose some value $l^k_O \leq l^*_k(b^k)$, and require that, at the end of the w-increasing step, $v^k_{D(k)}$ must be greater than or equal to l^k_O. Since the first method is a special case of the second, we

will consider only the latter method. Suppose our subsidiary objective is to use up as little of the allocation limit $\{b^k_{ij}\}$ as possible, yet increasing $v^k_{D(k)}$ to l^k_O. This objective is attractive because if we use up less of the slack for increasing the w values corresponding to commodity k, then the larger residual slack can be subsequently used to increase $v^{k'}_{D(k')}$, for some other commodity k'. The allocation problem for commodity k can now be formulated as follows:

Formulating the Allocation problem:

Let r^k_{ij}, for each arc $(i,j) \epsilon A$, be the decision variable denoting the amount by which w^k_{ij} must be increased. Then, the Allocation problem $[AP^k]$ for any given commodity k is

$[AP^k]$

$$\text{Minimize} \quad \sum_{(i,j)} r^k_{ij} \tag{9.1}$$

subject to

$$v^k_j - v^k_i \leq e^k_{ij} + r^k_{ij} \qquad \text{for all } (i,j) \epsilon A \tag{9.2}$$

$$r^k_{ij} \leq b^k_{ij} \qquad \text{for all } (i,j) \epsilon A \tag{9.3}$$

$$v^k_{D(k)} - v^k_{O(k)} \geq l^k_O. \tag{9.4}$$

Constraints (9.2) ensure that the optimal values of v^k_i in the subproblem $[AP^k]$ satisfy the dual constraints (8.2) of [DUND] after the w^k_{ij} values have increased by r^k_{ij}. Constraints (9.3) are the upper bounds that we imposed on the increases in the w values. Constraint (9.4) ensures that the length of the shortest path from O(k) to D(k), after updating the w values, is greater than or equal to l^k_O. Notice that, while we had set

(without loss of generality) the variable $v^k_{O(k)}$ to zero in [DUND], we have included it in the formulation [AP^k]; this convention helps us to interpret the dual of this problem. Also, we have not imposed any non-negativity constraints on the 'perturbations' r^k_{ij}, i.e., we now consider both increases and decreases in the w values. However, if r^k_{ij} is negative, it should be less than or equal to the current value w^k_{ij} in order to ensure that the new value of w^k_{ij} is non-negative; we will indicate later how this lower bound on r^k_{ij} can be easily incorporated in the solution procedure. Since the Allocation problem determines the new values for both the w^k_{ij} and the v^k_i variables, we refer to this method as the Composite procedure; it combines all three steps of the basic procedure.

A network flow reformulation of the Allocation problem [AP^k]:

Let g^k_{ij} and h^k_{ij}, for $(i,j)\epsilon A$, and g^k_O be the dual variables corresponding to constraints (9.2), (9.3) and (9.4), respectively. Then, the dual of [AP^k] is

[DAP^k]

$$\text{Minimize} \quad \sum_{(i,j)} e^k_{ij} g^k_{ij} + \sum_{(i,j)} b^k_{ij} h^k_{ij} + 1^k_O g^k_O$$

subject to

$$-\sum_j g^k_{ij} + \sum_j g^k_{ji} = \begin{cases} +g^k_O & \text{if } i = D(k) \\ -g^k_O & \text{if } i = D(k) \\ 0 & \text{otherwise} \end{cases} \qquad (10.2)$$

$$-g^k_{ij} + h^k_{ij} = -1 \qquad \text{for all } (i,j)\epsilon A \qquad (10.3)$$

$$g^k_{ij},\ h^k_{ij} \geq 0, \qquad \text{for all } (i,j)\epsilon A. \qquad (10.4)$$

Since the variable g^k_O is unrestricted in sign, we can replace it by the

difference $(g_{O}^{k+} - g_{O}^{k-})$ between two non-negative variables g_{O}^{k+} and g_{O}^{k-}. These two variables can now be interpreted as flows along a new 'direct' arc (O(k),D(k)) and a new 'return' arc (D(k),O(k)), respectively. The cost per unit of flow along these two arcs is 1_{O}^{k} and -1_{O}^{k}. Thus, if we define the new set of arcs A' as

$$A' = A \cup \{(O(k),D(k)),(D(k),O(k))\},$$

with $e_{O(k)D(k)} = 1_{O}^{k}$ and $e_{D(k),O(k)} = -1_{O}^{k}$,

then constraints (10.2) can be rewritten as

$$\sum_{j} g_{ij}^{k} - \sum_{j} g_{ji}^{k} = 0 \quad \text{for all } i \epsilon N, \tag{10.2a}$$

where the indices of summation in the left-hand side are over all arcs (i,j) and (j,i), respectively, belonging to the Augmented arc set A'.

We can now convert the revised dual problem into a network flow problem by performing a series of row operations. For each arc $(i,j) \epsilon A$, subtract the $(i,j)^{th}$ constraint (10.3) from the revised constraint (10.2a) corresponding to node j. After this manipulation, the dual problem becomes

[DAP'^{k}]

$$\text{Minimize} \quad \sum_{(i,j) \epsilon A'} e_{ij}^{k} g_{ij}^{k} + \sum_{(i,j) \epsilon A} b_{ij}^{k} h_{ij}^{k} \tag{10.5}$$

subject to

$$\sum_{j} g_{ij}^{k} - \sum_{j} g_{ji}^{k} = -d_{i} \qquad \text{for all } i \epsilon N \tag{10.6}$$

$$- g_{ij}^{k} + h_{ij}^{k} = -1 \qquad \text{for all } (i,j) \epsilon A \tag{10.7}$$

$$g_{ij}^{k},\ h_{ij}^{k} \geq 0 \qquad \text{for all } (i,j) \epsilon A, \tag{10.8}$$

where d_i is the In-degree of node i in the original graph G .

Observe that every column in the constraint matrix has exactly one +1 entry and one -1; all other elements are zero. Therefore, [DAP'^{k}] is a network

flow problem defined over the following EQUIVALENT network G":(N",A"):

- The node set N" of G" consists of

  (a) the original nodes i of the given network G, and

  (b) a node, denoted as [ij], corresponding to each arc (i,j) in the original graph .

- The arc set A" of G" consists of

  (a) the two arcs (O(k),D(k)) and (D(k),O(k)), with costs 1_O^k and -1_O^k respectively,

  (b) arcs of the form (i,[ij]) corresponding to every arc (i,j)ϵA, with cost e_{ij}^k; the variables g_{ij}^k denote the flow on these arcs, and

  (c) arcs of the form ([ij],j) corresponding to every arc (i,j)ϵA, with cost b_{ij}^k; the variables h_{ij}^k denote the flow on these arcs.

In the equivalent network problem, each node [ij], for (i,j)ϵA, has a supply of 1 unit, while every node iϵN has a demand of d_i units. Figure 17 is an example of an original network and the equivalent problem. (As an aside, we note that the transformation we have used is similar to the traditional transformation for converting capacitated network flow problems to equivalent uncapacitated problems.)

[DAP'^{k}] is thus a transhipment problem defined over the Equivalent network G". However, this transshipment problem does not contain any capacity constraints. Hence, [DAP'^{k}] can be solved efficiently in the following manner:

(1) Find, in the Equivalent graph, the shortest distance from each node [ij] of N" to all the nodes p of N" that correspond to nodes of the original graph. Denote this distance as $L_{ij,p}$.

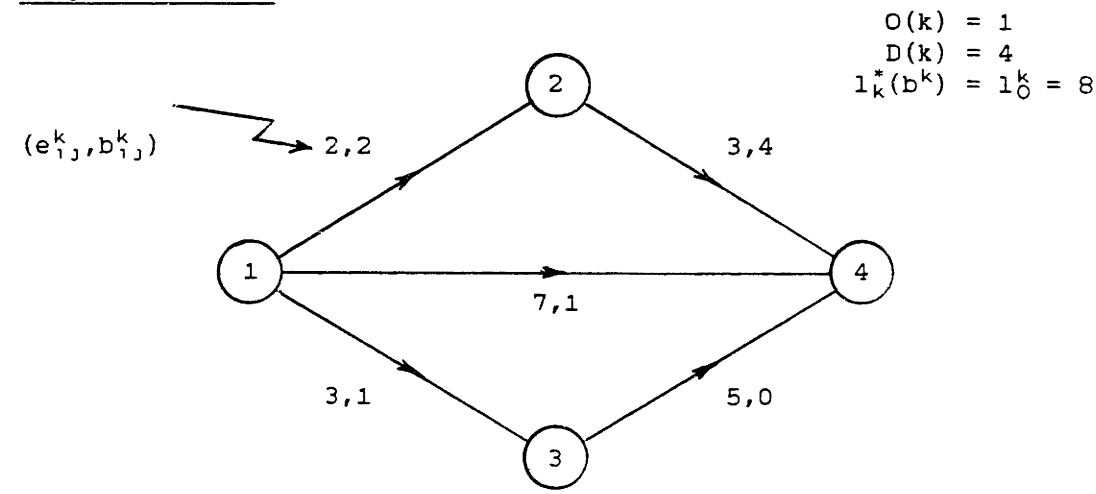
(2) Construct the Bipartite graph G_B (N_B,A_B) with

$$N_B = N_1 \cup N_2 ,$$

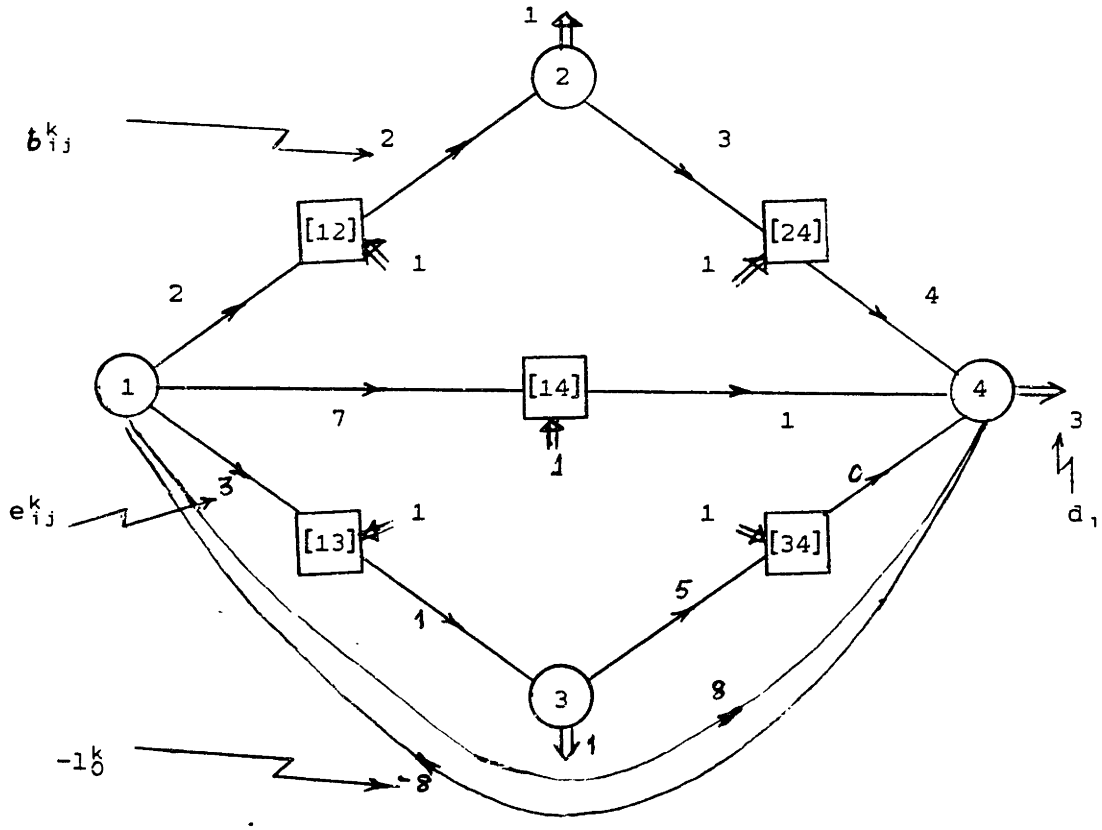
where $N_1 = \{[ij]: (i,j)\epsilon A\}$, and $N_2 = \{p: p\epsilon N\}$, and

Figure 17: Example of an Equivalent Network

Original Network:



Equivalent Network:



$$A_B = \{([ij],p):\ [ij] \epsilon N_1,\ p \epsilon N_2\}\ .$$

The cost of flow on arc $([ij],p)$ of A_B is set equal to $L_{ij,p}$. Each node of N_1 has a supply of 1 unit, while each node of N_2 has a demand of d_p units.

[DAP'^{k}] is then equivalent to the Hitchcock Transportation problem defined over the bipartite graph G_B. Observe that the number of nodes in G_B is the same as that in the Equivalent graph G".

This problem has several interesting characteristics.

(a) The 'return' arc (D(k),O(k)) is the only arc in the network G":(N",A") with a negative cost ($= -l^k_O$). Now, the shortest path from O(k) to D(k) in G" has length $l^*_k(b^k)$ (which was defined earlier to be the length of the shortest path for commodity k using $(e^k_{ij} + b^k_{ij})$ as arc lengths), and by definition, $l^k_O \leq l^*_k(b^k)$; therefore, G" does not contain any negative cycles.

(b) In the first part of the procedure for constructing the bipartite graph G_B, we must find the shortest distance $L_{ij,p}$ from each node [ij] to all nodes p of G". These distances can be found without constructing the Equivalent network G". Note that, in G", the only arc incident <u>from</u> every node [ij] is the arc $([ij],j)$, with cost b^k_{ij}. Hence, the shortest path from node [ij] to node p must contain arc $([ij],j)$ for any node $p \epsilon N$. Thus, $L_{ij,p}$ can be determined as follows:

> Add the 'direct' and 'return' arcs (O(k),D(k)) and (D(k),O(k)) to the original network G, with costs l^k_O and $-l^k_O$, respectively. Using $(e^k_{ij} + b^k_{ij})$ as the lengths for all the original arcs (i,j) of G, determine the length $L'_{j,p}$ of the shortest path from node j to node p of the Augmented graph. Then, the length of the shortest path from node [ij] to node p in G" is
>
> $$L_{ij,p} = L'_{j,p} + b_{ij}\ .$$

(c) A feasible solution to the Hitchcock transportation problem is to send 1 unit of flow from node [ij] to node j, for all $(i,j) \epsilon A$. The cost of this solution is

$$\sum_{(i,j) \epsilon A} b^k_{ij}\ .$$

Thus, this value is an upper bound on the optimal value of [DAP'^{k}] (and hence of [DAP^k]); this upper bound is quite obvious from the original problem itself (since setting $r^k_{ij} = b^k_{ij}$, for all $(i,j) \epsilon A$, is feasible in [AP^k]). If the optimal value of [DAP'^{k}] is strictly less than this value, then at least one unit of flow must be routed via the 'return' arc (D(k),O(k)) which has a negative cost.

Let us recapitulate the discussions so far. We first cast the problem

of 'allocating' the slacks s_{ij} for increasing the w^k_{ij}'s corresponding to a given commodity k as the optimization problem $[AP^k]$. We then showed that the dual of this problem can be transformed into a transshipment problem defined over an Equivalent graph G", which in turn is equivalent to a transportation problem over a bipartite graph G_B. We also showed that for determining the parameters of this transportation problem, we need not explicitly construct the Equivalent graph G". The optimal values of the decision variables r^k_{ij} and v^k_i in the original problem $[AP^k]$ can be obtained from the optimal dual solution of the equivalent transportation problem. In particular, r^k_{ij}, the amount by which w^k_{ij} must be increased, is the optimal dual value associated with the node [ij] of the bipartite graph G_B.

An enhancement of the Allocation problem $[AP^k]$:

We will now briefly indicate how the solution procedure can be modified to handle the additional constraints

$$r^k_{ij} \geq - w^k_{ij} \qquad \text{for all } (i,j) \epsilon A \ , \tag{9.3a}$$

in the Allocation problem $[AP^k]$. These constraints ensure that the w values do not become negative after the Composite method updates them. (Recall that, in the formulation $[AP^k]$, the variables r^k_{ij} are unrestricted in sign; thus, we allow both increases and decreases in the current w values). When these lower bounding constraints (9.3a) are added to $[AP^k]$, we must make the following changes.

- The dual problem $[DAP^k]$ will now contain additional variables, say h'^k_{ij}, for all $(i,j) \epsilon A$, each with an objective function coefficient of $- w^k_{ij}$. For convenience, we will negate these additional variables; then, all occurences of h^k_{ij} in the constraint set of the dual $[DAP^k]$ must be replaced by the difference $(h^k_{ij} - h'^k_{ij})$, and the objective function will now contain the additional term $+ \Sigma_{(i,j)} w^k_{ij} h'^k_{ij}$.
- In the Equivalent network corresponding to this modified problem, we must add the 'reverse' arc (j,[ij]), for all $(i,j) \epsilon A$; the flow on

this arc corresponds to the variable h'^{k}_{ij} and the cost per unit flow on this arc is w^{k}_{ij}.

The computational procedure discussed earlier applies to this modified version of the allocation problem as well. Also, in the allocation problem $[AP^k]$, we assigned a weight of 1 (in the objective function) to all the w-perturbations r^{k}_{ij}. However, the solution procedure can be easily adapted to situations in which each of these variables is assigned a different weight, say q^{k}_{ij}. For instance, by making these weights inversely proportional to the current slacks, we can model the requirement that w^{k}_{ij} must be preferentially increased for arcs with larger slack. The algorithm is easily modified to handle this variation: in the Equivalent network representation of the dual problem, the supply at each node [ij] becomes q^{k}_{ij}, and the demand at each node p (corresponding to the node p of the original graph) becomes $\Sigma_j\, q^{k}_{jp}$.

In summary, the Composite method implements the w- and v-updating steps of the basic procedure simultaneously by solving the allocation problem $[AP^k]$ once for each commodity $k \epsilon K$ in every iteration. We now present a formal statement of this method.

For each commodity $k \epsilon K$,

Solve the corresponding allocation problem $[AP^k]$ with some suitably chosen parameters

$$b^{k}_{ij} \leq s_{ij} \quad \text{for all } (i,j) \epsilon A, \text{ and}$$

$$l^{k}_{O} \leq l^{*}_{k}(b^{k}), \text{ the length of the shortest path from } O(k) \text{ to } D(k) \text{ using } (e^{k}_{ij}+b^{k}_{ij}) \text{ as arc lengths.}$$

Update $w^{k}_{ij} \leftarrow w^{k}_{ij} + r^{k}_{ij}$ for all $(i,j) \epsilon A$, where r^{k}_{ij} is the optimal solution of $[AP^k]$.

Update $s_{ij} \leftarrow s_{ij} - r^{k}_{ij}$ for all $(i,j) \epsilon A$.

Again, we might consider alternative rules for determining the order in which the commodities are considered. There are also a large number of possibilities for choosing the values of the parameters b^k_{ij} and l^k_0 as well as the weights q^k_{ij} that are assigned to the w-perturbation variables r^k_{ij} of $[AP^k]$.

Other applications:

Before discussing alternative methods for implementing the v-updating step of the basic dual ascent procedure, we consider some other contexts in which optimization problems similar to the Allocation problem $[AP^k]$ arise. This problem is, perhaps, best described as the "Minimum-cost Shortest path Expansion problem", since the objective here is to determine the least cost plan that increases the arc lengths in order to ensure that the length of the shortest path, using the revised arc lengths, is greater than a given threshold value. The upper bounds b_{ij} may be viewed as restrictions, imposed by the available resources, on the amount of increase in the length of arc (i,j), and the coefficients q_{ij} represent the cost per unit of the resource required to expand the length of arc (i,j). Two obvious contexts in which such a model would be useful are network reliability analysis and project management.

Consider first the project management problem. Given the time required to complete each activity and the precedence relationships between these activities, the PERT technique determines the 'critical path' (and hence the earliest project completion time) by identifying the longest path in an acyclic directed network, in which arcs represent activities and nodes correspond to events or milestones. Suppose each activity can be

'crashed' (i.e., the time required to complete the activity can be reduced) to a limited extent by employing some additional resources. The problem, then, is to determine the 'crashing' plan that completes the project within a given deadline using the minimum amount of resources (or at minimum additional cost). Since the PERT network is acyclic, we can convert this problem into a Minimum-cost shortest path expansion problem by assigning the negative of the time required to complete activity (i,j) as the current length for arc (i,j). Notice, however, that this method cannot be applied when several activities share a common scarce resource; in this case the resource limitation is a 'bundle' constraint rather than the simple upper bounds (9.3), and the problem becomes much harder to solve. Fulkerson [1961], Elmaghraby [1977], Moder et al. [1983], and others have considered similar optimization problems in the context of resource constrained project crashing.

Let us now consider a problem in network reliability. Suppose the arcs (i,j) of a given network fail independently with probability p_{ij}. Then, the least reliable path in the network between any two nodes s and t is the shortest path from s to t, using $\log(p_{ij})$ as arc lengths. Suppose the 'reliability' of each arc (i.e., the probability p_{ij}) can be increased by committing some additional limited resources that are not shared by more than one arc. The problem of finding the optimal plan for deploying these resources in order to increase the reliability of the least reliable path beyond a certain threshold level at minimum cost is the Minimum-cost Shortest path expansion problem. Golden [1978] has outlined a closely-related problem, where the objective is to increase the arc lengths (or transportation costs) in an adversary's network in order to increase

the length of the shortest supply route from a given origin to a destination. This model, however, places no upper bounds on the permissible increase in the arc lengths. Golden [1978] showed that the dual of this problem, without any row operations and transformations of the type that we discussed, is a Minimum-cost network flow problem. Finally, we mention that it is possible to adapt the algorithm for solving problem $[AP^k]$ to certain special situations in which the 'expansion' resources are shared by more than one arc.

In conclusion, in this section we have discussed three alternative methods for implementing the w-increasing step of the basic dual ascent procedure. The methods that we considered illustrate the tradeoff between increasing the dual objective function value z_D as much as possible by modifying the w variables and the computational effort required to achieve this increase. The Composite method, for instance, is the most sophisticated, but its computational requirements are significantly higher than those of the other two methods. Next, we consider alternative ways for updating the v values in the second step of the basic dual ascent procedure.

4.3.3 Alternative methods for the v-updating step:

After increasing the w values in the first step of the basic procedure, we must correspondingly update the v values. In particular, $v^k_{D(k)}$ must be increased for all commodities whose shortest path lengths increased in the preceding w-increasing step, and the v values for

intermediate nodes must be set equal to some optimal shortest path dual solution. If both the w and v values are updated simultaneously, as in the Composite method, this second step is not necessary. As was the case for the w-increasing step, there are several alternative ways to implement this v-updating step. Recall that, for any given set of w^k_{ij} values, the corresponding 'optimal' values of the variables v^k_i, for each commodity $k \epsilon K$, can be obtained by solving the subproblem $[SP^k(w)]$ that we discussed in Section 4.3.1 (in the context of Property 1). This subproblem is the dual of the shortest path problem from O(k) to D(k) with $e^k_{ij} = c^k_{ij} + w^k_{ij}$ as arc lengths, and will typically have multiple optimal solutions. The various v-updating methods that we discuss in this section identify different optimal values for the variables v^k_i. We first describe two simple methods and discuss their relative merits. This discussion motivates one criterion, that we have already employed in the Composite method, for choosing between the different shortest path dual optimal solutions. We then describe some enhancements of the simpler methods that incorporate this criterion. Before describing the different methods, we note that, for any given set of w values, the optimal value of v^k_i for certain nodes $i \epsilon N$ are uniquely defined. For instance, $v^k_{D(k)}$ must necessarily be equal to the length of the shortest path (using e^k_{ij} as arc lengths) from O(k) to D(k). Similarly, it is possible to show that for every node i that lies on at least one shortest path (from O(k) to D(k)) for commodity k, v^k_i must be equal to the shortest distance from O(k) to node i. Therefore, the only flexibility in the v-updating step is in the values that are assigned to v^k_i for 'intermediate' nodes i that do not lie on <u>any</u> shortest path for commodity k.

The forward shortest path method:

This approach sets v_i^k, for each commodity $k \epsilon K$ and all nodes $i \epsilon N$, equal to the shortest path distance from the origin O(k) to node i (using the modified arc lengths e_{ij}^k). As is well-known, these values of v_i^k are optimal in the shortest path dual problem $[SP^k(w)]$.

The reverse shortest path method:

Unlike the forward method, this method first computes, for each commodity k, the shortest path distance l_i^k (using e_{ij}^k as arc lengths) from every node i to the destination D(k). The method then updates the v values as

$$v_i^k \text{ <-- } l_{O(k)}^k - l_i^k \quad \text{for all } i \epsilon N \text{ and } k \epsilon K.$$

We refer to this method as the reverse shortest path method because it uses shortest path distances to the destination, rather than from the origin. We can easily show that this assignment of v values solves the subproblem $[SP^k(w)]$ optimally. First, notice that $l_{O(k)}^k$ must be equal to the length of the shortest path from O(k) to D(k) in the original graph G. Since the method sets $v_{O(k)}^k$ equal to $l_{O(k)}^k$, the objective function value corresponding to this solution is equal to the optimal value of $[SP^k(w)]$. We, therefore, must show only that the vector v is feasible in $[SP^k(w)]$. Since l_i^k is the shortest path distance from node i to node D(k), it must satisfy the condition

$$l_i^k - l_j^k \leq e_{ij}^k = c_{ij}^k + w_{ij}^k \quad \text{for all } (i,j) \epsilon A.$$

Therefore,

$$v_j^k - v_i^k = -(l_j^k - l_i^k)$$

$$\leq c^k_{ij} + w^k_{ij} \qquad \text{for all } (i,j) \epsilon A.$$

Hence, the solution $\{v^k_i\}$ determined by this method is feasible in $[SP^k(w)]$. Notice that, for intermediate nodes i that do not lie on the shortest path for commodity k, the v values obtained using this method are different from those obtained using the forward shortest path method. In fact, we can show that the latter values will always be greater than or equal to the former.

The advantage of both these methods is their simplicity and ease of implementation. However, both the methods lack the 'look-ahead' feature that we describe below: Recall that, after the v values are fixed in the v-updating step of the basic procedure, the w values are updated in the next step using the equation

$$w^k_{ij} = \text{Max } \{0, v^k_j - v^k_i - c^k_{ij}\} \quad \text{for all } (i,j) \epsilon A \text{ and } k \epsilon K. \qquad (11.1)$$

And, as discussed earlier, we would like the updated values of w^k_{ij} to be as small as possible (subject to the condition that z_D does not decrease), so that the slacks s_{ij} are maximized, thus providing greater scope for improving z_D during the next iteration. Therefore, we must choose, from among the set of alternate shortest path dual solutions, the solution $\{v^k_i\}$ that minimizes the w^k_{ij} values derived using equation (11.1). Neither the forward nor the reverse shortest path method uses such a selection criterion. We now discuss an enhancement of the forward shortest path method that attempts to heuristically adjust the v values so that the w values, when updated using equation (11.1), become smaller. We can develop a similar 'second stage' for the reverse shortest path method. Again, as we shall see, this improvement entails additional computational effort.

The v-improvement method:

Suppose the v values are first updated using the forward shortest path method, i.e., the current value of v_i^k, for all nodes $i \epsilon N$ and commodities $k \epsilon K$, is the shortest path distance from O(k) to node i. For the moment, we will assume that the w values have also been updated using equation (11.1); later, we will show that this intermediate w-updating step is not necessary. The v-improvement procedure that we next describe uses the following idea. Suppose, for some 'intermediate' node i and commodity k, $v_j^k - v_i^k < c_{ij}^k$ for all arcs (i,j) that are incident from node i. (Obviously, w_{ij}^k must be zero for all such arcs). Then, the value of v_i^k can decrease without violating the feasibility of the solution or increasing any w value. Suppose, in the current solution, $w_{li}^k > 0$ for some arc (l,i) incident to node i; then, as v_i^k decreases, w_{li}^k can also decrease (since $w_{li}^k \leftarrow \text{Max } \{0, v_l^k - v_i^k - c_{li}^k\}$). Thus, our objective is to identify such intermediate nodes i and to determine the maximum possible reduction in v_i^k. This problem can be formulated as follows: Let r_i^k be the amount by which v_i^k can decrease and let $u_i^k = v_i^k - r_i^k$ be the updated value of v_i^k after this stage. Then, we want to

$$\text{Minimize} \quad u_i^k = v_i^k - r_i^k$$

subject to

$$u_j^k - u_i^k \leq c_{ij}^k + w_{ij}^k \qquad \text{for all } (i,j) \epsilon A$$

$$u_{D(k)}^k = v_{D(k)}^k$$

which is equivalent to the problem

$$\text{Maximize} \quad r_i^k$$

subject to

$$r_i^k - r_j^k \leq c_{ij}^k + w_{ij}^k - (v_j^k - v_i^k) \quad \text{for all } (i,j) \epsilon A$$

$$r_{D(k)}^k = 0$$

Since the w values are assumed to satisfy equation (11.1) in the current dual solution,

$$\text{if } c_{ij}^k < v_j^k - v_i^k \implies w_{ij}^k > 0$$

$$\implies c_{ij}^k + w_{ij}^k - (v_j^k - v_i^k) = 0, \text{ and}$$

$$\text{if } c_{ij}^k \geq v_j^k - v_i^k \implies w_{ij}^k = 0$$

$$\implies c_{ij}^k + w_{ij}^k - (v_j^k - v_i^k) = c_{ij}^k - (v_j^k - v_i^k)$$

Therefore, we can replace the right hand side of constraint (11.2) by the expression Max $\{0, c_{ij}^k - (v_j^k - v_i^k)\}$. Thus, the problem reduces to one of finding the shortest path from D(k) to node i in the 'Reverse' graph (with all the original arc directions reversed), where each arc (j,i) is assigned the length p_{ij}^k = Max $\{0, c_{ij}^k - (v_j^k - v_i^k\} \geq 0$. The optimal value of r_i^k, i.e., the maximum amount by which the current value of v_i^k can decrease without reducing $v_{D(k)}^k$ or increasing any other value of w_{ij}^k, is the length of this shortest path. We can then update v and w values as follows:

$$v_i^k \leftarrow v_i^k - r_i^k$$

$$w_{ij}^k \leftarrow w_{ij}^k - \text{Min } \{w_{ij}^k, r_i^k - r_j^k\}$$

The updated dual values have the following characteristics.

(1) The new v values are obviously less than or equal to the original values. However, for every node i (including O(k) and D(k)) that lies on the current shortest path in G from O(k) to D(k), the v values remain unchanged. In particular, the dual objective function value is unchanged since $v_{D(k)}^k$ is not changed.

(2) When the w values are updated in this manner, they satisfy equation (11.1) with respect to the new v values, i.e., w_{ij}^k = Max $\{0,(v_j^k - v_i^k\}$ for all $(i,j) \epsilon A$ after the updating step.

Claim: The new w values are all less than or equal to their previous

values.

This result follows from the fact that $r_i^k - r_j^k \geq 0$ for all $(i,j) \epsilon A$, since the arc lengths p_{ij}^k in the subproblem are all non-negative. Therefore, Min $\{w_{ij}^k, r_i^k - r_j^k\} \geq 0$.

This v-improvement method reduces the value of v_i^k by the maximum possible extent, namely by r_i^k. Alternatively, we could reduce v_i^k just enough to make the w_{1i}^k values, for arcs (1,i) incident to node i, zero. The corresponding changes in the other v values, particularly for nodes along the shortest reverse path from D(k) to node i, would be different in this latter approach. Since our w-modification procedure changes several v values at the same time, it is very likely that many w values will reduce simultaneously. The procedure can be repeated several times until no additional w values are reduced. Finally, recall that we had assumed an intermediate w-updating step before initiating this v-modification procedure. However, this step is not necessary since the 'reverse arc lengths' $p_{ij}^k = \text{Max } \{0, c_{ij}^k - (v_j^k - v_i^k)\}$ do not depend on w_{ij}^k. Similarly, the w_{ij}^k values need not be updated at intermediate stages of this procedure.

This v-improvement method <u>decreases</u> w values by manipulating the v values suitably. Later, we discuss another method, based on complementary slackness conditions, for identifying and appropriately decreasing some w values. Unlike the Composite method, the v-improvement method is a heuristic because it may not necessarily find the optimal set of v values that minimize the sum of the w values.

So far, we have described several alternative ways to implement the w-modification and v-updating steps of the basic procedure. Combining

these components in various ways gives rise to different dual ascent procedures for the Uncapacitated Network Design problem. At the end of this chapter, we report on computational experience with one particular implementation of the dual ascent procedure.

As we mentioned at the outset, the basic procedure does not encompass all the possible ascent strategies; therefore, the final dual solution derived using the basic procedure can usually be improved further. Next, we describe improvement methods that are based on the LP complementary slackness conditions. We also describe dual-based heuristic procedures that use an intial solution derived from the dual ascent solution.

4.3.4 Using Complementary Slackness conditions for dual ascent:

The dual ascent algorithm is, in general, not guaranteed to give the optimal solution of [DUND]. Hence, there may be no primal LP solution that, along with the final dual ascent solution, satisfies all the LP complementary slackness (abbreviated as CS) conditions. In this section, we show how to exploit the CS violations in order to determine directions of change in the dual variables that enable a further increase in the dual objective function value. In the context of the Plant Location problem, Erlenkotter [1978] and Van Roy and Erlenkotter [1982] have devised dual adjustment procedures that use the CS conditions in this manner. Similarly, Magnanti and Wong [1984b] have outlined an iterative approach for the Uncapacitated Network Design problem, in which the CS conditions are used as a feedback mechanism for initializing the basic procedure at each iteration.

Let us begin by reviewing the complementary slackness conditions and examining some of their implications. We will label the three CS conditions as CS1, CS2 and CS3.

CS1: $x^k_{ij} > 0 \implies v^k_j - v^k_i = c^k_{ij} + w^k_{ij}$ for all $(i,j) \epsilon A$ and $k \epsilon K$.

CS2: $s_{ij} > 0 \implies y_{ij} = 0$ for all $(i,j) \epsilon A$.

CS3: $w^k_{ij} > 0 \implies x^k_{ij} = y_{ij}$ for all $(i,j) \epsilon A$ and $k \epsilon K$.

Consider, first, the condition CS1. According to this condition, we can allow positive flow x^k_{ij} of commodity k on arc (i,j) only if this arc lies on a shortest path for commodity k (using the Modified arc lengths e^k_{ij}). For our w-modification method, we first construct a primal solution, called the REFERENCE solution, in which each commodity flows on one of its current shortest paths. This solution, therefore, satisfies condition CS1. The reference design consists of all the arcs on which commodities are routed. If any commodity has alternate shortest paths, we choose the path that minimizes violations of condition CS2. We then use the complementary slackness violations by the current dual solution with respect to the Reference primal solution to determine appropriate modifications in the dual variables.

Now, consider the condition CS2. It specifies that y_{ij} can be positive only if the slack s_{ij} in constraint (9.3) of [DUND] is zero. A Reference solution that satisfies both CS1 and CS2 can be constructed only if each commodity k has a shortest path containing only arcs with zero slack. Both the Simultaneous w-increasing method and the Composite method ensure that at termination every commodity has at least one such path. (However, the final dual solution obtained using the Next Shortest path

method, with the exception of Magnanti and Wong's [1984b] implementation, may not satisfy this property.) Hence, when either of these methods is used in the w-increasing step, we can construct, from the final dual solution, a Reference primal solution that satisfies both CS1 and CS2.

Before examining the implications of the third CS condition, we will indicate how violations of CS2 can be used to change the dual variables appropriately when the Next shortest path method is used to increase w values in the basic procedure. Suppose, for some commodity k, every shortest path contains at least one arc with positive slack. Our strategy will be to reduce the slacks s_{ij} on several arcs, by increasing the w^k_{ij} values, so that commodity k has a ZERO-SLACK SHORTEST PATH from O(k) to D(k) after this modification. This objective can be achieved in several alternative ways. Consider, for instance, the following method.

> For all arcs (i,j) with positive slack s_{ij}, set $w^k_{ij} \leftarrow w^k_{ij} + s_{ij}$.
>
> Update the v values for commodity k using any of the v-updating methods devised for the basic procedure.
>
> Update the w values for commodity k using equation (11.1).

It is easy to see that in the new dual solution, commodity k must have a zero-slack shortest path. If l^*_k represents the length of the shortest path from O(k) to D(k) using $(c^k_{ij} + w^k_{ij} + s_{ij})$ as arc lengths, we can show that $v^k_{D(k)}$ must increase to this value l^*_k at the end of this procedure; thus, the dual objective function value also increases in the process. However, the method involves a duplication of effort, since it first increases the w values and later decreases them. As an alternative, we might consider a method that decreases the slacks s_{ij} for one arc at a time. Let (i',j') be the arc with the smallest positive slack from among all the arcs in the

current shortest paths for commodity k. When we decrease $s_{i',j'}$ to zero by increasing $w^k_{i',j'}$, the path containing arc (i',j') may no longer be a shortest path for commodity k. After identifying the new shortest path, we can repeat this procedure (of identifying the arc with the smallest positive slack in the new shortest path, and decreasing its slack to zero) several times until commodity k has a zero-slack shortest path. A third approach is to rank all the arcs of the network in order of increasing slack; starting with the first arc with positive slack, this method sequentially decreases the slacks on successive arcs till commodity k has a shortest path containing only arcs with zero slack. Finally, a labeling procedure similar to the one proposed by Magnanti and Wong [1984b] can also be used for this w-modification step. Unlike the Next shortest path method, all these w-modification schemes increase several w^k_{ij} values simultaneously, and all of them <u>increase</u> the dual objective function value z_D as well. Again, several variants are possible; for instance, in order to decrease the slack s_{ij} of a particular arc, we may choose to increase the w values corresponding to several commodities, rather than for just one commodity. Also, we can devise different rules for determining the order for considering the commodities. However, all these methods use the same basic principle, namely to modify the w variables so that violations (with respect to the Reference solution) of the complementary slackness condition CS2 are minimized.

So far, we have shown how to construct a Reference solution that, together with the current dual solution, satisfies both conditions CS1 and CS2. We will now examine one instance in which this solution violates condition CS3. This example will again enable us to identify w values that

can be modified to increase z_D. Suppose commodity k is routed on arc (i,j) in the the Reference solution; the slack s_{ij} must be zero, since this solution satisfies CS2. Suppose that $w^{k'}_{ij}$ is positive, for some commodity k' that is not routed on arc (i,j). Then, the Reference solution violates CS3, since $x^{k'}_{ij} = 0$ while $y_{ij} = 1$ in this solution; to satisfy CS3, $w^{k'}_{ij}$ must be zero. To eliminate this violation, we must either reduce $w^{k'}_{ij}$ to zero, or change other w values corresponding to commodity k so that commodity k' has a zero-slack shortest path via arc (i,j). We will adopt the former approach; we first present a general description of the problem that we want to address. For each arc (i,j) that is included in reference design, let K_{ij} be the set of commodities that are not routed on arc (i,j) in the referemce solution. For all commodities $k \epsilon K_{ij}$ with $w^k_{ij} > 0$, we want to reduce w^k_{ij} without reducing $v^k_{D(k)}$. Now, as w^k_{ij} decreases, the lengths of all the paths from O(k) to D(k) that contain arc (i,j) decrease. Therefore, while decreasing w^k_{ij}, we must ensure that none of these paths becomes shorter than the current shortest path. (Otherwise, $v^k_{D(k)}$ will decrease). Hence, if l^k_{ij} denotes the length of the shortest path containing arc (i,j), then w^k_{ij} can decrease by no more than $(l^k_{ij} - v^k_{D(k)})$. Thus, the method consists of the following steps:

> For each arc (i,j) belonging to the reference design, and for each commodity $k \epsilon K_{ij}$, determine the length l^k_{ij} of the shortest path from O(k) to D(k) via arc (i,j) (using $e^k_{ij} = c^k_{ij} + w^k_{ij}$ as arc lengths).
>
> Set $w^k_{ij} \leftarrow \text{Max } \{0, w^k_{ij} - (l^k_{ij} - v^k_{D(k)})\}$.

Notice that if this procedure decreases at least one w value, say w^k_{ij}, then the corresponding slack s_{ij} (which is currently zero) becomes positive. As

a result, some or all commodities that are currently routed on arc (i,j) (in the Reference solution) may no longer have any zero-slack shortest paths. Consequently, the dual objective function can be increased further during the next execution of the basic procedure. Thus, we have exploited a complemetary slackness violation to determine the direction of change in the dual variables. However, this procedure is not guaranteed to result in either a better dual solution or one that satisfies all the complementary slackness conditions. Indeed, it is very likely that no w value can be changed using this method (for instance, if arc (i,j) belongs to a current shortest path for all the commodities belonging to the set K_{ij}), in which case we can either terminate the dual ascent procedure or adopt the alternate approach for reducing violations of condition CS3.

Also, it is important to note that the Reference solution that we have constructed may not be the optimal primal solution (even if the current dual solution is optimal). Hence, while the current dual solution may violate several CS conditions with respect to the Reference solution, it may violate far fewer or none of these conditions with respect to some other primal solution. It might, therefore, be worthwhile to construct alternative primal solutions (especially when some commodities have more than one zero-slack shortest path) and to choose the solution with the least number of CS3 violations before proceeding with the w-modification phase. Alternatively, as Magnanti and Wong [1984b] have suggested, we can construct a good heuristic solution using the Reference solution as the starting point. Then, we can reinitiate the basic procedure, this time with restrictions imposed on the arc and commodity indices for which w values can be increased. For instance, if in the best heuristic solution

generated so far, arc (i,j) belongs to the design and commodity k is not routed on arc (i,j), we can impose the condition that w^k_{ij} must remain at zero; this restriction ensures that the resultant dual solution satisfies condition CS3 for arc (i,j) and commodity k (with respect to the current best heuristic solution).

In summary, thus far we have discussed the various components of a dual ascent algorithm for the Uncapacitated Network Design problem. We envisage the composite algorithm as consisting of two phases: the basic procedure, comprising of w-increasing, v-updating and w-updating steps, constitutes the first phase, while the second phase uses complementary slackness conditions for modifying the w values. We have described several alternative implementations for each step of the algorithm; choosing from among these competing methods would require computational comparisons for different problem structures. Many of the methods that we have described can in fact be interpreted as generalizations of dual ascent steps that have been suggested in the context of the Plant Location problem by Erlenkotter [1978] and Guignard and Spielberg [1979]. Next, we discuss dual-based heuristic methods for the Uncapacitated Network Design problem.

4.3.5 Dual-based Heuristic procedures:

Typically, heuristic methods for integer programming problems such as the NDP start with a good feasible solution and attempt to make local changes that monotonically decrease the cost of the solution (while maintaining feasibility) until they reach a local optimum. Heuristics are said to be "optimization-based" if either the starting solution is derived

from some partially completed optimization procedure or the local improvement steps rely on some optimization techniques. In this section, we discuss heuristics for the UNDP that use starting solutions derived from the dual ascent solution. We discuss two initialization methods before describing various local improvement procedures.

Constructing the initial feasible solution:

The Reference soluton that we constructed in the previous section is a logical starting solution for any UNDP heuristic. In this solution, each commodity is routed on one of its zero-slack shortest paths. The design contains all the arcs on which the various commodities are routed. This solution is attractive, since it (together with the final dual solution) satisfies two out of the three complementary slackness conditions. Note that all the arcs of this design have zero slack (in the final dual ascent solution); however, the network may contain other arcs with zero slack that are not included in this initial design.

An alternative approach, used by Magnanti and Wong [1984b], is to first include in the design all the arcs with zero slack (in the final dual solution). According to condition CS2, only arcs with zero slack (in an optimal dual solution) can have $y_{ij} > 0$; therefore, this choice is justified. Having established the network design, we can route each commodity on the 'cheapest' path in this design, where the original flow costs c^k_{ij} are used as arc lengths. We have already shown that at the end of the dual ascent procedure each commodity must have at least one zero-slack shortest path from its origin to its destination. Thus, the

chosen design must contain at least one feasible routing for each commodity. As a final step, this initialization procedure drops all the design arcs on which no commodity is routed.

Local Improvement procedures:

The most commonly used Local Improvement procedures for the NDP and related problems are the Add, Drop, and Interchange procedures. (See, for example, Billheimer and Gray [1973], Boorstyn and Frank [1977], Gerla and Kleinrock [1977]). The ADD procedure proceeds as follows: Suppose we add a new arc (i,j) to the current design. Then, the routing costs for some of the commodities may decrease, since the paths via arc (i,j) could be shorter than those along which the commodities are currently routed. Let p_{ij} be the total savings in routing costs for all commodities when arc (i,j) is added to the design; this savings can be easily determined by solving one shortest path problem (over the augmented network containing the new arc (i,j)) for each commodity. Then, the Net savings obtained by adding arc (i,j) to the design is $(p_{ij} - F_{ij})$. If some other arc in the current design becomes superfluous when arc (i,j) is added (i.e., when all the commodities that currently flow on a particular arc are rerouted along paths that do not contain this arc), the fixed cost of this arc must also be added to the net savings corresponding to arc (i,j). At each iteration, the algorithm evaluates the net savings for all the arcs not in the current design. If none of these savings is positive, the procedure terminates. Otherwise, it adds the arc that yields the greatest savings and repeats this process. DROP procedures operate in a similar manner, except that at each stage they drop arcs from the current design, rather than add new

arcs. Each iteration consists of identifying the arc, which when removed from the design, results in the maximum net savings. (If removing an arc destroys all the routes for any commodity, a negative net savings is assigned to this arc.) A logical extension of these algorithms is a composite ADD/DROP procedure that either alternates Add and Drop steps or uses them in tandem. An alternative method would be to Add or Drop more than one arc at each step. Similarly, we can use INTERCHANGE heuristics that replace an arc that is currently in the design with one that is not. All these algorithms identify locally optimal solutions, where the neighborhood of any given solution is defined as the set of all solutions whose designs differ from the given design in at most one arc.

Alternatively, suppose we define the neighbourhood of a given solution as the set of all solutions that differ in the routing of at most one commodity. A local optimum in this case can be found using a method that we call the FLOW REROUTING method. This method can be shown to be equivalent to the local improvement procedure proposed by Gallo and Sodini [1979] in the context of minimum concave-cost network flows. At each step, this method evaluates the best incremental savings obtained by routing a given commodity k on an alternate path, that may contain arcs not in the current design. Notice that, while evaluating such savings, we must account for commodity k's current routing cost, its new routing cost, the fixed costs of any arcs that are added to the design, and the fixed costs of any that are dropped from the current design. The commodity for which rerouting results in the highest positive incremental savings is rerouted and the process is repeated. If there is no such commodity, the algorithm terminates and the current solution is locally optimal in the sense defined

earlier. The problem of evaluating the best incremental savings for any given commodity k can be solved efficiently using the following procedure. Let P_k denote current route for commodity k. For every arc (i,j) on this path, define the parameter

$$p_{ij} = \begin{cases} c_{ij}^k & \text{if some other commodity is also currently routed on arc (i,j)} \\ c_{ij}^k + F_{ij} & \text{otherwise.} \end{cases}$$

p_{ij} represents the savings when commodity k does not flow on arc (i,j). Thus, the total savings when commodity k is 'withdrawn' from path P_k is

$$r_k = \sum_{(i,j) \in P_k} p_{ij} .$$

Having 'withdrawn' commodity k from path P_k, the path that yields the best incremental savings is clearly the shortest path from O(k) to D(k) with the 'new' incremental costs of flow along as arc lengths. (Note that the incremental costs on arcs belonging to P_k change when commodity k is withdrawn.) For all arcs (i,j) of path P_k, the incremental cost of routing one unit of commodity k is now p_{ij}. For arcs that do not belong to P_k, but which are included in the current design, the incremental cost is c_{ij}^k, while the remaining arcs have an incremental cost of $c_{ij}^k + F_{ij}$. Let P_k^* be the shortest path from O(k) to D(k) using these incremental costs as arc lengths, and let r_k^* be the length of this path. Then, the best incremental savings that can be achieved by rerouting commodity k is $(r_k - r_k^*)$. Notice that, in this new shortest path problem, the 'incremental' cost of the original path P_k is r_k. Hence, r_k^* must always be less than or equal to r_k; if $r_k^* = r_k$, none of commodity k's alternate routes yield net savings. The algorithm thus proceeds by solving a shortest path problem for each commodity, evaluating the differences $(r_k - r_k^*)$, and selecting the commodity for which the savings are the largest (and positive). It then

reroutes this commodity along the new shortest path. The procedure stops when the savings for all commodities are zero. Singhal [1984] has used a similar procedure for finding heuristic solutions to a routing problem in point-to-point delivery systems. In our computational experiments, both for the UNDP and the LTL Consolidation problem (described in Chapter 5), we used this Flow Rerouting method as the local improvement heuristic.

4.4 Dual Ascent procedures for the Capacitated Network Design problem

So far we have restricted our attention to the Uncapacitated Network Design problem. In this section, we discuss extensions of the algorithms developed in the last two sections to the capacitated case. Toward the end of this section, we also briefly indicate how the procedures can be modified to handle additional side constraints, such as those generated in Chapter 3.

We begin by reviewing the Arc-flow formulation [CNDAF] of the Capacitated Network Design problem (CNDP) and by formulating its LP dual. In order to highlight the main ideas, we consider the special case in which all commodities have unit demands. The methods that we discuss for this problem can be extended in obvious ways to the more general case with arbitrary demands. The Arc-flow formulation for the CNDP with unit demands is

[CNDAF]

$$\text{Minimize} \quad \sum_k \sum_{(i,j)} c_{ij}^k x_{ij}^k + \sum_{(i,j)} F_{ij} y_{ij} \tag{12.1}$$

subject to

$$-\sum_j x_{ij}^k + \sum_j x_{ji}^k = \begin{cases} -1 & \text{if } i = O(k) \\ +1 & \text{if } i = D(k) \\ 0 & \text{otherwise} \end{cases} \quad \text{for } k \epsilon K \tag{12.2}$$

$$x_{ij}^k \leq y_{ij} \quad \text{for } (i,j) \epsilon A,\ k \epsilon K \tag{12.3}$$

$$\sum_k x_{ij}^k \leq B_{ij} y_{ij} \quad \text{for } (i,j) \epsilon A \tag{12.4}$$

$$y_{ij} \leq 1 \quad \text{for } (i,j) \epsilon A \tag{12.5a}$$

$$x_{ij}^k,\ y_{ij} \geq 0 \quad \text{for } (i,j) \epsilon A,\ k \epsilon K \tag{12.5b}$$

$$y_{ij} \text{ integer} \quad \text{for } (i,j) \epsilon A \tag{12.5c}$$

where B_{ij} is the capacity of arc (i,j).

The LP relaxation of [CNDAF] is obtained by relaxing the integer constraints (12.5c). Notice that, unlike the UNDP, the upper bounding constraints (12.5a) are essential for this LP relaxation. Let v_i^k, w_{ij}^k, t_{ij}, and u_{ij} be the dual variables corresponding to the constraints (12.2), (12.3), (12.4), (12.5a) respectively. Then, the dual [DCND] of the LP relaxation of [CNDAF] is

[DCND]

$$\text{Maximize} \quad z_D = \sum_k v_{D(k)}^k - \sum_{(i,j)} u_{ij} \tag{13.1}$$

subject to

$$v_j^k - v_i^k \leq c_{ij}^k + w_{ij}^k + t_{ij} \quad \text{for } (i,j) \epsilon A,\ k \epsilon K \tag{13.2}$$

$$\sum_k w_{ij}^k + B_{ij} t_{ij} - u_{ij} \leq F_{ij} \quad \text{for } (i,j) \epsilon A \tag{13.3}$$

$$v_i^k \text{ unrestricted in sign} \quad \text{for } i \epsilon N,\ k \epsilon K \tag{13.4a}$$

$$w_{ij}^k,\ t_{ij},\ u_{ij} \geq 0 \quad \text{for } (i,j) \epsilon A,\ k \epsilon K. \tag{13.4b}$$

Again, without loss of generality, we have set the variable $v^k_{O(k)}$ to zero for all commodities $k \epsilon K$. We let s_{ij} denote the slack in constraint (13.3) corresponding to arc (i,j).

The basic principle underlying the dual procedure for this capacitated problem is similar to that used in the uncapacitated case. For any given set of values for w^k_{ij} and t_{ij} satisfying constraints (13.3), the largest dual objective function value is obtained by setting $v^k_{O(k)}$ equal to the length of the shortest path from O(k) to D(k) using $(c^k_{ij} + w^k_{ij} + t_{ij})$ as arc lengths. Hence, the MODIFIED arc lengths e^k_{ij} are now defined as

$$e^k_{ij} = c^k_{ij} + w^k_{ij} + t_{ij} \quad \text{for all } (i,j) \epsilon A, \; k \epsilon K.$$

Since e^k_{ij} depends on both w^k_{ij} and t_{ij}, our strategy will be to increase either w^k_{ij} or t_{ij} (or both) in such a way that the shortest path length for one or more one commodity increases. However, there are some important differences in the way that this modification is carried out in the capacitated context. We next discuss some of these differences.

(1) Increasing w^k_{ij} increases the modified arc length for just one commodity (in particular, it increases e^k_{ij}); however, increasing t_{ij} increases the modified length of arc (i,j) for <u>all</u> commodities. Hence, by increasing t_{ij} it is possible to increase the shortest path lengths for several commodities simultaneously; increasing w^k_{ij} can at best increase the shortest path length for commodity k alone.

(2) Every unit increase in t_{ij} decreases the 'resource' in the constraint (13.3) corresponding to arc (i,j) by B_{ij} units. Therefore, either the slack s_{ij} must decrease or the 'surplus' u_{ij} must increase by a total of B_{ij} units to accommodate a unit increase in t_{ij}. In contrast, in order to maintain feasibility in constraints (13.3), every unit increase in w^k_{ij} requires just a unit decrease in s_{ij} or a unit increase in the u_{ij}.

(3) For any arbitrary non-negative values of w^k_{ij} and t_{ij}, constraints (13.3) can be made feasible by appropriately choosing the u_{ij} values. However, for all $(i,j) \epsilon A$, u_{ij} has an objective function coefficient of -1; therefore, for any given values of w^k_{ij} and t_{ij}, u_{ij} must be set equal to Max $\{0, \; \Sigma_k w^k_{ij} + B_{ij} t_{ij} - F_{ij}\}$, in order to maximize z_D. Furthermore, while increasing w^k_{ij} or t_{ij} (for increasing the shortest

path length $v^k_{D(k)}$ for one or more commodities), we must account for any consequent increase in u_{ij} and the corresponding decrease in z_D. We illustrate this point with an example. Suppose arc (i,j) belongs to the current shortest path for commodity k, and increasing w^k_{ij} by 1 unit increases $v^k_{D(k)}$. If the current slack s_{ij} for arc (i,j) is zero, then increasing w^k_{ij} by 1 unit would entail increasing u_{ij} by 1 unit as well. Consequently, the increase in $v^k_{D(k)}$ is nullified by the increase in u_{ij}. This example demonstrates that it is not worthwhile to increase w^k_{ij} whenever $s_{ij} = 0$. However, as we show later, in some situations it is advantageous to increase t_{ij} (and u_{ij}) even when the slack s_{ij} is zero.

These three characteristics of the dual problem [DCND] highlight the tradeoff between modifying the w^k_{ij} values and changing t_{ij} at each stage of the dual ascent procedure. They also point to the need to account for the interaction between the w, t, and u values and the impact on the objective function of modifying the u_{ij}'s. Next, we identify situations when increasing w^k_{ij} is a better strategy (locally) than increasing t_{ij} and vice versa.

Suppose we are given a dual solution, and we want to determine, for a particular arc (i,j), whether to increase t_{ij} or to increase w^k_{ij} for one or more commodities k, in order to increase z_D. One possible measure for comparing the two methods is the increase in z_D <u>per unit increase</u> in t_{ij} and w^k_{ij}. However, as the preceding discussion suggests, this measure is inappropriate since increasing t_{ij} by 1 unit uses up B_{ij} units of the resource in constraint (13.3), whereas increasing w^k_{ij} by 1 unit requires just 1 unit of this resource. Therefore, if we let

$$F'_{ij} = F_{ij} - \sum_k w^k_{ij} - B_{ij}t_{ij} \qquad \text{for all } (i,j) \epsilon A,$$

an appropriate measure for comparison is the ratio r defined as the <u>increase in z_D per unit decrease in F'_{ij}</u>. In other words, we will compare the increase in z_D when t_{ij} increases by ϵ/B_{ij} units with the increase in

z_D when w_{ij}^k increases by ϵ units. We let r_t and r_w^k denote, respectively, the value of this ratio r for the two ascent methods, namely, increasing t_{ij} and increasing w_{ij}^k. (Note: The ratio r depends on the arc (i,j) that we are considering; however, for convenience, we omit the arc index for the ratio r.) In order to determine the best local ascent strategy, we must compare r_t with $r_w = \text{Max}_k\, r_w^k$. Suppose commodity k^* has the maximum ratio r_w^k among all commodities k. Then, we must increase $w_{ij}^{k^*}$ if $r_w = r_w^{k^*}$ is greater than r_t and increase t_{ij} otherwise, assuming that both r_w and r_t are positive. (If both these ratios are zero or negative, neither t_{ij} nor w_{ij}^k, for any commodity k, should be increased.) This idea of selecting the best 'local' strategy using this type of ratio has been employed, in different forms, in the dual ascent methods devised by Wong [1984] for the Steiner Network problem and by Magnanti and Wong [1984b] for the Uncapacitated Network Design problem.

We next identify circumstances under which r_t is greater than r_w and vice versa. For this purpose, we introduce the following definition.

Definition 4.2:

> Arc (i,j) is said to be CRITICAL for commodity k if it belongs to all the current shortest paths (using e_{ij}^k as arc lengths) for commodity k.

Recall from our earlier discussions that increasing the modified arc length e_{ij}^k increases the shortest path length $v_{D(k)}^k$ if and only if arc (i,j) is critical for commodity k. Let K_{ij} denote the set of commodities for which arc (i,j) is critical. To determine the effect on z_D (for evaluating the ratio r) of increasing either t_{ij} or w_{ij}^k by some small positive value, we must account for

(1) the change in the shortest path length $v^k_{D(k)}$ for each commodity k.

If commodity k belongs to the set K_{ij}, its shortest path length increases when either t_{ij} or w^k_{ij} increases, since both these modifications increase the modified arc length e^k_{ij}. $v^k_{D(k)}$ does not change if commodity k does not belong to K_{ij}; and,

(2) the change in the values of u_{ij}.

When the slack s_{ij} is zero, u_{ij} must be increased (and hence the second term of z_D decreases) when either t_{ij} or w^k_{ij} increases.

Therefore, for some small $\epsilon > 0$,

(a) increasing w^k_{ij} by ϵ units

(1) increases z_D by ϵ units if $s_{ij} > 0$ and $k \epsilon K_{ij}$,

(2) does not increase z_D if either $s_{ij} = 0$ and $k \epsilon K_{ij}$, or if $s_{ij} > 0$ and $k \notin K_{ij}$, and

(3) decreases z_D by ϵ units if $s_{ij} = 0$ and $k \notin K_{ij}$, and

(b) increasing t_{ij} by ϵ/B_{ij} units

(1) increases z_D by $|K_{ij}|\epsilon/B_{ij}$ units if $s_{ij} > 0$, and

(2) increases z_D by $(|K_{ij}|\epsilon/B_{ij}) - \epsilon$ units if $s_{ij} = 0$.

Fact (a) implies that $r_w = 1$ if $s_{ij} > 0$ and $|K_{ij}| > 1$; otherwise, $r_w \leq 0$. In particular, r_w is zero or negative whenever the slack s_{ij} is zero. This result is consistent with our earlier observation that it is not worthwhile to increase w^k_{ij} for any commodity k if the slack s_{ij} is zero. From fact (b) we see that

$$
\begin{aligned}
r_t > 1 \quad &\text{if } |K_{ij}| > B_{ij} \text{ and } s_{ij} > 0, \text{ or} \\
&\quad |K_{ij}| > B_{ij} + 1 \text{ and } s_{ij} = 0, \\
0 < r_t \leq 1 \quad &\text{if } B_{ij} \geq |K_{ij}| > 1 \text{ and } s_{ij} > 0, \text{ and} \\
r_t \leq 0 \quad &\text{if } |K_{ij}| = 0 \text{ or } |K_{ij}| < B_{ij} \text{ and } s_{ij} = 0.
\end{aligned}
$$

We summarize these observations in Table 1. The ratios in the table

Table 1: Values of the ratios r_t and r_w in different cases

	$\|K_{ij}\| = 0$	$1 \leq \|K_{ij}\| \leq B_{ij}$	$\|K_{ij}\| > B_{ij}$
$s_{ij} = 0$	$r_t < 0$ $r_w < 0$	$r_t \leq 0$ $r_w = 0$	$r_t > 0$ $r_w = 0$
$s_{ij} > 0$	$r_t = 0$ $r_w = 0$	$r_t \leq 1$ $r_w = 1$	$r_t > 1$ $r_w = 1$

suggest the following optimal local ascent strategy:

(a) if arc (i,j) is critical for more than B_{ij} commodities, increase t_{ij} (regardless of whether or not the slack s_{ij} is zero), and

(b) if arc (i,j) has a positive slack s_{ij} and is critical for at least one, but not more than B_{ij}, commodities, increase w^k_{ij} for one or more commodities $k \epsilon K_{ij}$.

Intuitively, if arc (i,j) is critical for more than B_{ij} commodities, we would expect the strategy of increasing t_{ij} to be better because increasing t_{ij} by ϵ increases the first component of z_D by more than $B_{ij}\epsilon$, while using up only $B_{ij}\epsilon$ units of the slack s_{ij} or u_{ij}. To achieve this same total increase in shortest path lengths by increasing the w^k_{ij}'s, we would have to use up more than $B_{ij}\epsilon$ units of the 'resource' of constraint (13.3). The decision rule confirms this intuition.

Having identified the rule for deciding between increasing t_{ij} and increasing w^k_{ij}, we now determine the exact amount by which t_{ij} or w^k_{ij} must

be increased. For this purpose, we use the notion of the NEXT PATH first introduced in Section 4.3.2 (in the context of the Next shortest path w-increasing method). The Next path P^k_{ij} for commodity k with respect to a given arc (i,j) is the shortest path from O(k) to D(k) (using the the modified arc lengths e^k_{ij}) that does not contain arc (i,j). Let $l(P^k_{ij})$ denote the length of the Next path with respect to arc (i,j); $v^k_{D(k)}$ is the length of the current shortest path for commodity k. For all arcs $(i,j) \epsilon A$ and commodities $k \epsilon K$, we define g^k_{ij} to be the difference $(l(P^k_{ij} - v:\{D(k)]^k)$. As noted earlier, g^k_{ij} denotes the maximum amount by which the shortest path length for commodity k can be increased by increasing the modified arc length e^k_{ij}. Also, observe that $g^k_{ij} > 0$ if and only if arc (i,j) is critical for commodity k (i.e., if and only if arc (i,j) belongs to all the shortest paths for commodity k). Let h_{ij} be the $(B_{ij} + 1)^{th}$ largest value of g^k_{ij}. Thus, h_{ij} is positive only if arc (i,j) is critical for more than B_{ij} commodities. In this case, we have already shown that increasing t_{ij} is the preferred ascent strategy. We now claim that t_{ij} must be increased by exactly h_{ij} units. If t_{ij} increases by a smaller amount, arc (i,j) continues to be critical for at least $(B_{ij}+1)$ commodities; hence, by our decision rule, t_{ij} must be increased further. On the other hand, when t_{ij} increases by more than h_{ij}, arc (i,j) belongs to the shortest paths for fewer than B_{ij} commodities. In this situation, it is easy to show that by decreasing t_{ij} (and increasing some w values correspondingly) we can either reduce u_{ij} (thus increasing z_D) or increase s_{ij}. Therefore, when $h_{ij} > 0$, increasing t_{ij} by h_{ij} units is the best local modification strategy. If h_{ij} is zero, the w^k_{ij} values for one or more commodities k belonging to the set K_{ij} must increase by a maximum of g^k_{ij} until the slack reduces to zero. For the CNDP, therefore, we can

modify the first step of the basic procedure discussed in Section 4.3.1 as follows:

w-increasing/t-increasing step:

For one or more arcs $(i,j) \epsilon A$,
- determine the parameters g^k_{ij} and h_{ij}.
- if $h_{ij} > 0$, increase t_{ij} by h_{ij} units.
- if $h_{ij} = 0$, for some $k \epsilon K_{ij}$, increase w^k_{ij} by a maximum of g^k_{ij} units until the slack s_{ij} reduces to zero.
- update s_{ij} and u_{ij}.

If no t or w value can be increased in this step, STOP.

Let us now disc the subsequent steps in the basic procedure for the CNDP. The v-modification procedures that we discussed earlier are applicable directly to the capacitated case as well. And, for the w-decreasing step we can use a property analagous to Property 2; namely, for given values of v^k_i and t_{ij}, the 'best' values of w^k_{ij} (i.e., the values of w^k_{ij} that minimize u_{ij} or maximize the slack s_{ij}) are given by the expression

$$w^k_{ij} = \text{Max} \{ 0, v^k_j - v^k_i - c^k_{ij} - t_{ij} \} \quad \text{for all } (i,j) \epsilon A, k \epsilon K.$$

In addition to these three steps, for the Capacitated Network Design problem, we introduce a new fourth step called the t-w interchange step which interchanges values of t_{ij} and w^k_{ij}. We next justify the need for such a step by outlining particular instances in which this interchange procedure increases the dual objective function. Suppose at the end of the w-decreasing step, more than B_{ij} w values corresponding to arc (i,j) are positive. This situation could arise, for instance, if arc (i,j) is critical for less than $(B_{ij}+1)$ commodities in several initial iterations of the basic procedure, prompting increases in w^k_{ij} rather than in t_{ij}. Let q

be the $(B_{ij}+1)^{th}$ largest value of w_{ij}^k; by our hypothesis, q must be positive. We claim that it is optimal to set

$$t_{ij} \leftarrow t_{ij} + q \qquad (14.1)$$

$$w_{ij}^k \leftarrow \text{Max } \{0,(w_{ij}^k - q)\} \text{ for all } k \epsilon K.$$

Notice that when t_{ij} and w_{ij}^k's are updated in this manner, the modified arc length e_{ij}^k does not decrease for any commodity; in fact, for those commodities with $w_{ij}^k < q$, e_{ij}^k increases. Consequently, the shortest path lengths $v_{D(k)}^k$ either increase or stay the same. Also, as a result of this updating step, the quantity $(\Sigma_k w_{ij}^k + B_{ij}t_{ij})$ decreases, since t_{ij} increases by q while $\Sigma_k w_{ij}^k$ decreases by at least $(B_{ij}+1)q$. Therefore, if $u_{ij} > 0$ in the current solution, u_{ij} decreases after this updating step, thus increasing z_D; if not, s_{ij} increases, providing greater scope for dual ascent in the next execution of the w-increasing/t-increasing step. Thus, when more than B_{ij} w values corresponding to arc (i,j) are positive, we see that it is advantageous to increase t_{ij} and decrease the w_{ij}^k's. Notice that after t_{ij} has increased, at most B_{ij} w values corresponding to arc (i,j) remain positive.

Next, we will consider the opposite situation when t_{ij} must be reduced and one or more w_{ij}^k values must be increased. Suppose, at some stage of the basic procedure, arc (i,j) was critical for more than B_{ij} commodities, prompting an increase in t_{ij}. Subsequently, suppose arc (i,j) belongs to the current shortest paths for fewer than B_{ij} commodities. This situation could arise, for instance, when the t value corresponding to some other arc (i1,j1) increases; if (i,j) and (i1,j1) lie on the same initial shortest path for commodity k, this path would no longer be the shortest for commodity k after $t_{i1,j1}$ increases. In this case, it is optimal to

decrease t_{ij} and increase some of the w values. Before justifying this claim, we define some parameters that will enable us to determine the exact amount by which t_{ij} must be decreased. For any arc (i,j) and commodity k, let Q^k_{ij} be the shortest path (using the modified arc lengths e^k_{ij}) from O(k) to D(k) <u>via</u> arc (i,j), and let $l(Q^k_{ij})$ be the length of this path. As we noted in Section 4.3.5, if l^*_k denotes the length of the current shortest path for commodity k, then the modified arc length e^k_{ij} can be decreased by a maximum of $(l(Q^k_{ij}) - l^*_k)$ without decreasing the value of $v^k_{D(k)}$. (Note: If arc (i,j) belongs to at least one current shortest path for commodity k, then $l(Q^k_{ij}) = l^*_k$.) Let q' be the B^{th}_{ij} smallest value of this difference $(l(Q^k_{ij}) - l^*_k)$. Since arc (i,j) belongs to the shortest paths for fewer than B_{ij} commodities, q' must be positive. Assume for simplicity, that $t_{ij} \geq q'$. Consider now the effects of the following t-modification and w-modification:

$$t_{ij} \text{ <-- } t_{ij} - q' \qquad (14.2)$$

$$w^k_{ij} \text{ <-- } w^k_{ij} + \text{Max } \{0, q' - (l(Q^k_{ij} - l^*_k)\} \quad \text{for all } k \epsilon K.$$

Thus, w^k_{ij} increases for at most $(B_{ij}-1)$ commodities; in particular, w^k_{ij} increases by q' if arc (i,j) belongs to a current shortest path for commodity k. It is easy to show that with this updating scheme, none of the shortest path lengths $v^k_{D(k)}$ decrease and that $\Sigma_k w^k_{ij}$ decreases by less than $B_{ij}q'$. Hence, if $u_{ij} > 0$ in the current solution, it decreases after this modification, thus increasing z_D. On the other hand, if $u_{ij} = 0$, then the slack s_{ij} increases as a result of this t-w interchange. Therefore, when $t_{ij} > 0$ and arc (i,j) lies on the shortest path for less than B_{ij} commodities, it is advantageous to increase w^k_{ij} values at the expense of t_{ij}. We now summarize these two t-w interchange strategies.

t-w interchange step:

For all arcs $(i,j) \in A$,

t-increase, w-decrease:
if more than B_{ij} w values corresponding to arc (i,j) are positive, increase t_{ij} and decrease the w^k_{ij}'s according to the equations (14.1), and
update s_{ij} and u_{ij} accordingly, and

t-decrease, w-increase:
if $t_{ij} > 0$ and arc (i,j) belongs to the shortest paths for fewer than B_{ij} commodities, increase the w^k_{ij}'s and decrease t_{ij} according to equations (14.2), and
update s_{ij} and u_{ij} accordingly.

Notice that, at the end of this t-w interchange step, each arc (i,j) has at most B_{ij} positive w values. As we shall see later, this characteristic ensures that the set of all arcs with zero slack in the final dual solution constitutes a 'feasible' design.

An alternative ascent procedure:

In the algorithm that we just described, the first step of the basic procedure was modified in order to account for the tradeoff between increasing t and w values at each iteration. An alternative dual ascent approach for the Capacitated Network Design problem uses the dual ascent algorithm that we devised for the Uncapacitated problem as a subroutine. This procedure consists of two of phases. Assuming that the t values are fixed, the first phase modifies the w values to increase z_D as much as possible. The algorithm for the UNDP can therefore be applied, without modification (except in the definition of the Modified arc lengths e^k_{ij}) for this phase. The second phase interchanges t and w values, using the t-w interchange step described earlier, in order to either increase z_D further or increase the slacks s_{ij} for some arcs. The algorithm terminates when z_D cannot be increased further in either phase. This procedure is especially

appealing because of its relative ease of implementation.

Capacitated problems with arbitrary demand patterns:

Finally, the algorithms just discussed can be easily adapted to solve Capacitated Network Design problem with arbitrary demands R_k for each commodity k. In this case, the forcing constraints are

$$x^k_{ij} \leq d^k_{ij}\, y_{ij} \quad \text{for all } (i,j)\epsilon A,\ k\epsilon K,$$

where $d^k_{ij} = \text{Min } \{B_{ij}, R_k\}$.

Therefore, the dual objective function becomes

$$z_D = \sum_k R_k v^k_{D(k)} - \sum_{(i,j)} u_{ij}$$

and the constraints (13.3) of [DCND] are

$$\sum_k d^k_{ij} w^k_{ij} + B_{ij} t_{ij} - u_{ij} \leq F_{ij} \quad \text{for all } (i,j)\epsilon A \qquad (13.3a)$$

Consequently, whenever w^k_{ij} increases by ϵ,

- the slack s_{ij} decreases (or u_{ij} increases) by $d^k_{ij}\epsilon$, and
- if arc (i,j) belongs to all the shortest paths for commodity k, z_D increases by $R_k\epsilon$.

Similarly, when t_{ij} increases by ϵ, the slack s_{ij} decreases by $B_{ij}\epsilon$ (assuming $s_{ij} > 0$ currently), and z_D increases by

$$\sum_{k\epsilon K_{ij}} R_k\epsilon,$$

where K_{ij} is the set of all commodities for which arc (i,j) is critical. Using these values for the increase in z_D and decrease in the slacks, we can now compute the ratios r_t and r_w that we discussed earlier, in order to decide whether to increase t_{ij} or w^k_{ij} at each step. Because the dual objective function coefficients R_k as well as the coefficients d^k_{ij} in constraints (13.3a) are different for different commodities, an additional ascent strategy is possible in this case; namely, to interchange w values

corresponding to the same arc,but different commodities (i.e., to increase w_{ij}^{k1} and simultaneously decrease w_{ij}^{k2} for two different commodities k1 and k2). Also, while scanning the commodity list for choosing the commodity k whose w value value must be increased next, we must give precedence to commodities with the largest ratio R_k/d_{ij}^k. Thus, the modifications for the general capacitated case are quite straightforward.

Heuristics for the CNDP:

For the Capacitated problem, unlike the UNDP, the task of finding a primal feasible solution from the final dual ascent solution is non-trivial. In fact, because of the multicommodity aspect of the problem, there is no simple way to find a feasible routing even when a 'feasible' design is provided. However, the final dual ascent solution can be exploited in two ways for finding an initial feasible solution.

(1) As we show below, the set of arcs that have zero slack in the final dual solution constitutes a 'feasible' design. Thus, we can use this set or an appropriate subset as our initial design, obviating the need to 'search' for a feasible design or to use an 'all-inclusive' design (i.e., include all the arcs of the given network) for the initial solution, and

(2) the problem has a feasible flow pattern, in the zero-slack design, in which each commodity flows only on one of its current shortest paths (in the final dual solution). This property considerably narrows down the number of possible paths for each commodity, and hence the number of 'flow combinations' that must be considered for finding the feasible routing.

We will now justify these two assertions.

Claim:

The Capacitated Network Design problem has a feasible routing in which each commodity k flows on one of its zero-slack shortest paths (in the final dual solution).

Proof:

First, there must be a feasible routing in which each commodity flows on one of its current shortest paths (that contain both zero and positive slack arcs). Otherwise, some arc (i,j) is critical for more than B_{ij} commodities; in this case, our ascent procedure would have increased the dual objective function value by increasing t_{ij} (whether or not $s_{ij} = 0$). Now, consider all the arcs (i,j) with positive slack s_{ij} that belong to a current shortest path for at least one commodity. If arc (i,j) is critical for some commodity k, then z_D increases when either w^k_{ij} or t_{ij} increases. Hence, assume that (i,j) is not critical for any commodity. Suppose, we now increase all the w values corresponding to arc (i,j), say by $s_{ij}/|K|$. This increase in the modified arc length e^k_{ij} makes all the paths containing arc (i,j) longer than the current shortest path, for every commodity. Therefore, after this updating step, all the current shortest paths for all commodities contain only zero-slack arcs. Repeating the argument used at the beginning of the proof, we see that there must be a feasible routing that uses only such paths for each commodity (otherwise, the dual objective function can be increased further). Therefore, in order to obtain a feasible solution, it suffices to consider routing each commodity on one of its zero-slack shortest paths.

This result suggests that we can reduce the number of arcs in the initial design even further by considering only those zero-slack arcs that belong to one or more current shortest paths. Having obtained such a 'feasible' design, we can adopt one of two strategies for 'completing' the

initial solution, i.e., for determining a feasible routing along the arcs of the given design. One method is to use a Multicommodity Maxflow algorithm for finding the feasible routing; to find the 'best' feasible routing, with respect to the given design, we would have to use a Minimum-cost Multicommodity flow algorithm. Alternatively, we could construct a feasible routing pattern using a systematic enumeration procedure. Since we need only consider routing each commodity on one of its zero-slack shortest paths, it may be possible to enumerate several potential routing combinations before finding a feasible solution.

Having obtained an initial feasible solution, the local (or incremental) improvement procedures, such as the Add/Drop, Interchange and Flow Rerouting methods, that we discussed earlier can be applied to obtain better solutions. However, it is impractical to identify the optimal flow pattern for each design that is evaluated at every iteration; instead, we must resort to some heuristic procedure for evaluating the savings for alternative designs in order to decide whether the current design must be changed. Therefore, while using the Add/Drop or Interchange method, we cannot usually guarantee that the final solution is a local optimum. The Flow Rerouting method does not suffer from this disadvantage, since at each stage it involves evaluating the impact of rerouting just one commodity.

Incorporating additional side constraints:

We have seen how the basic dual ascent algorithm can be modified to handle arc capacity constraints. We will now briefly discuss extensions of the algorithm to Network Design problems with additional side constraints.

Such constraints might include topological conditions imposed upon the network design (for example, upper bounds on the degree of each node in the final design), logical restrictions on the design variables (for example, to ensure that some set of arcs is not included if another is, and so on), valid inequalities that strengthen the LP formulation of the NDP (such as those discussed in Chapters 2 and 3), and a variety of other restrictions. We chose to deal with the capacitated case separately, rather than outline a general method that handles capacity as well as other side constraints, because the capacity constraints have a special structure that the methods outlined in the previous section exploit. In contrast, we now consider additional constraints of a more general form; accordingly, we will indicate only broad approaches that might be adopted to account for such constraints. For the Plant Location problem, Guignard and Spielberg [1979] have proposed extensions of the basic dual ascent algorithm in order to solve problems with similar additional side constraints.

One possible approach, when the problem contains additional side constraints, is to use Lagrangian relaxation. This scheme dualizes the additional constraints using some suitable Lagrange multipliers. For a given set of Lagrange multipliers, the corresponding subproblem is an 'unconstrained' Network Design problem; hence, the dual ascent algorithm for the unconstrained case can be used to find a lower bound for this subproblem. By using this algorithm as a subroutine, we can iteratively modify the Lagrange multipliers to obtain good lower bounds for the original problem. Since we do not solve the subproblems to optimality, the Lagrangian master problem can be solved only approximately.

Alternatively, as we did for the capacitated case, we could adapt the basic dual ascent algorithm itself to accomodate the additional side constraints. We will illustrate this approach using an example. Consider, for instance, an additional side constraint of the form:

$$\sum_{(i,j)\in A'} y_{ij} \geq a_0 \qquad (14.3)$$

where A' is some subset of arcs belonging to the given graph, and a_0 is a positive integer, with $a_0 \leq |A'|$.

The Cutset inequalities that we described in Chapter 3 belong this class of constraints. Let r_0 be the dual variable corresponding to constraint (14.3). Then, the dual of the LP relaxation for this constrained problem

(1) contains the additional term $+ a_0 r_0$ in the objective function, and

(2) assuming, for convenience, that all arcs are uncapacitated, the constraints (8.3) of [DUND] corresponding to all the arcs $(i,j)\in A'$ are now of the form

$$\sum_k w_{ij}^k + r_0 \leq F_{ij} \qquad \text{for all } (i,j)\in A. \qquad (8.3a)$$

One dual ascent procedure for this constrained NDP consists of two phases; the first phase uses the UNDP dual ascent algorithm to modify the w and v values, for a given value of r_0, while the second phase adjusts r_0. Let us examine the possible steps in the second phase. Notice that increasing r_0 by 1 unit increases the objective function by a_0 units and reduces the slack s_{ij} in each of the constraints (8.3a) by 1 unit. Hence, the following two steps can be used in Phase 2:

<u>r_0-increasing step</u>:

If $s_{ij} > 0$ for all arcs $(i,j)\in A'$, we must obviously increase r_0 by Min $\{s_{ij}: (i,j)\in A\}$ at least. It might be worthwhile to increase r_0 further, by reducing one or more of the w_{ij}^k values, if this w-reduction does not decrease $\Sigma_k v_{D(k)}^k$ by more than (a_0-1) units. In particular, suppose fewer than a_0 of the arcs belonging to the set A' have zero slacks. For each such arc, if we arbitrarily reduce any

currently positive w^k_{ij} value by 1 unit, then $v^k_{D(k)}$ can decrease by at most be 1 unit. Consequently, for any zero slack arc (i,j) in A', the total decrease in z_D when w^k_{ij} decreases for some $k \epsilon K$ is less than a_O; however, we can now increase r_O by 1 unit and achieve a net increase in z_D. Therefore, if we let r' be the $(a_O-1)^{th}$ smallest value of the slack s_{ij} among all the arcs $(i,j) \epsilon A'$, then r_O must be increased by at least r'. Further increases in r_O might be beneficial if we can identify a subset of commodities K', such that when one or more w values corresponding to these commodities decreases (in order to permit an increase in r_O), the consequent total decrease in the shortest path lengths is less than a_O.

r_O-decreasing step:

We must also evaluate the impact of decreasing r_O on the dual objective function value. Observe that, when r_O decreases by 1 unit, the slack in each of the constraints (8.3a) increases by 1 unit. This slack can then be utilized to increase one or more w^k_{ij} value, thus possibly increasing z_D by more than a_O units. For instance, if some zero slack arc of A' is critical for more than (a_O+1) commodities, then increasing the respective w^k_{ij} values by 1 unit and decreasing r_O by 1 unit yields a net increase in the dual objective function value. There are numerous other similar situations where decreasing r_O could result in dual ascent.

This two-phase strategy can be extended to handle general side constraints with arbitrary coefficients. The general underlying principle is to modify, in the first phase, the original w and v variables using the UNDP dual ascent procedure. Subsequently, the second phase adjusts the additional dual variables corresponding to the side constraints so that the dual objective function increases further. The algorithm alternates between these two phases until the dual objective function value cannot be increased further. As is evident from the previous example, it is difficult to specify exactly the different ascent steps that must be incoporated in the second phase when the additional constraints are assumed to take general forms. In contrast, for the capacity constraints, we were able to devise efficient and comprehensive procedures that accounted for all the possible alternatives for locally modifying the t_{ij} variables.

Instead of attempting to heuristically modify the additional dual

variables in Phase 2, we could alternatively formulate the problem of increasing the dual objective function value by changing these variables as a subsidiary optimization problem. This approach might be attractive, especially if in Phase 2 we keep some of the variables (such as the w values derived in Phase 1) fixed. We do not pursue this approach further here. Next, we discuss dual-based problem reduction procedures for the Network Design problem.

4.5 Problem Reduction Procedures for the Network Design Problem

For any given optimization problem, we use the term PROBLEM REDUCTION to refer to the process of fixing some of the variables before finding the optimal solution of the problem. In the Network Design context, we can reduce the problem by identifying arcs that definitely do or do not not belong to the optimal design and determining, if possible, whether certain commodities flow on some arcs in the optimal routing. Fixing variables in this manner reduces the problem size, and improves solution times. In addition, the lower and upper bounds for the 'reduced' problem are likely to be tighter than those obtained using the original problem formulation. The importance of problem reduction procedures for the Network Design problem is borne out by the successful experience of Magnanti et al. [1984] in solving Uncapacitated Network Design problems. Using preprocessing methods of the kind that we will discuss in this section, they were able to solve to optimality 7 out of their 24 test problems (whose sizes ranged upto 45 binary variables and 105,600 continuous variables or 90 binary variables and 15,080 continuous variables); for the

remaining problems, they were able to fix between 75 to 95 percent of the design variables, and thus produce residual problems that were small enough to be solved by branch-and-bound procedures (although they used Benders' decomposition instead). Just like procedures for generating valid inequalities, problem reduction can be achieved either by using objective function information and exploiting special properties of optimal solutions or by examining the constraint set alone. (Indeed, problem reduction methods can be viewed as techniques for generating special types of valid inequalities). Guignard and Spielberg [1981] and Crowder et al. [1983] have described preprocessing procedures (that generate logical inequalities besides fixing variables) based solely on the structure of the problem constraints in general zero-one programming problems. On the other hand, Balakrishnan [1982] has devised a preprocessing procedure for the Steiner Network problem that identifies several arcs and Steiner points that do and do not belong to the optimal solution by solving a series of related Minimal Spanning tree problems. For the Network Design problem, Billheimer and Gray [1973] and Magnanti et al. [1984] have developed problem reduction methods that rely on objective function information.

In this section, we will primarily consider objective function-based problem reduction methods. We focus on procedures that use the dual ascent solution to fix design and flow variables of the NDP. We also briefly examine extensions of these methods for generating valid inequalities for the NDP. This development complements our earlier discussion (in Chapters 2 and 3) on cuts based solely on the structure of the NDP constraint set. Even though our discussion is cast mainly in terms of the UNDP, most of the methods that we outline are applicable to the capacitated problem as well.

The only exceptions are those methods that rely on certain special properties of UNDP optimal solutions (for instance, when the cost coefficients are non-negative, the uncapacitated problem has an optimal solution in which each commodity flows on a single path).

In this section, we consider the following four types of methods:

Reduction 1: identifying arcs that do not belong to the optimal design.

Reduction 2: determining if commodity k must flow on arc (i,j) and if arc (i,j) must necessarily belong to the network design in the optimal solution.

Reduction 3: determining if commodity k must not flow on arc (i,j) in the optimal solution.

Other problem reduction and preprocessing methods: includes fixing design and flow variables based on the network topology.

At every stage of the problem reduction procedure, for each commodity k, we maintain three sets of arcs - Open(k), Closed(k), and Free(k). Open(k) and Closed(k) are the sets of arcs on which commodity k must and must not flow, respectively, in the optimal NDP solution. Free(k) consists of all arcs on which commodity k's flow is not yet fixed. Both Open(k) and Closed(k) are empty at the start of the algorithm, while Free(k) initially contains all arcs of the original network. We update these sets after each problem reduction step.

The first three methods are based on the dual ascent solution; they also require a known upper bound z^+ (say the value of the current incumbent) on the optimal value of the NDP. We illustrate the dual-based reduction principle with the following example. Suppose, we want to test if arc (i,j) must be EXCLUDED from the Network Design (i.e., if arc (i,j)

does not belong to the optimal design). For this purpose, we determine, using the dual ascent solution, a lower bound on the optimal value when the 'opposite' constraint $y_{ij} = 1$ is added to the primal problem. If this new lower bound is greater than z^+, then arc (i,j) cannot belong to the optimal design. Notice that we only obtain an underestimate of the increase in the lower bound when the opposite constraint is added; in this sense, the problem reduction methods that we consider here closely resemble the penalty methods traditionally used for fathoming in branch-and-bound schemes (see, for example, Garfinkel and Nemhauser [1972]). In the fourth category, we discuss methods for fixing variables based on the structure of the current network and some auxiliary calculations. The four reduction methods can be applied iteratively until they cannot fix any additional variables.

Throughout our discussion, we assume that $v = \{v_i^k\}$, $w = \{w_{ij}^k\}$ is the 'current' dual solution; this solution need not necessarily be the final dual ascent solution, since the problem reduction techniques can be applied at intermediate stages of the dual ascent algorithm as well. As before, z_D denotes the current value of the dual objective function and s_{ij} is the current slack in the dual constraint (9.3) (or (13.3)) corresponding to arc (i,j).

Reduction 1: Identifying arcs that MUST NOT belong to the optimal design.

We use the following test to identify arcs that definitely do not belong to the optimal design.

> For any arc $(i,j) \in A$, if $s_{ij} > (z^+ - z_D)$, then arc (i,j) must be EXCLUDED from the Network Design problem.

Proof:

Suppose arc (i,j) belongs to the optimal network design. Then, the constraint

$$y_{ij} = 1$$

is a valid inequality in the formulation [UNDAF]. Let r_{ij} be the dual variable corresponding to this constraint. Then, the dual constraint corresponding to arc (i,j) becomes

$$\sum_k w^k_{ij} + r_{ij} \leq F_{ij}$$

and the dual objective function becomes

$$\sum_k v^k_{D(k)} + r_{ij} .$$

By setting $r_{ij} \leftarrow s_{ij}$ and $s_{ij} \leftarrow 0$, we see that the new dual objective function value increases to $(z_D + s_{ij})$ which must be a lower bound for the original problem, i.e., $z_D + s_{ij} \leq z^+$. Consequently, if this last condition is violated, arc (i,j) cannot belong to the optimal design.

Magnanti et al. [1984] developed the same test using a Benders' cut derived from the dual solution; we next explain the connection between these approaches. For any given dual solution, we can construct a Benders' cut of the form

$$z \geq \sum_k v^k_{D(k)} + \sum_{(i,j)} (F_{ij} - \sum_k w^k_{ij}) y_{ij},$$

where z is the optimal value of the Network Design problem. Notice that the first term in the right-hand side is the current dual objective function value z_D. Also, the coefficient of each design variable y_{ij} is non-negative since the solution (v,w) is dual feasible. Now, if arc (i,j) belongs to an optimal design, we can set y_{ij} equal to 1, and the constant term z_D in the right-hand side of the Benders' cut increases by

$$F_{ij} - \sum_k w^k_{ij} = s_{ij} .$$

Therefore, if $s_{ij} > z^+ - z_D$, arc (i,j) cannot belong to the optimal design.

This reduction method can be modified for generating more general

valid inequalities for the NDP in the following manner:

If, for any subset of arcs $A' \subseteq A$,

$$\sum_{(i,j) \in A'} s_{ij} > z^+ - z_D,$$

then

$$\sum_{(i,j) \in A'} y_{ij} \leq |A'| - 1$$

is a valid inequality for the Arc-path formulation of the NDP.

In particular, suppose we arrange the arcs in order of non-increasing slacks s_{ij}. If the sum of the slacks for the first p arcs exceeds the difference $(z^+ - z_D)$, then at most (p-1) of these arcs can belong to the optimal design; to obtain the tightest constraint of this form, we must select the smallest value of p satisfying this condition. Finally, note that when $|A'| = 1$, we obtain the previous reduction test as a special case of this cut-generating procedure.

This reduction test, like all others that follow, will be most effective when z_D and z^+ are 'nearly' equal. In other words, the amount of reduction achieved depends critically on the tightness of the upper and lower bounds that we generate. Also, this test justifies the subsidiary objective of maximizing the slacks that we employed earlier in the dual ascent procedure (both in the Composite method and the v-improvement method); the higher the slack, for a given dual objective function value, the greater are the chances that the corresponding design variable can be eliminated by this method.

Reduction 2: Identifying flows that must flow on certain arcs and arcs that must belong to the optimal design.

We now outline a test for determining if commodity k must necessarily

flow on arc (i,j) (i.e., if $x^k_{ij} = 1$) in the optimal routing and if arc (i,j) necessarily belongs to the optimal design (i.e., if $y_{ij} = 1$). For this purpose, we define, for all arcs $(i,j) \in A$ and commodities $k \in K$ the following parameters:

p^k_{ij} = the length of the shortest path from O(k) to D(k) excluding arc (i,j), using the Modified arc lengths e^k_{ij},

$g^k_{ij} = p^k_{ij} - v^k_{D(k)} \geq 0$, and

$h_{ij} = \sum_k g^k_{ij}$

Notice that, g^k_{ij} is positive if and only if arc (i,j) belongs to <u>all</u> the current shortest paths for commodity k (i.e., if arc (i,j) is critical for commodity k); in fact, g^k_{ij} is the maximum increase in $v^k_{D(k)}$ that is possible by increasing the modified arc length e^k_{ij}. Then,

> If $g^k_{ij} > (z^+ - z_D)$, commodity k MUST flow on arc (i,j) and arc (i,j) must be INCLUDED in the optimal design. Furthermore, if $h_{ij} > (z^+ - z_D)$, then arc (i,j) must necessarily belong to the optimal design.

Notice that the second part of this test is useful only if the first part fails for all commodities. We will prove the validity of this result in two ways - one using the dual problem directly, as we did for Reduction 1, and the other using a Lagrangian relaxation of the original problem. We include both proofs mainly to demonstrate the close relationship between our dual ascent procedure and the Lagrangian relaxation methods that we considered in Section 4.2.

<u>Proof</u>:

Consider adding the constraint

$$x^k_{ij} \leq 0$$

that must be satisfied if commodity k does <u>not</u> flow on arc (i,j) in the optimal NDP solution. Let q^k_{ij} be the dual variable for this constraint. Then, the dual constraint (9.2) corresponding to arc (i,j) and commodity k becomes

$$v_j^k - v_i^k \leq c_{ij}^k + w_{ij}^k + q_{ij}^k;$$

the dual objective function remains unaltered. Notice that the Modified length e_{ij}^k of arc (i,j) for commodity k is now $(c_{ij}^k + w_{ij}^k + q_{ij}^k)$. Hence, if we set q_{ij}^k equal to g_{ij}^k, the shortest path length $v_{D(k)}^k$, and hence the dual objective function value, increases by g_{ij}^k. Consequently, z^+ must be greater than or equal to $(z_D + g_{ij}^k)$ if commodity does not flow on arc (i,j). Similarly, if we add the constraint

$$y_{ij} \leq 0,$$

we can show that the new dual objective function value must be at least

$$z_D + \sum_k g_{ij}^k = z_D + h_{ij} ,$$

implying that the z^+ must be greater than this quantity if arc (i,j) does not belong to the optimal design. Therefore, this reduction test is valid.

We will now prove this result by relating the dual ascent procedure to a Lagrangian relaxation scheme for [UNDAF]. Suppose we dualize constraints (1.3) of [UNDAF] using multipliers w_{ij}^k. Then, the corresponding Lagrangian subproblem LR(w) is

[LR(w)]

$$\text{Minimize} \quad \sum_k \sum_{(i,j)} (c_{ij}^k + w_{ij}^k) x_{ij}^k + \sum_{(i,j)} (F_{ij} - \sum_k w_{ij}^k) y \; U$$

subject to

$$\sum_j x_{ij}^k - \sum_j x_{ji}^k = \begin{cases} +1 & \text{if } i = O(k) \\ -1 & \text{if } i = D(k) \\ 0 & \text{otherwise} \end{cases} \qquad \text{for all } k \epsilon K$$

$$x_{ij}^k \geq 0, \; y_{ij} \text{ integer} \qquad \text{for all } (i,j) \epsilon A, \; k \epsilon K$$

For any given values of w_{ij}^k, an optimal solution of [LR(w)] can be determined as follows:

For each commodity $k \epsilon K$,
set x_{ij}^k equal to 1 if arc (i,j) belongs to the shortest path from O(k) to D(k), using the 'modified' arc lengths e_{ij}^k; otherwise, $x_{ij}^k = 0$, and
set y_{ij} equal to 1 if $(F_{ij} - \Sigma_k w_{ij}^k) < 0$; otherwise, $y_{ij} = 0$.

Notice that when the w^k_{ij} values obtained using the dual ascent procedure are used as multipliers in this Lagrangian relaxation, all the design variables y_{ij} are set equal to 0 in the corresponding optimal subproblem solution. Also, the subproblem objective function value, $z_{LR}(w)$ is then equal to z_D. Suppose, we add the constraint $x^k_{ij} = 0$ to this Lagrangian problem. Then, in the Lagrangian subproblem solution, commodity k would be routed on the shortest path from O(k) to D(k) that does not inclued arc (i,j) (using e^k_{ij} as arc lengths). Therefore, when the additional constraint is incorporated, the Lagrangian objective function value increases by g^k_{ij}. Hence, if $z_D + g^k_{ij} > z^+$, commodity k must necessarily flow on arc (i,j). We can similarly prove the second part of the reduction rule.

If this reduction step is successful, i.e., if we are able to identify an arc (i,j) that must necessarily belong to the optimal design using this test, then all the variables w^k_{ij} can be eliminated from the dual problem. The Modified arc length e^k_{ij} is then equal to c^k_{ij} for all commodities; the shortest path lengths $v^k_{D(k)}$ must be recalculated using these new values of e^k_{ij}, and the fixed cost F_{ij} must be added to the dual objective function value.

Again, we can extend this test to generate valid inequalities for the NDP. For any given subset of arcs A', let

$g^k_{A'}$ = length of shortest path from O(k) to D(k) excluding arcs belonging to A', using e^k_{ij} as arc lengths, for all $k \epsilon K$, and

$$h_{A'} = \sum_k g^k_{A'}$$

Then, the following result is valid.

If for any subset of arcs A',

$$g^k_{A'} > (z^+ - z_D)$$

for some commodity k, then every optimal UNDP solution must satisfy the inequalities

$$\sum_{(i,j)\in A'} x^k_{ij} \geq 1 \quad \text{and} \quad \sum_{(i,j)\in A'} y_{ij} \geq 1.$$

Further, if

$$h_{A'} > (z^+ - z_D),$$

then, the inequality

$$\sum_{(i,j)\in A'} y_{ij} \geq 1$$

is valid for the UNDP.

One way to identify such a subset A' is to sequentially choose one arc from the shortest path, the shortest path excluding the first arc, and so on until the length of the shortest path in the remaining network increases by more than $(z^+ - z_D)$.

Reduction 3: Identifying commodities that MUST NOT flow on certain arcs

Let g'^k_{ij} be the length of the shortest path from O(k) to D(k) via arc (i,j), using the Modified arc lengths e^k_{ij}. Then,

(a) if for some arc (i,j) that is free (i.e., arc (i,j) is not yet definitely included in or excluded from the optimal design),

$$g'^k_{ij} + s_{ij} > (z^+ - z_D),$$

then commodity k MUST NOT flow on arc (i,j), and

if for some arc (i,j) that is fixed open (i.e., arc (i,j) has been identified as an arc belonging to the optimal design),

$$g'^k_{ij} > (z^+ - z_D),$$

then commodity k MUST NOT flow on arc (i,j).

Proof:

Consider again the Lagrangian subproblem [LR(w)] corresponding to the set

of multipliers $w = \{w_{ij}^k\}$ obtained from the dual ascent solution. If commodity k flows on arc (i,j) in the optimal NDP solution, the constraints

$$x_{ij}^k = 1, \text{ and}$$
$$y_{ij} = 1$$

can be added to this subproblem, and the new Lagrangian objective function value must be a valid lower bound for the NDP. Now, if arc (i,j) is free, $z_{LR}(w)$ increases by $(g'^k_{ij} + F_{ij} - \Sigma_k w_{ij}^k) = (g'^k_{ij} + s_{ij})$; if arc (i,j) is fixed open, $z_{LR(w)}$ increases by just g'^k_{ij}. Hence, this reduction test is valid.

Again, the inequality

$$\sum_{(i,j)\in A'} x_{ij}^k \leq |A'| - 1$$

is valid for the Arc-flow formulation of the UNDP if, for any commodity k and arc subset A',

$$\sum_{(i,j)\in A'} g'^k_{ij} + \sum_{(i,j)\in A''} s_{ij} > z^+ - z_D,$$

where A" is the set of all arcs belonging to A' that are currently free. This inequality is, however, valid only for the UNDP, since in the optimal CNDP solution each commodity could flow over multiple paths. Also, unlike the case when $|A'| = 1$, there does not seem to be any efficient way to determine the shortest path that must contain more than one designated arc (except if these arcs constitute a directed subpath). Alternatively, let K' be a subset of commodities such that the g'^k_{ij} values corresponding to these commodities satisfy the condition

$$\sum_{k\in K'} g'^k_{ij} + s_{ij} > z^+ - z_D \quad \text{if arc (i,j) is free, or}$$

$$\sum_{k\in K'} g'^k_{ij} > z^+ - z_D \quad \text{if arc (i,j) is fixed open.}$$

Then

$$\sum_{k\in K'} x_{ij}^k \leq |K'| - 1$$

is valid for the NDP. In fact, if the subset K' consisting of the p

commodities with the smallest values of g'^{k}_{ij} satisfies this condition, then the following 'capacity' constraint

$$\sum_{k \in K'} x^{k}_{ij} \leq p - 1$$

is valid for the Arc-flow formulation of the NDP. Lamar [1983] has used similar valid 'capacity' inequalities to strengthen the weak Arc-flow formulation of the Uncapacitated Network Design problem.

Other Problem Reduction and Preprocessing methods:

The following obvious reduction tests identify additional flow and design variables that can be fixed to zero or one.

(1) Since the cost coefficients c^{k}_{ij} and F_{ij} are assumed to be non-negative, commodity k must not flow on any arc incident to the origin O(k) or incident from the destination D(k).

(2) For each commodity k, if using the arcs in the set Open(k) U Free(k) there is no path either from O(k) to node i or from node i to D(k), then commodity k must not flow on any arc (i,j) or (1,i) belonging to the set Free(k).

(3) For any arc (i,j) belonging to the set Open(k), if there is exactly one arc belonging to the set Free(k) (and none in Open(k)) that is incident to node i (or incident from node j), then commodity k must flow on this arc and this arc must belong to the optimal design.

(4) We know that the UNDP has an optimal solution in which each commodity

flows on a single path. Therefore, if arc (i,j) belongs to the set Open(k), then commodity k can be prohibited from flowing on all other arcs (l,j) and (i,l) belonging to the set Free(k) that are incident from node i or incident to node j.

(5) For any commodity k, let P_k and P_k' be the shortest and second shortest paths from O(k) to D(k), using the flow costs c_{ij}^k as arc lengths (and containing only arcs of Open(k) U Free(k)). We let $l(P_k)$ and $l(P_k')$ denote the lengths of these two paths. Let F_k be the sum of the fixed costs of arcs on P_k that do not belong to Open(k). Then, if

$$l(P'_k) > l(P_k) + F_k,$$

commodity k must flow on path P_k and all arcs belonging to this path must belong to the optimal design. This test is likely to be particularly useful at intermediate nodes of a branch-and-bound tree, when several design variables are fixed at zero or one.

(6) As before, let P_k be the shortest path from O(k) to D(k), using c_{ij}^k as the lengths of arcs (i,j) belonging to the set Open(k) U Free(k). Suppose we are given a heuristic solution, and let (i,j) be any arc belonging to the heuristic design. Let l_{ij}^k denote the cost of the shortest path excluding arc (i,j) (using the routing costs c_{ij}^k) in the current design. Then,

$$h_{ij} = \sum_k (l_{ij}^k - l(P_k))$$

is the maximum possible reduction in the total flow cost if arc (i,j) is included in the final design. Hence, if $(h_{ij} - F_{ij})$ is negative, arc (i,j) must not belong to the optimal design.

The problem reduction tests that we have just described can be applied repeatedly until no more reduction is obtained. Also, as we mentioned earlier, when applied to the reduced problem, the dual ascent procedure, is likely to improve the lower bound further. Similarly, the heuristic procedure (with appropriate modifications to account for arcs on which each commodity must flow) might yield better local minima after problem reduction. At the same time, the problem reduction procedure is most effective when the upper and lower bounds are tight. This interrelationship suggests that the three components - dual ascent, problem reduction, and heuristic - should be combined into a composite iterative procedure. We incorporated all three phases in the algorithms that we tested both for the UNDP and the LTL problem (discussed in Chapter 5).

4.6 Computational Results for the Uncapacitated Network Design Problem

We tested the dual ascent algorithm, together with the problem reduction and heuristic procedures, on two types of randomly generated directed, uncapacitated Network Design problems. The problem sizes varied from 10 nodes, 10 commodities, and 43 arcs to 25 nodes, 50 commodities, and 313 arcs. The integer programming formulation for the largest of these test problems contains 313 binary variables, and 15,650 continuous variables. In this section, we describe the details of our implementation and report on the computational results. We interpret these results and indicate ways in which the implementation could be made more efficient.

4.6.1 Generating random problems:

The two types of problems that we used for testing, designated as Type A and Type B problems, respectively, differ primarily in their demand and cost structures. We used two separate FORTRAN programs to generate problem instances of each type. We describe the structure of these programs below.

Type A problems:

The input parameters for this type of problem include the

Number of NODES,
Number of COMMODITIES,
DENSITY of arcs,
parameters for the FLOW COST for each arc and commodity,
parameters for the FIXED COST for each arc, and
SEED for the Random number generator.

The network generator first randomly locates the nodes of the network on a 100 x 100 grid. It then randomly selects, for each commodity, an origin and destination from the given set of nodes. Every time an origin-destination pair is generated, the program ensures that the same pair has not been chosen for some other commodity earlier; also, if the Euclidean distance between the origin and destination is below a given threshold value (which was initially set at 50), a new origin-destination pair is chosen. If no such pair exists, the program repeats the procedure after lowering the threshold distance. We specify such a threshold distance in order to increase the likelihood that the optimal route for each commodity contains more than one arc; consequently, several commodities are likely to share arcs in the optimal routing, rendering the problems more difficult to solve. Having determined the origins and destinations for all commodities, the program selects the arcs of the network randomly from the set of all possible node pairs. It uses the input parameter, DENSITY of arcs, as the probability with which any particular node pair must be chosen; thus, this parameter indirectly

determines the number of arcs in the network. In order to ensure that the problem has at least one feasible solution, the procedure adds 'direct' origin-destination arcs, for each commodity. Next, the variable costs c^k_{ij} for each commodity and the Fixed cost F_{ij} are determined for each arc. By appropriately specifying the cost parameters, we can generate a wide variety of cost functions - proportional directly or inversely to the Euclidean distance, with or without constant and random components, and so on. However, in all the Type A problem instances that we tested, we set c^k_{ij} equal to the Euclidean length of arc (i,j) and F_{ij} equal to $2*c^k_{ij}$. The parameters for the five Type A problem instances that we generated are presented in Table 2. These instances are labeled as problems A1 to A5.

Type B problems:

Type A and Type B problems differ primarily in their demand patterns and cost structures. The procedure for generating Type B problems is almost identical (except for the fact that we consider directed rather than undirected networks) to the one used by Magnanti et al. [1984]. As before, nodes of the network are first randomly positioned on a grid. However, instead of random Origin-Destination pairs, Type B problems have either an All-to-All or a Two-to-All demand pattern. In the former pattern, one unit of flow has to be transported from each node to all other nodes of the network, while in the latter pattern, the node closest to the origin and the one that is the farthest away from the origin are chosen as the two origins from which flow must be routed to each of the other nodes. Another distinction vis-a-vis Type A problems is that the NUMBER of arcs rather than the DENSITY is specified as an input parameter; also, the Euclidean lengths of the arcs in the network are required to be smaller

Table 2: Uncapacitated Network Design Test Problem Parameters

Problem Name \ Parameter		No. of Nodes	No. of Comm.	No. of Arcs		Density of arcs	LMAX	Flow Cost c_{ij}^k*	Flow Cost F_{ij}
				Total	Design				
Type A Problems	A1	10	10	43	43	0.5	-	$0.25*dist_{ij}$	$2*c_{ij}^k$
	A2	20	20	200	200	0.5	-	$0.25*dist_{ij}$	$2*c_{ij}^k$
	A3	20	20	144	144	0.3	-	$0.25*dist_{ij}$	$2*c_{ij}^k$
	A4	25	50	313	313	0.5	-	$0.25*dist_{ij}$	$2*c_{ij}^k$
	A5	25	50	216	216	0.3	-	$0.25*dist_{ij}$	$2*c_{ij}^k$
Type B Problems	B1	10	18	90	70	-	500	$1.0*dist_{ij}$	$110-c_{ij}^k$
	B2	10	90	90	70	-	500	$1.0*dist_{ij}$	$110-c_{ij}^k$
	B3	15	28	120	90	-	70	unif(50,150)	$10*c_{ij}^k$
	B4	25	48	260	180	-	50	unif(50,150)	unif(500,1000)

Notes: LMAX = Maximum permissible arc length (for Type B problems only)

$dist_{ij}$ = Euclidean distance between nodes i and j

Unif (a,b) = random variable that is uniformly distributed between a and b

* Flow cost is the same for all commodities.

than a maximum threshold length LMAX that is specified by the user. The procedure proposed by Knuth [1969] is used to randomly select the required number of such arcs from the set of all node pairs. In Type B problems, arcs of the network are classified into two categories - Design arcs and arcs that are Open. Open arcs are those that already belong to the network design, and their fixed costs are zero. The design decision is, therefore, to determine the subset of design arcs that must be included in the optimal configuration. Thus, the number of binary variables in the IP formulation for Type B problems is equal to the number of design arcs that are included in the original network. The network generator chooses the specified number of design arcs randomly from the set of all arcs that have been included in the network. Unlike Type A test problems, we specified different cost structures for each of the Type B test problems. Table 2 gives the problem parameters for the four Type B instances that we tested. These instances are labeled as problems B1 to B4.

For convenience, we appended a preprocessing routine to both these network generating programs. For each node i and each commodity k, this routine determines whether the given network contains at least one path from O(k) to node i and from node i to D(k). If not, node i is flagged so that the flow variables x^k_{ij} and x^k_{ji}, corresponding to all arcs (i,j) and (1,i) of the network that are incident to and from node i, can be eliminated from the problem formulation. However, for our test problems, this routine did not eliminate any variables.

Generating the initial heuristic solution:

Before initiating the dual ascent procedure (and the embedded heuristic algorithm), we used a 'stand-alone' procedure to obtain an initial heuristic solution. Unlike the heuristics that we discussed earlier, this algorithm does not rely on dual ascent solutions to generate the initial feasible solution. However, the subsequent local improvement methods used in the algorithm are the same as those outlined before (ADD, DROP, and FLOW REROUTING). We tested several combinations of the different initialization and local improvement schemes on the 9 randomly generated problems. Before discussing the results of this exercise, we briefly describe the two alternative initialization procedures that we considered.

The first method, that we call the SEQUENTIAL ROUTING METHOD, proceeds as follows. At stage t of the procedure, (t-1) commodities have been routed, and the t^{th} commodity must be selected and routed. For each commodity k that has not yet been routed, the method first determines the shortest path from O(k) to D(k) using the 'incremental' arc costs (i.e., c^k_{ij} if arc (i,j) belongs to the current 'partial' design, $c^k_{ij} + F_{ij}$ otherwise). Then, it chooses the commodity with the smallest incremental routing cost as the next commodity to be routed; this commodity is routed along its shortest path and all arcs of this path that do not belong to the current partial design are added to the design. Finally, when all commodities have been routed, the procedure determines the best routing for each commodity (using just the flow costs c^k_{ij}) along the arcs of the final design and reroutes the corresponding flow if necessary. Any arcs in the final design along which no commodity is routed are discarded.

The second method, which we designate as the ALL-INCLUSIVE method, starts with an initial design containing all the arcs of the network. Each commodity is routed along its shortest path in the network (using c^k_{ij} as arc lengths). Arcs on which no commodity is routed are then removed from the design.

To obtain locally optimal solutions starting with the initial feasible solution, we tested two of the local improvement procedures discussed earlier (in Section 4.3.4) - the Flow Rerouting method and the Drop-Add procedure. Starting with the Drop phase, the Drop-Add method alternately applies Add and Drop phases until the heuristic solution does not improve in two successive phases. We also tested procedures in which the Drop-Add and Flow Rerouting methods are applied in tandem.

We list below the initialization and local improvement schemes used in the four heuristic procedures that were tested:

HEUR1: Initialization by Sequential routing method.
Local improvement by Flow Rerouting method.

HEUR2: Initialization by All-inclusive method.
Local improvement by Drop-Add method.

HEUR3: Initialization by Sequential routing method.
Local improvement by Flow Rerouting/Drop-Add cycles.

HEUR4: Initialization by All-inclusive method.
Local improvement by Drop-Add/Flow Rerouting cycles.

All the heuristic procedures were coded in FORTRAN; no special data structures or updating schemes were employed. Table 3 summarizes the results using these four stand-alone heuristic algorithms for the 9 test problems. Each cell of the table indicates both the cost of the final heuristic solution (or upper bound) and the CPU time (in terms of secs on

Table 3: Performance of different heuristic procedures

Problem Name	Cost of init soln Sequen	Cost of init soln All-incl	Cost of heur solution using HEUR1	HEUR2	HEUR3	HEUR4	Best upper bound *
A1	391	484	381 (4)	448 (4)	381 (4)	393 (4)	381
A2	731	950	719 (34)	928 (54)	719 (73)	746 (84)	668
A3	767	950	741 (30)	929 (33)	741 (48)	766 (53)	740
A4	1678	2261	1598 (305)	1846 (539)	1579 (699)	1593 (767)	1441
A5	1744	2275	1711 (183)	2030 (222)	1671 (473)	1716 (571)	1595
B1	1529	1777	1503 (12)	1679 (20)	1503 (19)	1571 (22)	1352
B2	6344	6918	6142 (136)	5221 (134)	6081 (164)	5133 (163)	5133
B3	12298	27338	12297 (39)	16572 (68)	12297 (62)	14074 (77)	9718
B4	12692	33836	12692 (173)	21294 (633)	12692 (427)	13897 (1070)	11541

Figures in parentheses are CPU secs on PRIME 850.

* Best upper bound obtained using dual-based heuristic procedure.

the PRIME 850). The table also includes, as a measure for comparison, the best upper bound derived using the dual ascent-based heuristic (which is described later). The second method, HEUR2, did not find the best initial feasible solution in any of the 9 problem instances; the fourth method, HEUR4, found the best solution in only 1 instance (problem B2). HEUR3 gave better solutions than HEUR1 in all problem instances; this result is to be expected since HEUR3 is an enhancement of HEUR1. However, the Drop-Add phase of this algorithm was effective only in the three problems A4, A5, and B2; in the other problems, this phase did not improve upon the final solution of the Flow rerouting phase. For Problem A5, the locally optimal solution was found after two Flow Rerouting/Drop-Add cycles while only one such cycle was required for Problems A4 and B2. On average, the local improvement procedure in HEUR3 improved the initial heuristic solution by about 3.3 percent (i.e., the cost of the initial solution as a percentage of the final heuristic solution was 103.3 on average). The cost of the best initial heuristic solution as a percentage of the best final upper bound (i.e., the best upper bound obtained by the dual ascent-based heuristic) was higher for Type B problems (about 112 percent) than for Type A problems (about 104.4 percent). This phenomenon probably occured because Type B problems have much higher fixed costs, relative to flow costs, compared to Type A problems; therefore, the Type B problems that we used probably had many more 'dispersed' local optima. The 'stand-alone' heuristic solution had, on average (over all the 9 test problems), a 7.7 percent higher cost compared to the best upper bound obtained finally. Since HEUR1 produced fairly good heuristic solutions and since HEUR3 requires almost twice as much computation time to improve the HEUR1 solution, we decided to use just the Flow Rerouting method for local

improvement within the dual ascent framework. Next, we describe the implementation of the dual ascent algorithm that we tested.

The Dual Ascent algorithm:

After a preliminary test of various alternative dual ascent schemes, we decided to use an algorithm with the following structure:

Part 1: Step 1: Increase w values (using Simultaneous method)
Step 2: Update v values (using Forward shortest path method)
Decrease intermediate v values (using v-improvement method)
Step 3: Update w values

Part 2: Flow Rerouting Heuristic with Reference solution as initial feasible solution.

Part 3: Decrease w values to reduce Complementary slackness violations.

Part 4: Interchange w values to reduce Complementary slackness violations.

Part 1 is the basic procedure that we outlined earlier. The algorithm transfers control to Part 2 only if either the termination criterion is satisfied in the w-increasing or v-increasing step or the number of iterations within Part 1 exceeds a user-specified limit. The Reference solution, in which each commodity is routed along a current zero-slack shortest path (using the modified arc lengths e^k_{ij}) serves as the starting point for the Flow Rerouting local improvement procedure. Subsequently, in Part 3, w values which violate complementary slackness conditions (with respect to the reference solution) are reduced if possible using the method described in Section 4.3.4. If no w value is reduced in Part 3, the program transfers control to Part 4; otherwise, it reinitiates the basic procedure in Part 1 with the updated w values. Part 4 consists of a

procedure to interchange w values corresponding to the same arc but different commodities (i.e., to decrease some values and increase others) in order to reduce some complementary slackness violations. Again, if Part 4 modifies any w values, Part 1 is reinitiated. Otherwise, the algorithm terminates. The user can also specify the maximum number of times Part 1 can be initiated in any given run of the algorithm. The problem reduction phase was deliberately not included as part of this algorithm, so that iterations between the problem reduction phase and the dual ascent phase can be controlled manually. The problem reduction routine included almost all the methods described in Section 4.5. (The only tests that were omitted were (5) and (6) described under 'Other methods'). Valid inequalities such as those discussed in Chapter 3 and Section 4.5, were not added to the original formulation.

All the procedures were programmed in FORTRAN and no special data structures or updating techniques were employed. The primary objective of the computational tests was to determine the tightness of the lower and upper bounds that are obtained using this composite procedure. We did not embed this procedure in a branch-and-bound or enumeration scheme.

For all problem instances, we use the following initial dual solution:

$w_{ij}^k = 0$ for all $(i,j) \epsilon A$, $k \epsilon K$,

$s_{ij} = F_{ij}$ for all $(i,j) \epsilon A$, and

v_i^k = length of shortest path using c_{ij}^k as arc lengths from $O(k)$ to node i, for all $i \epsilon N$, $k \epsilon K$.

Table 4 summarizes the computation times as well as the bounds that were obtained for each of the 9 problem instances. For those problems that

required a second run after problem reduction, the two corresponding runs are shown separately. For all the runs, a maximum of 50 iterations within Part 1 (each time Part 1 is initiated) and a maximum of 20 Part1-Part2 cycles were permitted. The problem reduction results (computation times and extent of reduction achieved) are presented separately in Table 5.

Observations and interpretation of the results:

The best lower bound (i.e., final dual objective function value) as a percentage of the best upper bound (i.e., cost of the best heuristic solution) ranges from 95.4 percent to 100 percent, with an average of 98.1 percent. The CPU time required for ascent and for the heuristic procedure are of comparable magnitudes, while the time required for problem reduction (mainly for Type A problems) is significantly higher. This is probably due to the fact that the problem reduction procedure requires extensive shortest path computations. As expected, the effectiveness of the problem reduction procedure seems to depend critically on the gap between the best upper and lower bounds. Also, except for problems B3 and B4, the tests for identifying arcs that must definitely belong to the optimal design and commodities that must necessarily flow on certain arcs (Reduction 2) were not successful. In all instances, the dual-based heuristic procedure improved the initial lower bound. A detailed examination of the results, especially for Type A problems, revealed that a large fraction of the increase in the dual objective function is achieved in the first few cycles of the dual ascent algorithm. Subsequently, the algorithm continues to cycle through all the parts of the procedure mainly because new w values are reduced in Parts 3 and 4 rather than because of any increase in z_D in

Table 4: Computational Results for the Uncapacitated Network Design Problem

Statistic \ Problem		A1	A2	A3	A4	A5	B1	B2	B3	B4
1. Final Lower Bound z_D	Run 1	372.6	649.5	709.4	1374.4	1539.9	1335	5086.2	9676.2	11541
	Run 2	-	653.3	715.7	-	1549.0	-	5101.6	-	-
2. Best Upper Bound $\hat{z}$		381	668	740	1441	1595	1352	5133	9718	11541
3. % $\frac{\text{Best Lower Bound}}{\text{Best Upper Bound}}$		97.8%	97.8%	96.7%	95.4%	97.1%	98.7%	99.4%	99.6%	100%
4. CPU times (secs)* for Dual Ascent:	Run 1	14.8	316.6	255.1	2294.3	1087.3	27.67	149.6	143.6	499.2
	Run 2	-	151.2	102.9	-	486.2	-	15.0	-	-
(b) % $\frac{\text{CPU time for Part 1}}{\text{Total time for ascent}}$		54.1%	48.8%	55.8%	67.2%	61.9%	81.2%	80.0%	69.1%	77.3%
(c) for Heuristic:	Run 1	11.8	209.6	159.5	2261.5	1066.0	13.0	100.8	89.6	228.4
	Run 2	-	227.2	139.4	-	1204.5	-	12.9	-	-
No. of times Part 1 Initiated	Run 1	18	20	20	20	18	5	7	11	7
	Run 2	-	20	20	-	19	-	3	-	-

* CPU time (secs) on PRIME 850.

Table 5: Problem Reduction for the Uncapacitated Network Design Problem

Statistic \ Problem	A1	A2[2]	A3[2]	A4	A5	B1	B2[2]	B3	B4
1. Original problem:									
# of x-variables	430	4000	2880	15650	10800	1620	8100	3360	12480
# of y-variables	43	200	144	313	216	70	70	90	180
2. Problem Reduction:									
# of x-variables fixed at 0	338	2004	1114	340	982	1481	6848	3282	12368
fixed at 1	-	-	-	-	-	-	-	3	2
# of y-variables fixed at 0	18	39	10	-	-	54	18	87	178
fixed at 1	-	-	-	-	-	-	-	3	2
3. % Reduction in									
x-variables	78.6%	50.1%	38.7%	2.2%	9.1%	91.4%	84.5%	97.8%	99.1%
y-variables	41.9%	19.5%	6.9%	0%	0%	77.1%	25.7%	100.0%	100.0%
4. CPU time for Problem[1] Reduction (secs)	5.6	1134.0	679.0	5321.5	1539.9	14.2	129.4	16.3	115.5

[1] CPU time (secs) on PRIME 850.

[2] Two problem reduction runs were used for these problems; the total reduction obtained and total CPU time are given in the table.

Part 1. The other contributing factor to this cycling phenomenon is the loss of precision and the consequent roundoff errors in the floating point arithmetic operations. Unfortunately, since Step 1 of Part 1 'allocates' the slack s_{ij} equally to several commodities, it was not possible to code the entire program using integer variables alone. There were some noticeable differences between the results for the Type A and the Type B problem instances that we considered. We list some of these differences below:

(1) The percentage gap between the best lower bound and the best upper bound is smaller for the Type B problems than for the Type A problems, suggesting that the Type B problem instances that we considered were 'easier' for the algorithm that we tested. One likely reason for this phenomenon is the fact that Type B problems contain some arcs that are already open while Type A problems do not contain any such arcs. Indeed, the results for problems B3 and B4 suggest that the set of Open arcs constitutes an 'almost' optimal design. For these two problems, the reduction procedure succeeded in identifying the optimal design completely, and only 3 and 2 design arcs, respectively, were included in the corresponding optimal designs.

(2) The number of Part 1-Part 2 cycles required for Type B problems is significantly lower than that for Type A problems. For all Type B problems, termination in each run occured well before the specified maximum of 20 iterations and the cycling phenomenon mentioned earlier was minimal.

(3) Type B problems seem to require lesser time for problem reduction as compared to Type A problems. Perhaps the smaller gaps for Type B problems enabled elimination of a much larger number of flows and arcs in the first iteration of the problem reduction procedure, thus reducing the the number of iterations and the time required before termination.

In summary, the results demonstrate that the dual ascent technique yields fairly tight lower bounds and that the dual-based heuristic is quite effective. They also highlight the influence of the problem structure on algorithmic performance. In terms of computation time, there does seem to be additional scope for improvement in our implementation. For instance,

the time for problem reduction can be reduced significantly by eliminating some of the ineffective tests. Since the dual ascent, heuristic, and problem reduction algorithms rely heavily on repeated shortest path computations, using appropriate data structures (such as a Forward Star representation of the arcs) and an efficient shortest path subroutine is bound to improve the performance. Finally, the contribution of Parts 3 and 4 in terms of improving the dual objective function must be evaluated more systematically, and if necessary the structure of the dual ascent algorithm must be reorganized so that these phases are used less frequently.

4.7 Concluding Remarks

In this chapter, we have studied dual ascent procedures for the Network Design problem. We examined the basic principles underlying such methods and elaborated on several ways in which the basic algorithm can be implemented, both for the uncapacitated and the capacitated problems. Since dual ascent procedures are necessarily heuristic in nature, none of the implementations is provably superior to the others and the appropriate choice might well depend on the structure of the problem under consideration. We discussed several dual-based heuristic procedures before outlining methods for fixing variables using the dual solution. We also indicated the nature of the close relationship between dual ascent and Lagrangian relaxation. The computational tests, though not exhaustive, confirm our initial hypothesis that dual ascent is an attractive technique not only for generating tight lower bounds but also as a basis for good heuristic procedures for the NDP. We discuss several directions for

further work in this area in Chapter 6. In the next chapter, we consider a problem of considerable practical interest, namely, the LTL Consolidation problem, that can be modeled as a special case of the Network Design problem. For this problem, we devise a composite Lagrangian-based procedure that generates both lower and upper bounds. We also report on computational experience using this algorithm.

CHAPTER 5

A LAGRANGIAN-BASED ALGORITHM FOR THE PIECEWISE LINEAR, CONCAVE COST MULTICOMMODITY NETWORK FLOW PROBLEM

In many 'real-world' planning contexts, decision-makers must contend with cost functions that exhibit strong economies of scale. While modeling such problems, a local linear approximation might be acceptable in some instances, especially if the problem under consideration is operational, rather than tactical or strategic, in nature. However, for medium and long-term planning problems, it may be essential to consider the concavity of the cost function in the problem formulation. In this chapter, we consider network problems in which the cost of flow along each arc is a piecewise linear, concave function of the total flow along that arc. Such problems arise in transportation planning, design of computer networks, plant location and capacity expansion planning, production planning/inventory management, and water resource management. This problem can be formulated as a Network Design problem over a suitably defined network; solving it to optimality would necessitate the use of a branch-and-bound or enumeration procedure. Our emphasis, as in the last chapter, is to derive good lower and upper bounds for the problem rather than to solve it optimally. We develop and test a composite Lagrangian-based procedure that exploits the special structure of this problem.

This chapter is organized as follows. We first present a formal definition of the problem and discuss a specific application, namely the

LTL (Less-than-Truckload) Consolidation problem, that motivated our work in this area. We then briefly review the literature on related minimum concave-cost models. As in Chapter 4, we use a Lagrangian relaxation of the integer programming formulation of the problem for three purposes - to generate good lower bounds, as the basis for a heuristic procedure, and for problem reduction. We describe each of these segments in detail before reporting on our computational experience with the algorithm on a series of randomly generated problems.

5.1 Problem Definition and an Application

The Concave-cost Network Flow problem (abbreviated as CCNFP) that we consider in this chapter involves routing multiple commodities drawn from a given set K. For each commodity $k \epsilon K$, R_k units must be transported from the commodity's origin O(k) to its destination D(k) along the arcs of the given network G:(N,A). The commodities can be routed via any number of intermediate 'transshipment' points. The cost of flow along each arc is a piecewise linear, concave function. Therefore, for each arc (i,j) of the network, the set of all possible flows on that arc is partitioned into several prespecified 'cost ranges'. The incremental cost per unit of flow on the arc is constant within each of these ranges, and this per unit cost is lower for higher ranges. For convenience, we will assume that the total number of ranges r^+ is the same for all arcs; our algorithm is applicable, however, to the more general case in which this number varies by arc. We assume that the ranges are indexed from 1 to r^+, with range 1 having a lower limit for flow of 0 units and range r^+ having an upper limit that is

greater than or equal to the maximum possible flow on arc (i,j) (or equal to $\Sigma_k R_k$, say). Let M_{ij}^{r-1} and M_{ij}^{r} denote the lower and upper limits, respectively, of range r for arc (i,j), with the convention that $M_{ij}^{0} = 0$ for all arcs. Let c_{ij}^{r} denote the cost per unit of <u>additional</u> flow in range r on arc (i,j). Costs that are incurred for processing the commodities at transshipment nodes can be incorporated in the model using the familiar node-splitting device. We assume that the flow cost c_{ij}^{r} is non-negative for all arcs and ranges and is the same for all commodities k. Thus,

$$0 = M_{ij}^{0} < M_{ij}^{1} < M_{ij}^{2} < \ldots < M_{ij}^{r^+}, \text{ and}$$

$$c_{ij}^{1} > c_{ij}^{2} > \ldots > c_{ij}^{r^+} \geq 0,$$

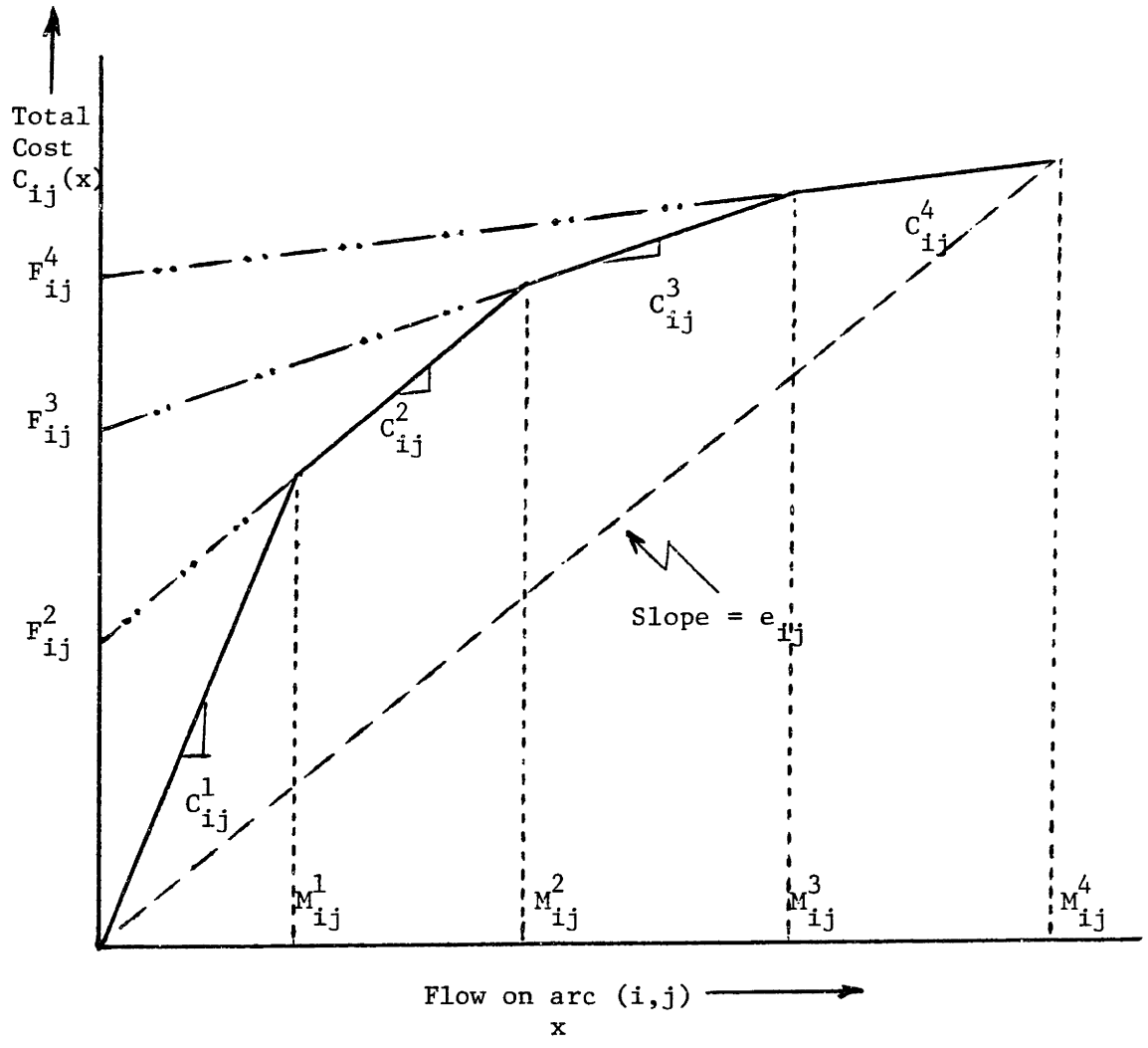
for all arcs (i,j) in the network. For convenience, we will assume that no fixed charge is incurred for using arcs; the algorithm that we develop also solves problems with arc fixed costs. Figure 18 shows a typical total cost curve for flow along arc (i,j). The total cost of routing x units of flow on arc (i,j), denoted as $C_{ij}(x)$, is

$$C_{ij}(x) = \sum_{r=1}^{r'-1} c_{ij}^{r} M_{ij}^{r} + (x - M_{ij}^{r'-1}) c_{ij}^{r'}$$

where $r' \leq r^+$ is the range for which $M_{ij}^{r'-1} \leq x \leq M_{ij}^{r'}$. The objective is to find the flow pattern that satisfies all the demands at minimum total cost.

Transportation planning is a natural context in which this model is widely applicable. Consider, for instance, the problem facing a firm that must transport relatively small quantities of finished goods (or components) from several widely dispersed warehouses (or supplier locations) to its numerous customers (or assembly plants). One distribution strategy is to dispatch the required flows directly from one or more of the warehouses to each of the customers. This strategy is usually not cost effective, however, because it does not exploit the

Figure 18: Total cost of flow on arc (i,j)



economies of scale in transportation costs. Typically, freight carriers offer discounts that increase with the volume shipped. These discount rates are normally available only when the quantities to be shipped are larger than a truckload; however, shipments to individual customers do not usually exceed even half a truckload. Thus, the firm can reduce its distribution costs significantly by 'consolidating' several shipments into truckloads, and dispatching these truckloads to intermediate points close to the customer locations where they are 'broken' into Less-than-Truckload (LTL) shipments that are finally delivered to individual customers. In the transportation literature, the locations at which small shipments are accumulated and dispatched in bulk are called CONSOLIDATION points, and the points where truckloads are 'disassembled' are called BREAKBULK points. For strategic reasons, both consolidation and breakbulk points must usually be located in or around large metropolitan areas. Hence, the potential locations for these intermediate transhipment points are prespecified. The problem, then, is to determine the optimal dispatching policy, i.e. the route along which each shipment (or commodity) must be dispatched in order to exploit the transportation economies of scale to the maximum possible extent. Implicit in this decision is the choice (from the given set of potential locations) of the intermediate transhipment points that must be used and the assignment of all the sources and destinations to the chosen consolidation and breakbulk points, respectively.

This transportation planning problem (or variants of it) has often been referred to in the literature as the LTL Consolidation problem. As is evident from this discussion, it is advantageous to locate consolidation points close to the sources (in this case, warehouses) and breakbulk points

close to the destinations (customer locations, in our example). However, as is often the case, if both the origins and the destinations are widely dispersed, the choice of the intermediate points that must be used and the subsequent routing decisions are not so straightforward. A formal optimization model such as the CCNFP must then be used for aiding the distribution policy decisions. So far, we have described the problem from the point of view of a manufacturer or wholesaler who employs the services of one or more freight operators. However, manufacturers who transport material themselves and freight operators who handle LTL shipments (such as the package delivery services) also face a similar problem. In these cases, the transportation costs for small volumes (LTL shipments) is likely to be lumpy; however, for larger volumes, we can assume the cost function to be continuous because of the various possible combinations of truck and trailer sizes. Therefore, if we have the option of using outside carriers for small shipments, then the piecewise linear, concave cost approximation of the total cost function might be acceptable. As mentioned earlier, operating costs of the consolidation and breakbulk centers can also be incorporated in the model. Finally, note that, in our model, we assume that beyond each 'discount threshold', the lower rates are applicable only to any incremental volume, rather than to the entire shipment; when the entire shipment can be transported at the discounted rate, the total cost function is no longer concave and our model is inappropriate.

Branch-and-bound and Dynamic programming are the two main optimization approaches that have been discussed in the literature for solving the general concave-cost network flow problem. Zangwill [1968] has exploited a characterization of extreme point solutions for the single commodity

problem to develop a dynamic programming algorithm for the single source, multiple destination, concave cost problem defined over an acyclic network. Erickson et al. [1981] have proposed a dynamic programming procedure that generalizes the algorithm originally proposed by Dreyfus and Wagner [1972] for solving the Steiner tree problem. Soland [1974] describes a branch-and-bound algorithm for the Concave-cost Plant location problem; this algorithm is derived as a special case of a method for minimizing separable concave functions over a polyhedral set. Taha [1973] has proposed a branch-and-bound algorithm that employs a special cutting plane procedure for finding the minimum of a concave function over a convex polyhedron (that is not necessarily a network flow polyhedron). Gallo and Sodini [1979] have developed a vertex following algorithm for finding a local optimum for the general concave-cost multicommodity flow problem. They report computational results for problems with up to 48 nodes, 174 arcs and 5 commodities. As mentioned earlier, for the UNDP, this procedure becomes equivalent to the Flow Rerouting local improvement heuristic that we employed in Chapter 4 (and that we again use here). All these papers deal with problems having general concave-cost objective functions; and, with the exception of Soland [1974], all of them assume that the networks are uncapacitated. In addition to this literature, there are several other papers that deal with particular application areas (such as Zadeh [1973],[1974] for communication networks, Florian and Klein [1971], Love [1973], and Swoveland [1975] for production planning, and Fong and Rao [1975], and Luss [1979] for capacity expansion). Finally, Powell and Sheffi [1983] and Lamar [1983] have considered variants of the LTL Consolidation problem that we just described. Both these papers represent the problem as a Fixed-charge network design problem rather than as a

concave-cost minimization problem. Powell and Sheffi [1983] propose a heuristic approach that trades off the level of service against the cost of the load plan. Lamar [1983], on the other hand, solves the LTL problem as an Uncapacitated Network Design problem without any side constraints; his algorithm successively strengthens the weak Arc-flow formulation of the UNDP (using valid 'capacity' inequalities similar to those discussed in Section 4.5) to generate lower bounds at each node of the branch-and-bound tree.

5.2 Integer Programming Formulation of the Concave-cost Network Flow Problem

The piecewise linear, concave-cost network flow problem can be formulated as a mixed-integer program in a variety of ways. One approach is to include in the formulation r^+ binary variables corresponding to each arc (i,j) of the network, with the r^{th} of these variables denoting whether or not the total flow that is routed on arc (i,j) exceeds the upper bound of the r^{th} range. This formulation is sometimes called the Delta method. (See, for example, Bradley et al. [1977].) Alternatively, the Lamda method models the piecewise linear, concave cost function by representing the flow on each arc as the weighted combination of the upper and lower limits of the cost ranges; this formulation requires an adjacency condition, expressed in terms of r^+ binary variables for each arc, to ensure that the cost function is properly represented. We formulate the CCNFP in a slightly different manner. In our formulation, the r^+ cost ranges on each arc are essentially modeled as r^+ parallel arcs, each with a

different 'fixed-charge', flow cost, and capacity. In order to determine the parameters for this equivalent network, let us return to the total flow cost function shown in Figure 18. Let F_{ij}^{r} be the intercept on the y-axis of the linear cost segment corresponding to range r on arc (i,j). Since no fixed-charge is incurred for using arcs of of the original network, $F_{ij}^{1} = 0$; hence,

$$F_{ij}^{r} = C_{ij}(M_{ij}^{r}) - c_{ij}^{r}M_{ij}^{r} \quad \text{for all } 1 \leq r \leq r^{+},$$

where $C_{ij}(x)$ is the total cost of routing x units of flow on arc (i,j). Notice that, because of the concavity of the cost function,

$$0 = F_{ij}^{1} < F_{ij}^{2} < \ldots < F_{ij}^{r^{+}},$$

for all arcs $(i,j) \epsilon A$. Consider now an equivalent network G":(N,A") that contains r^{+} parallel arcs $(i,j,1),\ldots,(i,j,r^{+})$ in place of each arc (i,j) in the original network. Suppose we assign the fixed cost F_{ij}^{r} and the flow cost c_{ij}^{r} to arc (i,j,r) of this new network. Then, the Network Design problem defined over G" is equivalent to the original CCNFP problem. This equivalence is readily established by observing that a flow of x units in the r^{th} range of arc (i,j) in the original network corresponds to a flow of x units on the arc (i,j,r) in G"; furthermore, because of the manner in which the parameters F_{ij}^{r} are defined, the cost incurred in both cases is the same. Consequently, an optimal solution for either problem can be used to determine an optimal solution with the same cost for the other problem. This observation forms the basis for the IP formulation of the CCNFP that we present next. However, in our subsequent discussions, we do not explicitly use the equivalent graph G' just defined.

Define the decision variables $x_{ij}^{k^{r}}$ and y_{ij}^{r}, for all $(i,j) \epsilon A$, $1 \leq r \leq r^{+}$ and $k \epsilon K$, as follows:

$x_{ij}^{k\,r}$ = Number of units of flow of commodity k routed in the r^{th} range on arc (i,j), for all $(i,j)\epsilon A$, $k\epsilon K$, $1 \leq r \leq r^+$, and

y_{ij}^r = $\begin{cases} 1 & \text{if the } r^{th} \text{ range of arc } (i,j) \text{ is used} \\ 0 & \text{otherwise,} \end{cases}$ for all $(i,j)\epsilon A$, $1 \leq r \leq r^+$.

Then, the CCNFP can be represented mathematically as

[CCNFP]

$$\text{Minimize} \sum_{(i,j)} \sum_{r} c_{ij}^r \Big(\sum_{k} x_{ij}^{k\,r} \Big) + \sum_{(i,j)} \sum_{r} F_{ij} y_{ij}^r \qquad (15.1)$$

subject to

$$\sum_{j} \Big(\sum_{r} x_{ij}^{k\,r} \Big) - \sum_{j} \Big(\sum_{r} x_{ji}^{k\,r} \Big) = \begin{cases} +R_k & \text{if } i = O(k) \\ -R_k & \text{if } i = D(k) \\ 0 & \text{otherwise} \end{cases} \quad \text{for } k\epsilon K \qquad (15.2)$$

$$x_{ij}^{k\,r} \leq d_{ij}^{k\,r} y_{ij}^r \qquad \text{for } (i,j), r, k \qquad (15.3)$$

$$\sum_{k} x_{ij}^{k\,r} \geq M_{ij}^{r-1} y_{ij}^r \qquad \text{for } (i,j), r \qquad (15.4)$$

$$\sum_{k} x_{ij}^{k\,r} \leq M_{ij}^r y_{ij}^r \qquad \text{for } (i,j), r \qquad (15.5)$$

$$\sum_{r} y_{ij}^r = 1 \qquad \text{for } (i,j) \qquad (15.6)$$

$$y_{ij}^r = 0 \text{ or } 1, \; x_{ij}^{k\,r} \geq 0 \qquad \text{for } (i,j), r, k, \qquad (15.7)$$

where $d_{ij}^{k\,r} = \text{Min } \{R_k, M_{ij}^r\}$.

Constraints (15.2) are the flow conservation equations, and constraints (15.3) are the forcing constraints. (15.4) and (15.5) are the lower and upper bounds, respectively, for each range on each arc. Constraints (15.6) specify that exactly one range must be chosen for each arc. For simplicity, we will henceforth assume that, for all commodities k, demand R_k is less than or equal to the upper limit M_{ij}^1 of the first cost range on

each arc (i,j); then, the coefficient d_{ij}^{kr} in the forcing constraint (15.3) becomes equal to R_k. Notice that the lower and upper bounds (constraints (15.4) and (15.5)) are redundant in the IP formulation of the CCNFP. However, these constraints are not redundant in the Lagrangian relaxation that we discuss later. Finally, observe that since $F_{ij}^1 = 0$, the first range can be selected without incurring any additional cost whenever no flow is routed on arc (i,j). Hence, there is no loss of generality in specifying that <u>exactly</u>, rather than <u>at most</u>, one range must be selected (in constraints (15.6)).

In our composite algorithm, we use the Lagrangian relaxation obtained by dualizing the flow conservation equations (15.2) to

(1) generate lower bounds on the optimal value of [CCNFP]:
We solve the Lagrangian dual problem approximately by employing both a multiplier adjustment method to monotonically increase the Lagrangian lower bound and a subgradient optimization procedure.

(2) generate upper bounds:
The algorithm uses the current best Lagrangian subproblem solution to identify an intial heuristic solution. With this solution as the starting point, the Flow Rerouting method determines a locally optimal CCNFP solution.

(3) fix the variables in the problem:
We use Lagrangian-based problem reduction techniques similar to those discussed in Section 4.5.

Beasley [1983] has proposed a composite algorithm for the Set covering problem that also incoporates dual ascent, subgradient optimization and problem reduction. Next, we describe each of these three components of the composite algorithm in greater detail.

5.3 The Lagrangian Relaxation scheme

Consider the relaxation of [CCNFP] that is obtained by dualizing the flow conservation equations (15.2) using multipliers v_i^k for all $i \epsilon N$ and $k \epsilon K$. For each commodity k, one of the constraints in (15.2) is redundant; consequently, one of the Lagrange multipliers corresponding to commodity k can be set arbitrarily. We will assume that $v_{D(k)}^k = 0$ for all $k \epsilon K$. Then, the Lagrangian subproblem [SP(v)] corresponding to a given vector of multipliers $v = \{v_i^k\}_{i \epsilon N, k \epsilon K}$ is

[SP(v)]

$$\text{Minimize} \sum_{(i,j)} \sum_{r} \sum_{k} (c_{ij}^r + v_i^k - v_j^k) x_{ij}^{kr} + \sum_{(i,j)} \sum_{r} F_{ij}^r y_{ij}^r + \sum_{k} R_k v_{D(k)}^k$$

subject to (15.3), (15.4), (15.5), (15.6), and (15.7).

We let $z^L(v)$ denote the optimal value of [SP(v)]. As is well-known, $z^L(v)$ is a lower bound on the optimal value z^* of [CCNFP] for any given vector v of Lagrange multipliers. The best lower bound z^{LD} is obtained by solving the following Lagrangian dual problem [LD]:

[LD]

$$z^{LD} = \max_{v} z^L(v).$$

We next discuss a method for solving the subproblem [SP(v)] and for iteratively finding a set of multipliers that yields good, though not necessarily the best, lower bounds.

5.3.1 Solving the Lagrangian subproblem:

The Lagrangian subproblem [SP(v)] separates by arc, and can hence be solved by solving a series of smaller subproblems, one corresponding to

each arc of the network. For any given set of multipliers $\{v_i^k\}$, we let $[SP_{ij}(v)]$, for all $(i,j) \epsilon A$, denote the subproblem corresponding to arc (i,j). This subproblem has the following form:

$[SP_{ij}](v)]$

$$z_{ij}(v) = \text{Minimize} \quad \sum_r \sum_k b_{ij}^{kr} x_{ij}^{kr} + \sum_r F_{ij}^r y_{ij}^r \qquad (16.1)$$

subject to

$$x_{ij}^{kr} \leq R_k y_{ij}^r \qquad \text{for all } r,k \qquad (16.2)$$

$$M_{ij}^{r-1} y_{ij}^r \leq \sum_r x^{k,r} \leq M_{ij}^r y_{ij}^r \qquad \text{for all } r \qquad (16.$$

$$\sum_r y_{ij}^r = 1 \qquad (16.4)$$

$$y_{ij}^r \epsilon \{0,1\}, \; x_{ij}^{kr} \geq 0 \qquad \text{for all } r,k, \qquad (16.5)$$

where $b_{ij}^{kr} = c_{ij}^r + v_i^k - v_j^k$ for all r,k.

The coefficient b_{ij}^{kr} is the modified flow cost for commodity k in the r^{th} range of arc (i,j). Unlike the original flow cost c_{ij}^r, this modified cost varies by commodity and depends on the current values of the Lagrange multipliers v_i^k.

Once the subproblems $[SP_{ij}(v)]$ have been solved for all arcs $(i,j) \epsilon A$, the optimal value of $[SP(v)]$ can be determined as

$$z^L(v) = \sum_{(i,j)} z_{ij}(v) + \sum_k R_k v_{D(k)}^k. \qquad (16.6)$$

The subproblem $[SP_{ij}(v)]$ does not satisfy the so-called 'Integrality' property (see, Geoffrion [1974]), i.e., the LP relaxation of this problem does not necessarily have integer optimal solutions. Therefore, the best

lower bound z^{LD} obtained using this relaxation is at least as good as the optimal value of the LP relaxation of [CCNFP]. Geoffrion and McBride [1978] have observed, in the context of capacitated facility location problems, that this type of relaxation (obtained by dualizing the flow conservation equations) very often yields lower bounds that are significantly better than the LP lower bound.

Let us now examine the procedure for solving $[SP_{ij}(v)]$. In order to satisfy constraint (16.4), exactly one of the variables y^r_{ij}, for $1 \leq r \leq r^+$, must be set to 1. Observe that, for any range r,

(a) if y^r_{ij} is set equal to 0, then, for all commodities $k \in K$, x^{kr}_{ij} must also be 0.

(b) if y^r_{ij} is set equal to 1, then the corresponding 'optimal' values of x^{kr}_{ij} can be determined by solving the following problem:

$[SP^r_{ij}(v)]$

$$z^r_{ij}(v) = \text{Min} \quad \sum_k b^{kr}_{ij} x^{kr}_{ij} + F^r_{ij}$$

subject to

$$x^{kr}_{ij} \leq R_k \qquad \text{for all } k \in K$$

$$M^{r-1}_{ij} \leq \sum_k x^{kr}_{ij} \leq M^r_{ij}$$

$$x^{kr}_{ij} \geq 0 \qquad \text{for all } k \in K.$$

$[SP_{ij}(v)]$ can, therefore, be solved by

- evaluating $z^r_{ij}(v)$ for each range r, $1 \leq r \leq r^+$,
- identifying the range r^* for which $z^r_{ij}(v)$ is the minimum, and
- setting

$$y^r_{ij} = \begin{cases} 1 & \text{if } r = r^* \\ 0 & \text{otherwise, and} \end{cases}$$

$$x_{ij}^{kr} = \begin{cases} 0 \text{ for all } k \in K, & \text{if } r \neq r^* \\ \text{optimal solution of } [SP_{ij}^{r^*}(v)] & \text{if } r = r^*. \end{cases}$$

Solving the subproblem $[SP_{ij}^{r}(v)]$:

The subproblem $[SP_{ij}^{r}(v)]$ is a linear knapsack problem and can be solved efficiently in the following manner.

- Arrange all the commodities in increasing order of b_{ij}^{kr}.

  We assume, without loss of generality, that the commodities are indexed in increasing order of b_{ij}^{kr}, i.e.,

$$b_{ij}^{1r} \leq b_{ij}^{2r} \leq \ldots\ldots \leq b_{ij}^{|K|r}.$$

- Let k0, k1 and k2 be the three commodity indices defined as

$$k0 = \{ \text{Max } k : b_{ij}^{kr} \leq 0 \},$$

$$k1 = \{ \text{Min } k : \sum_{k'=1}^{k} R_{k'} \geq M_{ij}^{r-1} \}, \text{ and}$$

$$k2 = \{ \text{Max } k : \sum_{k'=1}^{k} R_{k'} \leq M_{ij}^{r} \}.$$

- Then, the optimal solution of $[SP_{ij}^{r}(v)]$ can be determined as follows:

  Case 1: $k0 < k1$

$$x_{ij}^{kr} = \begin{cases} R_k & \text{if } k < k1 \\ M_{ij}^{r-1} - \sum_{k'=1}^{k1-1} R_{k'} & \text{if } k = k1 \\ 0 & \text{otherwise.} \end{cases}$$

  Case 2: $k1 \leq k0 \leq k2$

$$x_{ij}^{kr} = \begin{cases} R_k & \text{if } k \leq k0 \\ 0 & \text{otherwise.} \end{cases}$$

Case 3: $k0 > k2$

$$x_{ij}^{k\,r} = \begin{cases} R_k & \text{if } k \leq k2 \\ M_{ij}^r - \sum_{k'=1}^{k2} R_{k'} & \text{if } k = k2+1 \\ 0 & \text{otherwise.} \end{cases}$$

Thus, for each arc (i,j) and range r, once the commodities are arranged in increasing order of $b_{ij}^{k\,r}$, the subproblem $[SP_{ij}^r(v)]$ can be solved in $O(|K|)$ time. Notice also that the commodities must be sorted only once, say for range 1, for each arc; the order of the commodities remains the same for all other ranges. This fact follows from the observation that

$$\begin{aligned} b_{ij}^{k\,r+1} &= c_{ij}^{r+1} + v_i^k - v_j^k \\ &= b_{ij}^{k\,r} + (c_{ij}^{r+1} - c_{ij}^r); \end{aligned}$$

hence,

$$b_{ij}^{k',r+1} \leq b_{ij}^{k\,r+1} \text{ iff } b_{ij}^{k',r} \leq b_{ij}^{k\,r} \text{ for any } k,k' \in K \text{ and } 1 \leq r < r^+.$$

Therefore, the total time required to solve the r^+ subproblems $[SP_{ij}^r(v)]$, for $1 \leq r \leq r^+$, corresponding to arc (i,j) is $O(|K|\log|K|) + O(r^+|K|) = O(|K|(r^+ + \log|K|))$. (The term $O(|K|\log|K|)$ corresponds to the time required to sort the commodities in order of increasing $b_{ij}^{k\,r}$'s.) Therefore, for any given set of Lagrange multipliers $\{v_i^k\}$, we can solve the Lagrangian subproblem [SP(v)] in $O(m|K|(r^+ + \log|K|))$ operations, where m is the total number of arcs in the given network. Leung et al. [1984] solve a routing problem for point-to-point delivery systems using a similar relaxation. Their subproblems reduce to integer knapsack problems because they model the flow cost as a step function. They use an ascent procedure, similar to the multiplier adjustment method that we describe later, to modify the Lagrangian multipliers and obtain good lower bounds.

Notice that, with minor modifications, this subproblem solution procedure can handle, for one or more arcs (i,j) and commodities k, additional constraints of the form

$$\Sigma_r \, x_{ij}^{kr} = R_k \qquad \text{or} \tag{16.7a}$$

$$\Sigma_r \, x_{ij}^{kr} = 0, \tag{16.7b}$$

in the subproblem [SP(v)]. (We later exploit this fact for problem reduction.) Constraint (16.7a) specifies that R_k units of commodity k must be routed on arc (i,j), while (16.7b) prohibits commodity k from flowing on arc (i,j). For any given arc (i,j), let K_a and K_b, respectively, be the subsets of commodities for which constraints (16.7a) and (16.7b) are added to the Lagrangian subproblem [SP(v)], and let R^a be the total demand for all commodities that belong to the set K_a. Then, while solving the 'augmented' subproblem $[SP_{ij}(v)]$ corresponding to arc (i,j), we must eliminate all commodities belonging to the set K_b from the sorted list of commodities. Also, we must solve the knapsack problem $[SP_{ij}^r(v)]$ only for those ranges r for which $M_{ij}^r \geq R^a$; by placing all the commodities belonging to the set K_a at the beginning of the 'sorted' list of commodities, we can ensure that the subproblem solutions satisfy constraints (16.7a).

So far, we have seen that for any given set of multipliers, the Lagrangian subproblems can be solved very efficiently. Before discussing methods for iteratively modifying the multipliers to increase the Lagrangian lower bound, we further explore the nature of the subproblem objective function. For this purpose, it is useful to plot the Lagrangian subproblem objective function value $z_{ij}(v)$ as a function of the total flow on arc (i,j). Suppose we add to the subproblem $[SP_{ij}(v)]$ the constraint

$$\sum_{k} \sum_{r} x_{ij}^{kr} = q, \qquad (16.8)$$

where q is the required <u>total</u> flow on arc (i,j), for $0 \leq q \leq \epsilon_k R_k$. Let $z_{ij}(v,q)$ denote the optimal value of $[SP_{ij}(v)]$ with the additional constraint (16.8). Notice that the value of x uniquely determines the range in which arc (i,j) must operate. Let r' be this range, i.e., $M_{ij}^{r'-1} \leq q \leq M_{ij}^{r'}$. Then, we can evaluate $z_{ij}(v,q)$ in the following manner.

- Index the commodities (from 1 to $|K|$) in order of increasing $b_{ij}^{kr'}$ values.
- Let k' be the smallest commodity index for which

$$S_{k'} = \sum_{k=1}^{k'} R_k \geq q.$$

- Then,

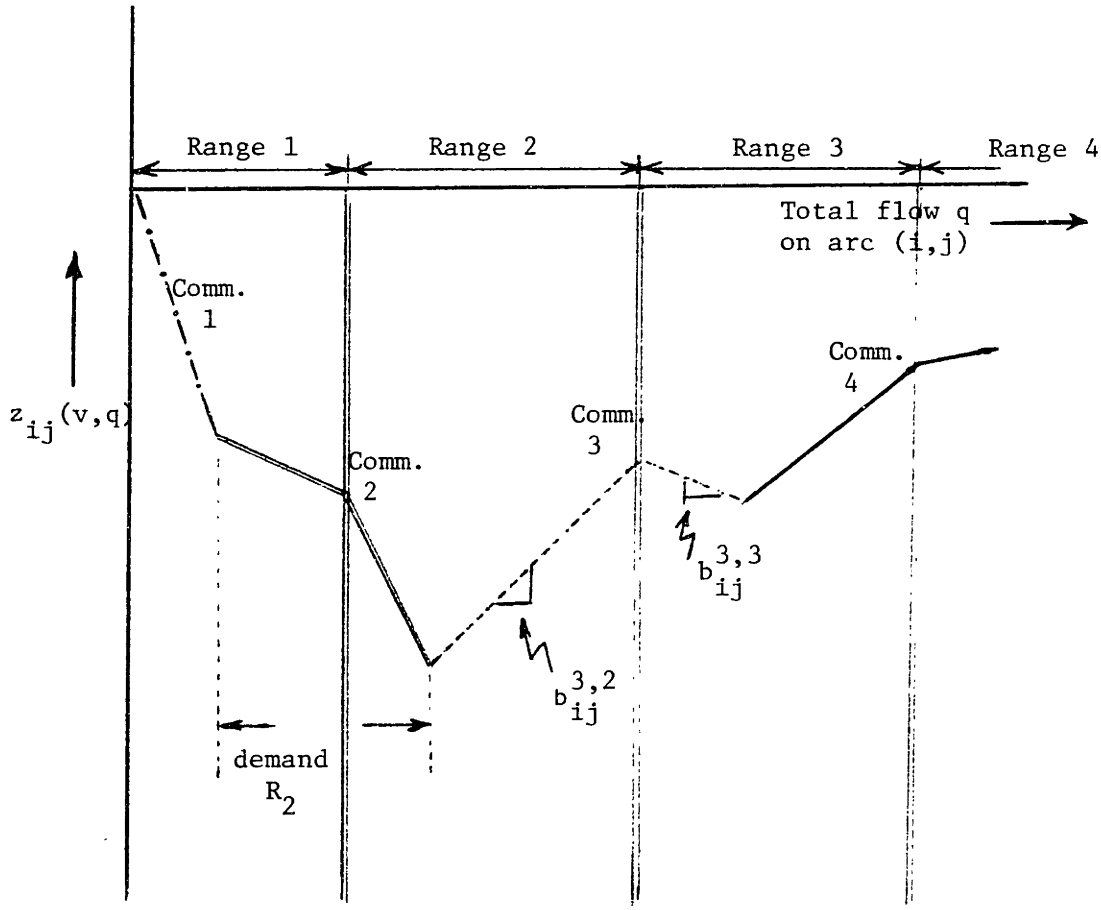
$$z_{ij}(v,q) = \sum_{k=1}^{k'-1} b_{ij}^{kr'} R_k + b_{ij}^{k',r'}(q - S_{k'}) + F_{ij}^{r'}. \qquad (16.9)$$

It is possible to show that, because of the manner in which the parameters F_{ij}^{r} are defined, $z_{ij}(v,q)$ is a continuous function of q. Figure 19 shows the graph of $z_{ij}(v,q)$ as a function of q, for some given set of Lagrange multipliers $v = \{v_i^k\}$. In this example, commodities 1 and 2 have negative modified costs $b_{ij}^{1\,1}$ and $b_{ij}^{2\,1}$, respectively, in range 1. The function is piecewise linear and convex <u>within each cost range</u> because the commodities are arranged in order of increasing b_{ij}^{kr}. However, since the modified cost b_{ij}^{kr}, for any commodity k, decreases from one cost range to the next, $z_{ij}(v,q)$ is not convex across all cost ranges.

We can now express the optimal value $z_{ij}(v)$ of the original subproblem $[SP_{ij}(v)]$ in terms of the parametrized subproblem objective function value $z_{ij}(v,q)$ as

$$z_{ij}(v) = \min_{0 \leq q \leq \sum_k R_k} z_{ij}(v,q)$$

Figure 19: z_{ij} as a function of the total flow on arc (i,j)



In the example of Figure 19, the optimal solution is

$$y_{ij}^{2} = 1 , \quad y_{ij}^{r} = 0 \quad \text{for all } r \neq 2$$

$$x_{ij}^{k\,r} = 0 \quad \text{for all } r \neq 2$$

$$x_{ij}^{1\,2} = R_1, \quad x_{ij}^{2\,2} = R_2, \quad x_{ij}^{k\,2} = 0 \quad \text{for all } k > 2,$$

and the 'operating' range for this solution is range 2. It is easy to see that, in general, the Lagrangian subproblem $[SP_{ij}(v)]$ has an optimal solution with

$$x_{ij}^{k\,r} = 0 \text{ or } R_k \quad \text{for all } (i,j),\ k \text{ and } r,$$

for any given set of Lagrange multipliers $\{v_i^k\}$. Indeed, our algorithm finds such a solution. Later, we will use Figure 19 to determine the effects on the optimal value $z_{ij}(v)$ of changing the modified costs $b_{ij}^{k\,r}$ (by changing the Lagrange multipliers).

Our final objective is to identify a set of 'good' multipliers for which the corresponding subproblem objective function value $z^L(v)$ is a tight lower bound on the optimal value of [CCNFP]. For this purpose, we will incorporate the procedure for solving the Lagrangian subproblems in an iterative scheme that modifies the Lagrange multipliers at each step. Before describing the techniques used to modify the multipliers, we discuss the procedure used for initializing these multipliers.

5.3.2 Initializing the Lagrange multipliers:

The multipliers $\{v_i^k\}$ are similar to the dual variables corresponding to the flow conservation equations in a shortest path problem. This interpretation suggests that v_i^k must be set equal to the length of the shortest path from origin O(k) to node i using a modified arc length that

depends on the flow costs c_{ij}^r. We next present one such initialization method.

For each arc $(i,j) \epsilon A$, define the modified length e_{ij} as

$$e_{ij} = c_{ij}^{r+} + (F_{ij}^{r+}/M_{ij}^{r+}).$$

The parameter e_{ij} is the slope of the line joining the origin to the point $(M_{ij}^{r+}, C_{ij}(M_{ij}^{r+}))$ in the graph of the total cost function shown in Figure 18. Thus, for all possible flow values x, $e_{ij}x$ underestimates the total cost $C_{ij}(x)$ of routing x units of flow on arc (i,j). (Recall that we have assumed M_{ij}^{r+}, the upper limit of the last cost range, to be greater than or equal to the maximum possible flow through arc (i,j)). Our first initialization method, therefore, sets

v_i^k <-- length of shortest path from O(k) to node i using e_{ij} as arc lengths, for all $i \epsilon N$, $k \epsilon K$.

As an aside, we note that a similar technique of underestimating the total cost function to derive lower bounds has been used earlier in the plant location context. (See, for example, Efroymson and Ray [1966].)

Claim:

With this choice of multipliers v_i^k, the solution

$$y_{ij}^1 = 1; \quad y_{ij}^r = 0 \quad \text{for all } 1 < r \leq r^+, \text{ and}$$

$$x_{ij}^{k\,r} = 0 \quad \text{for all commodities } k \epsilon K \text{ and ranges } 1 \leq r \leq r^+,$$

is optimal for the Lagrangian subproblem [SP(v)]. Consequently, the initial lower bound using this method is $\Sigma_k R_k v_{O(k)}^k$.

Proof:

Since v_i^k is the length of the shortest path from O(k) to node i using

e_{ij} as arc lengths,

$$v_j^k - v_i^k \leq e_{ij} = c_{ij}^{r+} + f_{ij}^+$$

where $f_{ij}^+ = F_{ij}^{r+}/M_{ij}^{r+}$.

Thus,

$$b_{ij}^{kr} = c_{ij}^r + v_i^k - v_j^k$$

$$\geq c_{ij}^r - c_{ij}^{r+} - f_{ij}^+ \qquad \text{(a)}$$

Now, since the total cost function $C_{ij}(x)$ is piecewise linear and concave, for all ranges $r = 1,2,\ldots,r^+$,

$$c_{ij}^r + F_{ij}^r/M_{ij}^r \geq c_{ij}^{r+} + f_{ij}^+. \qquad \text{(b)}$$

From (a) and (b), we obtain

$$b_{ij}^{kr} \geq -F_{ij}^r/M_{ij}^r \quad \text{for all } (i,j),r,k. \qquad \text{(c)}$$

For any arc (i,j) and any range r, let $\{x_{ij}^{kr}\}_k$ be the optimal solution for the subproblem $[SP_{ij}^r(v)]$. Then,

$$\begin{aligned} z_{ij}^r(v) &= \sum_k b_{ij}^{kr} x_{ij}^{kr} + F_{ij}^r \\ &\geq \sum_k (-F_{ij}^r/M_{ij}^r) x_{ij}^{kr} + F_{ij}^r \qquad \text{from (c)} \\ &\geq (-F_{ij}^r/M_{ij}^r) M_{ij}^r + F_{ij}^r \qquad \text{since } \sum_k x_{ij}^{kr} \leq M_{ij}^r \text{ in } \{SP_{ij}^r(v)\} \\ &= 0. \end{aligned}$$

Hence, for all arcs $(i,j) \in A$, it is optimal to set

$$y_{ij}^1 = 1; \quad y_{ij}^r = 0 \quad \text{for all } r > 1, \text{ and}$$

$$x_{ij}^{kr} = 0 \quad \text{for all } k \in K, \ 1 \leq r \leq r^+.$$

And, for this solution

$$z_{ij}^1(v) = 0 \leq z_{ij}^r(v) \qquad \text{for all } 1 \leq r \leq r^+,$$

implying that

$$z_{ij}(v) = \text{Min}_r \, z_{ij}^r(v) = z_{ij}^1(v) = 0.$$

Therefore, by equation (16.6), $z^L(v) = \Sigma_k R_k$.

The lower bound obtained using this method can be improved by reducing the upper limit of flow M^{r+}_{ij} on each arc (i,j) as far as possible. For instance, if we can determine, using one of the problem reduction techniques described later, that commodity k must not flow on arc (i,j) in the optimal CCNFP solution, then M^{r+}_{ij} can be reduced by R_k (from its initial level of $\Sigma_k R_k$). Therefore, after the problem reduction phase, it might be worthwhile to reinitialize the multipliers using this method (with the updated values of M^{r+}_{ij}) and to compute the new 'tighter' lower bound; the current best lower bound can then be updated if necessary.

An alternative initialization scheme is to set v^k_i, for all nodes i and commodities $k \epsilon K$, equal to the length of the shortest path from O(k) to node i using c^{r+}_{ij} as arc lengths. ($c^{r+}_{ij}x$ also underestimates the total cost $C_{ij}(x)$.) However, since $e_{ij} > c^{r+}_{ij}$, the values assigned to $v^k_{O(k)}$ in this method are smaller than before. Again, in this case, we can show that, in the optimal Lagrangian subproblem solution, $y^1_{ij} = 1$ for all arcs $(i,j) \epsilon A$ and all other variables are equal to 0. Hence, this method yields a smaller initial lower bound than the first method. In our implementation of the algorithm, we used the parameters e_{ij} as arc lengths to initialize the Lagrange multipliers.

Next, we discuss two techniques - a multiplier adjustment method and a subgradient procedure - for modifying the Lagrange multipliers iteratively in order to obtain tight lower bounds.

5.3.3 Obtaining good lower bounds : The Multiplier Adjustment method:

Recall that, in order to obtain the best Lagrangian lower bound, we must solve the Lagrangian dual problem [LD] optimally. Instead of finding the optimal solution to this problem, we will determine a near-optimal set of multipliers using two methods that are applied in tandem - a multiplier adjustment method and a subgradient procedure. We describe the first of these two methods in this section.

The Multiplier Adjustment (or Dual Ascent) method modifies the Lagrangian multipliers iteratively in a heuristic, but efficient, manner so that the Lagrangian subproblem objective function value $z^L(v)$ increases monotonically. (As we discussed in Chapter 4, the dual ascent procedure can also be interpreted as a method for adjusting the multipliers in a suitably defined Lagrangian relaxation scheme. In this chapter, we use the term multiplier adjustment rather than dual ascent in order to distinguish the algorithm described next from the dual ascent procedure for the NDP that we developed in Chapter 4.) We have already cited, in Chapter 4, various problems for which this technique has proved successful.

Our ascent strategy will be to change, if possible, the Lagrange multipliers v_i^k corresponding to each commodity k in a manner that

(1) increases $v^k_{D(k)}$, and

(2) leaves $z_{ij}(v)$ (the optimal value of the subproblem $SP_{ij}(v)$) unchanged for all arcs $(i,j) \in A$.

Since $z^L(v) = \Sigma_{i,j}\, z_{ij}(v) + \Sigma_k\, R_k v^k_{D(k)}$, this modification will increase the Lagrangian objective function value $z^L(v)$ at every step. (The second

condition helps us to determine the ascent directions relatively easily. When $z_{ij}(v)$ is also allowed to increase, the procedure becomes much more involved and computationally expensive.) In order to develop an ascent method based on this principle, let us first explore the effects of changing the multiplier v_i^k corresponding to some commodity k and any intermediate node i (i.e., a node that is neither the origin nor the destination for commodity k). In all of our subsequent discussions, we will denote the total flow of commodity k on arc (i,j) as x_{ij}^k, i.e. $x_{ij}^k = \Sigma_r x_{ij}^{kr}$.

Changing the Lagrange multipliers v_i^k affects the optimal Lagrangian supbroblem solution and value in the following manner.

(1) When v_i^k changes relative to v_j^k, the modified cost $b_{ij}^{kr} = c_{ij}^r - (v_j^k - v_i^k)$ also changes, for all ranges r.

(2) Consequently, commodity k's ranking in the sorted commodity list (used for solving the knapsack subproblem $[SP_{ij}(v)]$) changes relative to the other commodities. Therefore, the Lagrangian subproblem solution may change, i.e., commodity k may either be dropped from the knapsack solution if it is currently included in the knapsack or vice versa.

(3) If commodity k is included in the current knapsack solution, the optimal value $z_{ij}(v)$ changes with b_{ij}^{kr}.

Let us now examine in more detail, using the example of Figure 19, the impact on $z_{ij}(v)$ and the Lagrangian solution of increasing and decreasing b_{ij}^{kr} for the two cases - when commodity k belongs to the current knapsack solution and when $x_{ij}^k = 0$ in the current solution. We begin by introducing a definition.

Definition 5.1:

> The OPERATING RANGE of any arc (i,j) is the range r for which $y^r_{ij} = 1$ in the current Lagrangian subproblem solution. We let r(i,j) denote the operating range of arc (i,j).

In Figure 19, the current operating range r(i,j) is 2; commodities 1 and 2 belong to the current knapsack solution, while $x^3_{ij} = x^4_{ij} = 0$.

Suppose $b^k_{ij}{}^r$ decreases for either commodity 1 or 2. Then, the optimal Lagrangian solution remains unchanged, while the value of $z_{ij}(v)$ decreases. On the other hand, as $b^1_{ij}{}^r$ increases, first commodity 1's ranking changes with respect to commodity 2 (when $b^1_{ij}{}^r$ becomes greater than $b^2_{ij}{}^r$). Subsequently, commodity 1 is dropped from the optimal knapsack solution; $z_{ij}(v)$ continues to increase until this point. Similarly, when $b^2_{ij}{}^r$ increases, $z_{ij}(v)$ increases until commodity 2 drops from the knapsack. Let us now consider some commodity, say commodity 4, with zero flow in the current subproblem solution. When $b^4_{ij}{}^r$ increases, the subproblem solution and value remain unchanged. However, when $b^4_{ij}{}^r$ decreases, commodity 4 moves up in the sorted list; and, for values of $b^4_{ij}{}^r$ below a certain threshold value, it becomes optimal to increase commodity 4's flow from its current value of zero to its demand R_4. Until this point $z_{ij}(v)$ remains unchanged; subsequently, it decreases. From Figure 19 it is clear that this threshold must occur before $b^4_{ij}{}^2$ (the modified cost for commodity 4 in the current operating range 2) becomes equal to 0. In fact, when $b^4_{ij}{}^2$ is equal to zero, other commodities (such as commodity 3 in the figure) with zero flows in the current solution might also be added to the knapsack solution; thus, with this value for $b^4_{ij}{}^2$, the new objective function value $z_{ij}(v)$ might be lower than its current value. Thus, changing the modified

cost $b_{ij}^{k}{}^{r}$ may not only change x_{ij}^{k}, but may also affect the flow $x_{ij}^{k'}$ of other commodities k' besides changing the range of operation. We next summarize this discussion in more general terms.

Suppose v_i^k increases by some amount $u > 0$. Then,

Case 1: For all arcs (i,j) incident from node i,

$(v_i^k - v_j^k)$ increases by u. Hence, $b_{ij}^{k}{}^{r} = c_{ij}^{r} - v_j^k + v_i^k$ increases by for all ranges $1 \leq r \leq r^+$. Consequently,

(a) if $x_{ij}^k > 0$ in the current solution,

for small u, $z_{ij}(v)$ increases, and when u exceeds a certain threshold value, it becomes optimal to set $x_{ij}^k = 0$.

(b) if $x_{ij}^k = 0$ in the current solution,

$z_{ij}(v)$ does not change, and $x_{ij}^k = 0$ is optimal for all $u > 0$.

Case 2: For all arcs (j,i) incident to node i,

$(v_j^k - v_i^k)$ and hence $b_{ji}^{k}{}^{r}$ decreases by u, for all ranges $1 \leq r \leq r^+$. Therefore,

(a) if $x_{ji}^k > 0$ in the current solution,

$z_{ji}(v)$ decreases and it is optimal to set $x_{ji}^k = R_k$ for all $u > 0$.

(b) if $x_{ji}^k = 0$ in the current solution,

for small u, $z_{ji}(v)$ does not change and $x_{ji}^k = 0$ remains optimal, and when u is increased beyond a certain value, it becomes optimal to set $x_{ji}^k = R_k$, and $z_{ji}(v)$ starts decreasing.

There is a similar (but reverse effect) on the optimal subproblem solution when v_i^k decreases from its current value. Notice that

Fact 1: in all cases only the net change in $(v_i^k - v_j^k)$ determines the change in $z_{ij}(v)$ and the optimal solution of $[SP_{ij}(v)]$,

Fact 2: for any commodity k with $x_{ij}^k > 0$ in the current solution (cases 1(a) and 2(a)), $z_{ij}(v)$ changes whenever $(v_i^k - v_j^k)$ changes, and

Fact 3: for any commodity with $x_{ij}^k = 0$ in the current solution, when $b_{ij}^{k\,r(i,j)}$ decreases to zero (where r(i,j) is the current operating range for arc (i,j)) commodity k must belong to the new knapsack solution and the new value of $z_{ij}(v)$ may be lower than its current value.

Since, in our ascent method, we do not want to change $z_{ij}(v)$ for any arc (i,j), by Fact 2, we must ensure that $(v_i^k - v_j^k)$ does not change whenever x_{ij}^k is positive in the current Lagrangian solution. We will, therefore, focus on arcs (i,j) and commodities k for which x_{ij}^k is currently zero. For such arcs and commodities, we must determine the maximum decrease in $(v_i^k - v_j^k)$ that will leave $z_{ij}(v)$ unchanged. We will then increase v_j^k relative to v_i^k for a sequence of arcs (i,j) along one or more subpaths to the destination D(k), thus increasing $v_{D(k)}^k$ and (hence $z^L(v)$).

Consider an arc (i,j) and commodity k for which x_{ij}^k is zero in the current solution. In this case, $b_{ij}^{k\,r(i,j)}$ must be positive, i.e., $(v_i^k - v_j^k)$ must be greater than $-c_{ij}^{r(i,j)}$, where r(i,j) is the current operating range for arc (i,j). (As Figure 19 demonstrates, commodity k must otherwise be included in the knapsack solution.) From Fact 3 we know that when b_{ij}^k,r(i,j)] decreases to zero, i.e., when the difference $(v_i^k - v_j^k)$ reduces to $- c_{ij}^{r(i,j)}$, it becomes optimal to set $x_{ij}^k = R_k$; furthermore, the optimal value $z_{ij}(v)$ of $[SP_{ij}(v)]$ may decrease because of this change. Therefore, $-c_{ij}^{r(i,j)}$ is a lower limit on the amount to which $(v_i^k - v_j^k)$ can decrease without reducing $z_{ij}(v)$. In order to determine the exact value of the maximum permissible decrease in $(v_i^k - v_j^k)$ that leaves $z_{ij}(v)$ unchanged,

we would have to resort to an iterative procedure that solves the subproblem $[SP_{ij}(v)]$ repeatedly for different values of $(v_i^k - v_j^k)$ (since changing $b_i^{k,r}$ for one commodity, affects the flow values $x_{ij}^{k'}$ for other commodities as well). Instead of finding this limit exactly, we use the following procedure to underestimate the maximum permissible decrease in $(v_i^k - v_j^k)$. We let w_{ij}^k be the estimate of the smallest permissible value for this difference. The procedure first sets $w_{ij}^k = -c_{ij}^{r(i,j)}$ (i.e., it <u>decreases</u> $(v_i^k - v_j^k)$ to the lower limit that we just identified) and recalculates the new subproblem objective function value. If this new value is lower than the current value of $z_{ij}(v)$, the procedure <u>increases</u> w_{ij}^k by a 'conservative' factor which depends on the new optimal value. It is possible to show that, when the difference $(v_i^k - v_j^k)$ is equal to this 'corrected' value of w_{ij}^k, x_{ij}^k must be zero in the optimal subproblem solution and $z_{ij}(v)$ must remain at its current value.

<u>Step 1</u>:

Let $w_{ij}^k \leftarrow\!- -c_{ij}^{r(i,j)}$.

Using $$b_{ij}^{k',r} = \begin{cases} c_{ij}^r + v_i^{k'} - v_j^{k'} & \text{if } k' \neq k \\ c_{ij}^r + w_{ij}^k & \text{if } k' = k, \end{cases}$$

solve the revised Lagrangian subproblem $[SP_{ij}(v)]$ corresponding to arc (i,j). Let z_{ij}^* be the optimal value of this new subproblem. (As mentioned earlier, $z_{ij}^* \leq z_{ij}(v)$ always, and z_{ij}^* might even be strictly less than $z_{ij}(v)$).

<u>Step 2</u>:

Update $w_{ij}^k \leftarrow\!- w_{ij}^k + (z_{ij}(v) - z_{ij}^*)/R_k$.

Since Step 2 may increase w^k_{ij} by more than is necessary (to ensure that $z_{ij}(v)$ remains unchanged), the final value of w^k_{ij} may be greater than the current value of the difference $(v^k_i - v^k_j)$. Therefore, our final estimate w^k_{ij} of the smallest permissible difference $(v^k_i - v^k_j)$ is obtained by using the equation

$$w^k_{ij} = \text{Min } \{ v^k_i - v^k_j, w^k_{ij} \}.$$

Thus,

$$u^k_{ij} = \text{Max } \{ 0, (v^k_i - v^k_j - w^k_{ij}) \}$$

represents the amount by which $(v^k_i - v^k_j)$ can decrease without changing $z_{ij}(v)$. This value of u^k_{ij} underestimates the maximum permissible decrease in $(v^k_i - v^k_j)$.

Recall that we plan to decrease this difference $(v^k_i - v^k_j)$ by increasing v^k_j relative to v^k_i. Therefore, after calculating these estimates u^k_{ij} for all arcs (i,j) with $x^k_{ij} = 0$ in the current solution, we can determine the amount by which each multiplier v^k_j corresponding to commodity k must be increased by solving the following shortest path problem. We define u^k_{ij} to be zero for all arcs (i,j) with $x^k_{ij} > 0$ in the current Lagrangian solution, and let p^k_i be the length of the shortest path from O(k) to node i using u^k_{ij} as arc lengths. Then, when the dual multipliers v^k_i are updated as

$$v^k_i \text{ <-- } v^k_i + p^k_i \quad \text{for all nodes } i \epsilon N,$$

the lower bound increases by exactly $p^k_{D(k)}$. Using the shortest path lengths p^k_i to update all the multipliers v^k_i corresponding to commodity k ensures that

(1) for all arcs $(i,j) \epsilon A$, the difference $(v^k_i - v^k_j)$ decreases by at most u^k_{ij}, and

(2) $v^k_{D(k)}$ increases by the maximum possible extent subject to the limitations imposed by the allowable decreases u^k_{ij}.

Notice that if there is a path from O(k) to D(k), all of whose arcs have $x^k_{ij} > 0$ in the current solution (i.e., if there is a feasible flow embedded in the current Lagrangian solution), then $v^k_{D(k)}$ does not increase. The algorithm thus exploits the infeasibility of the current solution to increase the Lagrangian lower bound. However, there is no guarantee that, after the multipliers are adjusted in this manner, the new optimal Lagrangian solution will contain a feasible flow.

This procedure can be repeated iteratively, for all commodities, until the Lagrangian lower bound z^L cannot be increased further. However, the method may not necessarily find the optimal set of Lagrange multipliers. As was the case for the dual ascent procedures of Chapter 4, several alternative heuristic rules for scanning the commodity list are possible. For instance, if the commodities are arranged in order of decreasing demand R_k, then the initial iterations of the multiplier adjustment method would yield the largest increases in the Lagrangian lower bound $z^L(v)$. In our implementation of the algorithm, we did not specify any particular order for considering the commodities.

Finally, note that this ascent procedure can be applied before and/or after any other multiplier initialization or adjustment procedure (such as subgradient optimization). Our implementation of the composite algorithm for the CCNFP, uses this procedure both before and after the subgradient procedure that is discussed next.

5.3.4 Obtaining good lower bounds: Subgradient optimization:

Subgradient optimization has been a popular method for finding near-optimal solutions to the Lagrangian dual problem. Many of the applications and improvements of this method were motivated by the successful use of this technique by Held and Karp ([1970],[1971]) to find lower bounds for the Traveling Salesman problem. Camerini et al. [1975], Crowder [1976], and others have suggested improvements in the method. Held et al. [1974] discuss convergence properties of the method.

In its basic form, the subgradient method iteratively updates the current vector of Lagrange multipliers $v = \{v_i^k\}$ according to the rule

$v_i^k \leftarrow v_i^k + \epsilon d_i^k$, (16.10) where d_i^k is a subgradient of the Lagrangian subproblem objective function at the current iteration, and ϵ is a suitably chosen step size. If $\{x_{ij}^{kr}\}$ is the current Lagrangian solution, then, for all $i \epsilon N$ and $k \epsilon K$,

$$d_i^k = \begin{cases} 0 & \text{if } i = D(k) \\ \sum_j \sum_r x_{ij}^{kr} - \sum_j \sum_r x_{ij}^{kr} & \text{if } i \neq O(k), D(k) \\ R_k - \sum_j \sum_r x_{ij}^{kr} & \text{if } i = D(k) \end{cases} \qquad (16.11)$$

is a subgradient of $z^L(v)$ at the current value v of the Lagrange multipliers. Let ϵ^t be the step size used in the t^{th} iteration of the procedure. Then, a sufficient condition (refer Held et al. [1974]) for $z^L(v^t)$ to converge to the optimal value z^{LD} is that

$$\epsilon^t \dashrightarrow 0 \quad \text{and} \quad \sum_{i=1}^{t} \epsilon^i \dashrightarrow \infty \ . \qquad (16.12)$$

The step size that is commonly used is

$$\epsilon^t = \lambda^t(z^+ - z^L(v^t))/||d^t||^2$$

where

z^+ is an upper bound on the optimal value of [CCNFP],

v^t, d^t, and ϵ^t are respectively the dual multipliers, the subgradient, and the step size at the t^{th} iteration, and

λ^t is a step size multiplier between 0 and 2.

In our implementation of the subgradient procedure for the CCNFP,

(1) the user must specify the following parameters while initiating the subgradient procedure:

λ^0 = the Initial value of the step size multiplier, $0 \lambda^0 < 2$,

t_{max} = the Maximum number of subgradient iterations to be executed,

t_n = the number of consecutive subgradient iterations with no increase in $z^L(v)$, after which the step size multiplier is halved,

z_0^+ = the Initial upper bound on the optimal value of [CCNFP], and

r = the 'Discounting factor' for computing the weighted subgradient (discussed next).

(2) At every iteration, a WEIGHTED SUBGRADIENT $\{d'^k_i\}$ is used to update the Lagrange multipliers (instead of the 'unweighted' subgradient discussed earlier). The weighted subgradient is determined as follows.

Let $\{d''^k_i\}$ be the weighted subgradient that was used for updating the multipliers $\{v^k_i\}$ in the previous iteration, and let $\{d^k_i\}$ be the subgradient derived (using equations (16.11)) from the Lagrangian subproblem solution in the current iteration. Then, the weighted subgradient $\{d'^k_i\}$ (to be used for modifying the Lagrange

multipliers next) for the current iteration is computed as

$$d'^{k}_{i} \leftarrow\!- \; d^{k}_{i} + r.d''^{k}_{i} \quad \text{for all } i \epsilon N,\ k \epsilon K,$$

where r is the user supplied discounting factor.

The new values of the Lagrange multipliers are updated using equation (16.10) with d'^{k}_{i} instead of d^{k}_{i}. Notice that, when r = 0, this method becomes identical to the unweighted procedure discussed earlier. Crowder [1976] has observed that using a weighted subgradient with a discounting factor of $0 < r < 1$ improves computational performance of the subgradient procedure.

(3) A step size updating rule that has worked well in practice, but that may not satisfy the convergence condition (16.12), is to halve the step size multiplier periodically (Fisher [1981]). We use a similar rule in our implementation. The user must specify an intial valuλ^{0} of the step size multiplier that is between 0 and 2. Subsequently, the current step size multiplier is halved, whenever, for t_{h} consecutive iterations, the Lagrangian objective function value $z^{L}(v)$ does not improve. (Note: $z^{L}(v)$ need not necessarily increase at each iteration of the subgradient procedure; the algorithm only ensures that the values of $\{v^{k}_{i}\}$ become closer to the optimal value at each iteration).

(4) The user must specify the initial value of the upper bound z^{+}_{0}; in our implementation, we obtain this value using a 'stand-alone' heuristic algorithm. However, we periodically apply a Lagrangian-based heuristic procedure at intermediate stages of the subgradient algorithm in order to identify better heuristic solutions. If this

procedure finds a better solution, the upper bound z^+ is updated and, in all subsequent iterations, the step size is computed using the new upper bound. The user can control the 'periodicity' of this heuristic procedure (i.e., the number of subgradient iterations between two successive applications of the heuristic algorithm). Later, we briefly discuss the structure of the heuristic procedures used for the CCNFP.

(5) Stopping criteria:

The subgradient procedure terminates either when t_{max} (specified by the user) subgradient iterations have been executed, or if the step size ϵ is less than a fixed TOLERANCE level t.

Because of the limit imposed on the maximum number of subgradient iterations and the rule used for changing the step size multiplier, this procedure is not guaranteed to give the optimal set of Lagrange multipliers. However, many empirical studies have shown that similar schemes frequently generate near-optimal solutions to the Lagrangian dual problem. (For example, see Held et al. [1974], and Crowder [1976]).

So far, we have discussed two components - initializing the Lagrange multipliers and modifying these multipliers in order to improve the lower bound - of the composite algorithm for finding upper and lower bounds on the optimal value of [CCNFP]. The algorithm contains two other components, namely a problem reduction phase and a heuristic procedure. We next briefly discuss these two methods. These methods are very similar to those devised for the Network Design problem in Chapter 4. We outline them

mainly for completeness.

5.4 Problem Reduction Methods

As before, the problem reduction procedure attempts to determine, using the Lagrangian solution and information about the topology of the network, for all arcs $(i,j) \epsilon A$, if commodity $k \epsilon K$ MUST or MUST NOT flow on arc (i,j) in any optimal solution of [CCNFP]. For each commodity k, the sets Open(k), Closed(k), and Free(k) denote the sets of arcs on which commodity k MUST, MUST NOT and MAY flow, respectively. Initially, the first two of these sets are empty, while the last contains all the arcs of the given network. These sets are updated whenever additional x_{ij}^k variables are fixed. Let

z^+ = the current best upper bound, i.e., the cost of the current incumbent,

$\{v_i^k\}$ = the multiplier values with the highest Lagrangian objective function value $z^L(v)$ so far, and

$\{x_{ij}^{k\,r}\}$ = the optimal Lagrangian solution corresponding to the set of multipliers $\{v_i^k\}$.

As in Chapter 4, the problem reduction tests are based on the increase in the lower bound when certain constraints are appended to the Lagrangian subproblem. In particular, to test if commodity k MUST (or MUST NOT) necessarily flow on arc (i,j) in the optimal CCNFP solution, we determine the new Lagrangian lower bound z^* (using the current best multipliers) when constraints of the form

$$x^k_{ij} = 0 \quad (\text{or } x^k_{ij} = R_k)$$

are added to the problem formulation. If z^* exceeds the current best upper bound z^+, then commodity k must (or must not) flow on arc (i,j). As mentioned earlier, Lagrangian subproblems with additional constraints of this type can be easily solved using a minor variant of the knapsack algorithm. Since CCNFP involves minimizing a concave function over a network polyhedron with a single source and single destination for each commodity, it has an optimal solution in which each commodity $k \epsilon K$ flows on a single path from O(k) to D(k). (Therefore, this path has a flow of value R_k.) We exploit this property both in the Lagrangian-based reduction tests and to obtain further reduction based on the topology of the residual networks. We now list the series of tests used in the composite algorithm for the CCNFP that we tested.

Reduction 1:

For each commodity $k \epsilon K$ and each arc $(i,j) \epsilon$ Free(k), let $z^1(ij,k)$ be the Lagrangian objective function value (for the current best set of multipliers $v = \{v^k_i\}$) when the constraints

$$x^k_{ij} = R_k,$$

$$x^k_{ip} = 0 \qquad \text{for all } (i,p) \epsilon \text{Free}(k),\ p \neq j, \text{ and}$$

$$x^k_{pj} = 0 \qquad \text{for all } (lpj) \epsilon \text{Free}(k),\ p \neq i,$$

are added to the Lagrangian subproblem. If $z^1(ij,k) > z^+$, then commodity k MUST NOT flow on arc (i,j). In this case, we must update

$$\text{Closed}(k) \leftarrow \text{Closed}(k) \cup \{(i,j)\}, \text{ and}$$

$$\text{Free}(k) \leftarrow \text{Free}(k) - \{(i,j)\}.$$

Reduction 2:

For each commodity $k \epsilon K$ and arc $(i,j) \epsilon Free(k)$, let $z^2(ij,k)$ be the Lagrangian objective function value when the constraint

$$x^k_{ij} = 0$$

is added to the Lagrangian subproblem. If $z^2(ij,k) > z^+$, then commodity k MUST flow on arc (i,j), and we must update

$$Open(k) \quad <-- \quad Open(k) \cup \{(i,j)\}, \text{ and}$$

$$Free(k) \quad <-- \quad Free(k) - \{(i,j)\}.$$

Reduction 3:

For each commodity $k \epsilon K$ and every intermediate node $i \epsilon N$ that has no arc of Open(k) incident either to or from it, let $z^3(i,k)$ be the new Lagrangian objective function value when the constraints

$$x^k_{ij} = 0 \quad \text{for all } (i,j) \epsilon Free(k), \text{ and}$$

$$x^k_{pi} = 0 \quad \text{for all } (p,i) \epsilon Free(k),$$

are added to the Lagrangian subproblem. If $z^3(i,k) > z^+$, then commodity k MUST flow <u>via</u> node i. Moreover, if the set Free(k) contains exactly one arc, say (i,j) (or (p,i)), that is incident from (or incident to) node i, then commodity k MUST flow on this arc. In this case, we must update

$$Open(k) \quad <-- \quad Open(k) \cup \{(i,j)\} \quad (\text{or } Open(k) \cup \{(p,i)\}), \text{ and}$$

$$Free(k) \quad <-- \quad Free(k) - \{(i,j)\} \quad (\text{or } Free(k) - \{(p,i)\}).$$

Further, we must update the Lagrangian solution $\{x^k_{ij}{}^r\}$ and the current best lower bound $z^L(v)$ if x^k_{ij} (or x^k_{pi}) is zero in the current Lagrangian solution.

Reduction 4:

For each commodity $k \epsilon K$ and every intermediate node $i \epsilon N$ that has no arc of Open(k) incident to or from it, for all arcs $(i,j) \epsilon$ Free(k), let $z^4(ij,k)$ be the new Lagrangian objective function when the constraints

$$x^k_{ij} = R_k$$

$$x^k_{ip} = 0 \qquad \text{for all } (i,p) \epsilon \text{Free}(k),\ p \neq j$$

are added to the Lagrangian subproblem. $z^4(ji,k)$ is similarly defined for all arcs (j,i) belonging to Free(k). Let

$$z_{out}(i,k) = \text{Min } \{ z^4(ij,k) : (i,j) \epsilon \text{Free}(k) \}, \text{ and}$$

$$z_{in}(i,k) = \text{Min } \{ z^4(ji,k) : (j,i) \epsilon \text{Free}(k) \}.$$

Then, $(z_{out}(i,k) - z^L(v)) + (z_{in}(i,k) - z^L(v))$ represents the minimum increase in the Lagrangian objective function if commodity k is 'forced' to flow via node i. Therefore, if this increase is greater than $(z^+ - z^L(v))$, i.e., if

$$z_{out}(i,k) + z_{in}(i,k) > z^+ + z^L(v),$$

commodity k MUST NOT flow via node i. In this case, we must update

$$\text{Closed}(k) \leftarrow\!- \text{Closed}(k) \cup \{(i,j),(j,i) \epsilon \text{Free}(k)\}, \text{ and}$$

$$\text{Free}(k) \leftarrow\!- \text{Free}(k) - \{(i,j),(j,i) \epsilon \text{Free}(k)\}.$$

Further, if x^k_{ij} or $x^k_{ji} = R_k$ in the current Lagrangian solution for any arc $(i,j),(j,i) \epsilon$ Free(k), then $z^L(v)$ and the Lagrangian solution must be correspondingly updated.

In each of these methods, if the current Lagrangian solution does not violate any of the constraints that are added, then the test will not be

successful (since the lower bound will not increase when the additional constraints are incorporated). For instance, in Reduction 2, if x^k_{ij} is currently zero, then $z^2(ij,k) = z^L(v)$, and the test will be ineffective. Therefore, the number of arcs, nodes and commodities for which these tests must be performed might be considerably smaller than the total number of possible combinations. Besides these Lagrangian-based tests, we use the following conditions, based on the configuration of arcs belonging to the sets Free(k) and Open(k), to determine if further reduction is possible.

Other reduction methods:

(a) If arc (i,j) belongs to Open(k) for some commodity $k \epsilon K$, then commodity k MUST NOT flow on any other arc (i,p) incident from node i, or arc (p,j) incident to node j.

(b) For each commodity $k \epsilon K$ and each node $i \epsilon N$, if commodity k must flow through node i (for example, if i = O(k),D(k) or if arc (i,j) belongs to Open(k)) and if there is exactly one arc in the set Free(k) (and none in Open(k)) that is either incident to or from node i, then commodity k MUST flow on this arc.

(c) For each commodity $k \epsilon K$, if, using the arcs in the set Free(k) U Open(k), there is no path either from O(k) to some node i or from node i to D(k), then commodity k MUST NOT flow on any arc (i,j) or (j,i) belonging to Free(k).

As before, we update the sets Open(k), Closed(k) and Free(k) after each reduction. Further, if $x^k_{ij} = R_k$ in the current solution when arc (i,j) is added to Closed(k) or if $x^k_{ij} = 0$ when (i,j) is added to Open(k), the current lower bound $z^L(v)$ and the Lagrangian solution are recomputed.

This last set of tests does not involve solving the Lagrangian problem, and hence requires much less computational effort (compared to Reductions 1 to 4). It is therefore advantageous to apply one or all of these tests immediately after each of the first four reductions. For example, when arc (i,j) is added to Open(k) (say, using Reduction 2) we can immediately prohibit flow on all the other arcs (i,p) and (p,j) that belong to Free(k) and are incident either to node j or from node i. This intermediate step can thus reduce number of cases for which Reduction 1 must be applied in the next iteration.

Let us now examine some of the algorithmic implications of problem reduction. First, setting x_{ij}^k equal to 0 or R_k for some arcs (i,j) and commodities k might increase the lower bound, even with the current set of Lagrange multipliers (since the Lagrangian subproblems are more constrained now). In addition, a subsequent application of the multiplier adjustment and subgradient procedures might further increase this lower bound. Second, every time commodity k is prohibited from flowing on some arc (i,j), the upper limit on the possible flow M_{ij}^{r+} (and possibly the number of ranges) on arc (i,j) reduces by R_k. As a result, the Lagrangian subproblems can be solved faster. (Recall that, the time required to solve the subproblem $SP_{ij}(v)$ is proportional to the number of cost ranges for arc (i,j)). Also, the lower bound that is obtained using the multipliers initialization method that we discussed earlier improves, since the parameter $e_{ij} = c_{ij}^{r+} + F_{ij}^{r+}/M_{ij}^{r+}$ (and hence the lengths of the shortest paths using e_{ij} as arc lengths) increases as M_{ij}^{r+} decreases. Finally, by prohibiting commodity k's flow on arcs belonging to the set Closed(k), the local improvement procedure might yield better heuristic solutions (since

some solutions that were locally optimal before problem reduction might no longer be 'feasible'). Also, if we can modify the heuristic initialization and local improvement methods to always route commodity k through some or all arcs belonging to Open(k), the resulting upper bounds might be tighter.

Our implementation repeats the various problem reduction steps iteratively (not necessarily in the order in which they were presented) until no further reduction is possible. Also, for Reduction 1, our algorithm computes $z^1(ij,k)$ after adding the constraints

$$x^k_{ij} = 0, \text{ and}$$

$$\underline{\text{either}} \quad x^k_{ip} = 0 \text{ for all } (i,p) \epsilon \text{Free}(k),\ p \neq j, \qquad \text{(a)}$$

$$\underline{\text{or}} \quad x^k_{pj} = 0 \text{ for all } (p,j) \epsilon \text{Free}(k),\ p \neq i, \qquad \text{(b)}$$

to the formulation (instead of <u>both</u> (a) and (b) simultaneously as proposed earlier). Consequently, the new test is less stringent. Our implementation also contains a Preprocessing routine that is applied before initiating the composite algorithm. This procedure determines, for each node i and commodity k, if there is at least one path in the original graph from O(k) to node i and from node i to D(k). If not, commodity k's flow on all arcs incident to and from node i are set to zero and the upper limits on the flow on these arcs are reduced.

5.5 Lagrangian-based Heuristic Procedures

As in our algorithm for the Network Design problem, at intermediate stages of the Lagrangian relaxation procedure, we use the current best

Lagrangian solution to construct an initial feasible solution for the CCNFP. Using this solution as a starting point, the Flow Rerouting method determines a locally optimal solution. Let $\{v_i^k\}$ be the current best set of Lagrange multipliers and let $\{x_{ij}^{kr}\}$ be the corresponding optimal Lagrangian solution. Then, the initial heuristic solution is determined as follows: For each commodity $k \epsilon K$, let A^k be the set of arcs for which $x_{ij}^k > 0$ in the current solution and let K' be the subset of commodities k that have at least one path from O(k) to D(k) containing only arcs of A^k.

Each commodity $k \epsilon K'$ is routed on any arbitrary path from O(k) to D(k) in A^k.

For commodities k that do not belong to the set K':

(a) when applied immediately after the multiplier adjustment phase (i.e., in the Initial and Final Heuristic phases), the heuristic algorithm routes commodity k on the shortest path from O(k) to D(k) using the latest values of u_{ij}^k as arc lengths. (Recall that, u_{ij}^k is an underestimate of the maximum amount by which $(v_i^k - v_j^k)$ can be decreased without changing $z_{ij}(v)$), and

(b) when applied at intermediate stages of the subgradient procedure, the heuristic algorithm routes commodity k on the same path that was used in the previous application of the heuristic phase.

We adopted this approach after some preliminary experimentation with several alternative initialization schemes.

The principle underlying the Flow Rerouting local improvement technique for the CCNFP is the same as that outlined in Chapter 4, namely, to reroute the commodity with the best alternate routing at each iteration. The best alternate routing for any commodity k is the shortest path from O(k) to D(k) using p_{ij} as arc lengths, where

$$p_{ij} = \begin{cases} C_{ij}(f_{ij}) - C_{ij}(f_{ij}-R_k) & \text{if } (i,j) \epsilon P^k \\ C_{ij}(f_{ij}+R_k) - C_{ij}(f_{ij}) & \text{if } (i,j) \notin P^k, \end{cases}$$

with f_{ij} = Total flow on arc (i,j) in the current heuristic solution,

$C_{ij}(f)$ = Total cost of routing f units of flow on arc (i,j), and

P^k = path from O(k) to D(k) on which commodity k is currently routed.

The difference between the cost of the current routing and the length of this shortest path gives the maximum possible savings (with respect to the current routing pattern) that can be realized by rerouting commodity k. After computing this savings for all commodities, the algorithm reroutes the commodity with the greatest savings and repeats the procedure. The algorithm terminates when no further savings can be obtained by rerouting any commodity.

Before initiating the CCNFP algorithm, we use a 'stand-alone' procedure to generate an initial, locally optimal heuristic solution. The starting heuristic solution is obtained using the Sequential Routing algorithm described in Section 4.6. The Flow Rerouting method then finds a local optimum.

5.6 Structure of the Composite Concave-cost Network Flow Algorithm:

Having discussed the various components of the Composite CCNFP algorithm, we now present an outline of the overall algorithm. This outline illustrates the interrelationships between the different components within the algorithm.

Composite algorithm for the

Concave-cost Network Flow problem

Step 1: INITIALIZATION

For each commodity $k \epsilon K$ and every node $i \epsilon N$, set v_i^k equal to the length of the shortest path from O(k) to node i, using

$$e_{ij} = c_{ij}^{r+} + (F_{ij}^{r+}/M_{ij}^{r+})$$

as arc lengths.

Step 2: INITIAL ASCENT

Apply the multiplier adjustment method.

Step 3: INITIAL HEURISTIC

Construct an initial feasible solution using the results of the Initial ascent.
Find the locally optimal solution using the Flow Rerouting method.

Step 4: SUBGRADIENT OPTIMIZATION and INTERMEDIATE HEURISTIC

Apply the subgradient method to modify the Lagrange multipliers, using appropriate control parameters λ^0, t_{max}, t_h, and r.
Apply the heuristic at intermediate stages of subgradient procedure by initializing the Flow Rerouting method using the current best Lagrangian solution.

Step 5: FINAL ASCENT

Starting with the current best multipliers derived in Step 4, apply the multiplier adjustment method.

Step 6: FINAL HEURISTIC

Initialize the Flow Rerouting method using the final multipliers obtained in Step 5.

Step 7: PROBLEM REDUCTION

Use the current best multipliers and current best heuristic solution to fix variables in the problem.
Recalculate, for the best set of multipliers, the optimal Lagrangian solution for the reduced problem.

At the end of each run of this algorithm, it might be possible to

improve both the upper and lower bounds further, for instance, if either the subgradient procedure was terminated prematurely in the previous run or if the problem was reduced in Step 7 of the last run. In this case, the algorithm must be initiated again, starting with the current best Lagrange multipliers and heuristic solution; the multiplier initialization step (Step 1) and the initial heuristic step (Step 2) must be omitted in such 'continuation' runs; the initial ascent step (Step 3) may also be omitted, since the ascent procedure was applied (in Step 5) after subgradient optimization in the previous run. Next, we present computational results using this Composite algorithm for several randomly generated CCNFP test problems. Subsequently, in Section 5.6, we discuss variants of this algorithm.

5.7 Computational Results

We tested the Lagrangian-based algorithm for the CCNFP on two types of problems:

(1) General networks, which may have arbitrary topologies and demand patterns, and

(2) Three-layer networks, in which origins and destinations are distinct and do not serve as transshipment nodes, and every origin-destination path passes through exactly two intermediate nodes.

For both problem types, we randomly generated 5 problem instances for each of 5 problem sizes. The number of variables in the the IP formulations of these problems varied from 168 binary and 1680 continuous variables to 1488 binary and 89,280 continuous variables. We discuss the computational results for the General networks first.

5.7.1 GENERAL NETWORKS:

This type of problem has an arbitrary network configuration and every node of the given network is a candidate origin and/or destination. Transshipment is allowed at all nodes of the network.

Graph generation procedure:

The technique that we used to generate random General network problems is almost identical to the procedure described in Section 4.6 for generating Type A Uncapacitated Network Design problems. The CCNFP has a number of additional parameters, such as the demand R_k for each commodity, the number of cost ranges, the width of each range for each arc, and so on. The UNDP random network generator was therefore modified to generate values for these additional parameters. We next list some of these enhancements:

(a) For programming convenience, we assumed (both for the network generator and the Composite algorithm) that all the arcs have the same number of cost ranges initially. (Subsequently, the number of ranges may be decreased due to problem reduction.); the user must specify this number. However, the Composite algorithm can be used to solve problems with varying number of ranges by specifying, for each arc, a very large lower and upper limit for any 'redundant' ranges.

(b) The problem generator assumes that the demand for all commodities are identically and uniformly distributed between user-specified minimum and maximum values.

(c) By specifying the parameters appropriately, the user can make the variable cost c^1_{ij} for the first range on arc (i,j) directly or inversely proportional to the Euclidean length of arc (i,j); this cost may also include a random and a fixed component. In addition, the user must specify the percentage decrease in the variable cost from one range to the next. This decrease can also contain random and fixed components and can be specified separately for each range.

(d) In order to calculate the upper limit M^r_{ij} for range r on each arc

(i,j), the user must specify the width of range r in terms of the average demand. Again, this width can be specified separately for each range and can include both a fixed and a random term. For all arcs (i,j), the network generator sets the upper limit M^{r+}_{ij} equal to the sum of all demands.

As in the NDP problem generator, the origin and destination for each commodity are chosen so that the Euclidean distance between them is greater than a prespecified threshold value. This restriction encourages 'consolidation' of different commodities especially when the variable cost is directly proportional to the Euclidean distance. The user-specified arc DENSITY determines the number of arcs that are finally included in the network. To ensure that the problem instances are feasible, the problem generator adds to the network direct origin-destination arcs for each commodity . By changing just the seed for the random number generator and keeping all other parameters fixed, we can generate several problem instances of the same size (except for the number of arcs which is not under direct control of the user; however, this number is not likely to vary significantly across such instances). The network topology, demand pattern, and cost coefficients in each of these instances might, however, be different. The Preprocessing routine described earlier was appended to the graph generation procedure for convenience.

For our computational experiments, we tested the CCNFP algorithm on 5 problem sizes that were labeled GEN1 to GEN5 in order of increasing size. For each problem size, we generated 5 different instances by changing only the seed for the random number generator. Table 6 shows the problem sizes for each of the 5 categories.

Even though the network generator is capable of generating problems

Table 6: General Network Problem Size Parameters

Problem Size	GEN1	GEN2	GEN3	GEN4	GEN5
No. of NODES	10	15	20	30	40
ARCS: Density	0.5	0.5	0.5	0.4	0.2
Number	47 to 54	109 to 133	196 to 216	364 to 393	340 to 359
No. of COMMODITIES	10	20	30	50	60
No. of RANGES	4	4	4	4	4

with a wide variety of cost functions, range widths, and so on, we decided to eliminate much of the randomness in the different instances in order to make the comparison and interpretation more meaningful. Thus, for all our problems

(1) Number of cost ranges for each arc = 4,

(2) Demand for all commodities = 30,

(3) Variable cost on arc (i,j) is proportional only to the Euclidean distance d_{ij} between nodes i and j. In particular,

$$c^1_{ij} = 0.25*d_{ij},$$

$$c^2_{ij} = 0.60*c^1_{ij},$$

$$c^3_{ij} = 0.70*c^2_{ij}, \text{ and}$$

$$c^4_{ij} = 0.80*c^3_{ij}.$$

Thus, for instance, there is a 40 percent decrease in the per unit flow cost from range 1 to range 2.

The widths of each of the 4 cost ranges were chosen so that the optimal CCNFP solution contains, with high probability, several arcs that operate

in the 'higher' cost ranges in order to exploit the economies of scale. (If, in the optimal solution, all arcs operate in the first cost range, then the CCNFP is equivalent to a series of shortest path problems, one for each commodity, using c^1_{ij} as arc lengths). Table 7 gives the range widths (in terms of the average demand) for the 5 problem categories.

Table 7: Range widths for General Networks*

Problem \ Range:	1	2	3	4
GEN1	1	2	3	4
GEN2	1	2	3	9
GEN3	1	3	5	11
GEN4	1	3	5	21
GEN5	1	5	9	25

*All range widths expressed in terms of Average demand

Thus, the width of the second range on each arc in problem GEN4 is 3*Average demand = 90 units of flow. (As required, in each problem type, the sum of the range widths is equal to the number of commodities.)

Solving the General CCNFP Test problems:

The composite algorithm described in this chapter was coded in FORTRAN and tested using a PRIME 850 computer. It did not use any special data structures. Djikstra's [1959] algorithm was used to find shortest paths in the graph. While initiating the CCNFP algorithm, the user must specify several control parameters, such as the maximum number of subgradient iterations to be executed, the initial value of the step size multiplier, and so on. We now describe the settings used for solving the General

network test problems.

(1) The maximum number of subgradient iterations for the initial run of the CCNFP algorithm was set at 100. (The subgradient procedue might terminate earlier if either the step size becomes too small or the Lagrangian subproblem solution is primal feasible.) Subsequent 'continuation' runs of 100 subgradient iterations each were initiated if the gap between the best Lagrangian lower and the best upper bound was relatively large and if no significant problem reduction was achieved in the initial run. In continuation runs, the multiplier and heuristic initialization and the initial ascent phases were omitted.

(2) In the first run, the step size multiplier λ was initialized to 2.0; the step size multiplier was halved whenever the Lagrangian objective function value did not improve for 10 consecutive subgradient iterations. In subsequent runs, λ was initialized to half the initial value of the previous run (i.e., to 1.0 in the second run, 0.5 in the third run, and so on); it was halved after 5 consecutive 'no improvement' iterations.

(3) We observed, especially during continuation runs of the CCNFP algorithm, that a relatively large number of initial subgradient iterations were required before the Lagrangian objective function value improved. Consequently, our original step size updating rule halved the step size multiplier λ was several times before the algorithm improved the Lagrangian objective funciton value $z^L(v)$; this phenomenon invariably caused slower subsequent improvement and early termination of the subgradient phase (because of small step size). We, therefore, introduced an additional user-specified parameter t_{init}, and modified the step size updating rule so that the step size multiplier is not changed during the first t_{init} iterations or until the first improvement in $z^L(v)$ is realized, whichever occurs earlier. In our computational experiments, we set t_{init} equal to 20 iterations for all runs.

(4) In the CCNFP algorithm, the user can control the frequency with which the intermediate heuristic procedure is applied during subgradient optimization. In all of our runs, the heuristic procedure was initiated once every 10 subgradient iterations. If the initial heuristic solution obtained from the current Lagrangian solution is identical to the initial solution derived during the previous execution of the heuristic procedure, then the local improvement algorithm is not applied. Also, recall that, for each commodity k, some path from O(k) to D(k) using arcs with positive flow in the current Lagrangian solution is selected as the initial routing for commodity k. Our algorithm terminates the heuristic procedure if no such path exists for more than 10 percent of the commodities; otherwise, for all commodities that cannot be routed in this manner, the algorithm uses the previous initial routing.

(5) After some initial experimentation, we set the discounting factor for computing the weighted subgradient equal to 0.2 for all the runs.

Computational results for General CCNFP problems:

Table 8 shows the average, minimum and maximum values of different performance measures for each of the five problem sizes. (Appendix B contains the statistics for individual problem instances.) We now interpret some of the key figures. We express all indicators pertaining to the lower and upper bounding components of the algorithm as a percentage of the best upper bound that was obtained finally.

Quality of the final Lagrangian lower bounds:

On average, over all the 25 problem instances, the final lower bound as a percentage of the best upper bound was 98.3 percent. In 4 out of the 25 instances, this gap was zero, while the largest gap was 5.4 percent. As might be expected, this gap seems to be greater for the larger problems.

Effectiveness of the Lagrangian-based heuristic:

In all but 4 instances, the Lagrangian-based heuristic improved upon the initial heuristic solution (generated using the stand-alone algorithm). On average, the Lagrangian-based heuristic improved the solution by 2.3 percent (expressed as a percentage of the best upper bound), while the maximum improvement was 9.0 percent.

Effectiveness of the Initialization procedure:

The initial lower bound as a percentage of the best upper bound was 46.8 percent on average, with a high and low of 58.3 percent and 39.6 percent respectively. As the problem size increases, the percentage gap between the initial lower bound and the best upper bound seems to increase.

Effectiveness of the Multiplier Adjustment procedure:

The percentage improvement in the Lagrangian lower bound due to the Initial Ascent phase was markedly higher than the improvement brought about by subsequent multiplier adjusment phases. The average, minimum and maximum improvements caused by the initial ascent and subsequent ascent phases (as a percentage of the best upper bound) are

TABLE 8
Summary statistics for test runs
on General Network problems

Problem class	SITR	IUB% BUB	ILB% BUB	LBIA% BUB	FLB% BUB	% Free Var Init	% Free Var Final
GEN1	72.8	100.2	54.4	74.8	99.9	81.1	2.9
GEN2	155.4	103.1	48.5	72.7	98.7	87.3	23.4
GEN3	158.8	103.8	45.4	70.3	98.4	90.2	39.1
GEN4	279.2	103.9	44.6	66.4	96.2	93.5	87.0
GEN4	279.6	100.6	40.9	63.9	98.5	95.0	73.8

Legend:
SITR = No. of Subgradient iterations
ILB = Initial Lower bound (after Step 1)
LBIA = Lower Bound after Initial Ascent (Step 2)
FLB = Final Lower bound
IUB = Initial Upper bound
BUB = Best Upper bound
% Free variables : expressed as a % of Total no. of possible flow variables.

(All figures are average values for 5 instances).

Problem class	CPU time (secs) for Init Ascent	Subseq Ascent	Subgrad Opt	Prob Reducn	Heur.*
GEN1	3.1	1.0	21.5	2.0	0.4
GEN2	26.1	7.7	138.9	72.2	2.8
GEN3	140.7	61.8	444.0	366.9	14.2
GEN4	1026.0	1091.0	3299.4	4975.8	139.7
GEN5	1757.5	1058.7	4034.0	8963.2	110.2

CPU times in seconds on PRIME 850
* in terms of seconds per successful application of heuristic algorithm

	Initial Ascent	Subsequent Ascent
Average improvement	22.9	0.1
Minimum improvement	13.2	0.0
Maximum improvement	27.7	0.3

In 7 out of the 25 problem instances, no improvement was achieved in the subsequent ascent phases. The percentage improvement due to Initial ascent does not seem to depend on the problem size.

Performance of the Subgradient procedure:

As the problem size increases, more subgradient iterations are required before the procedure terminates (due to small step size). As a percentage of the best upper bound, the average, minimum, and maximum improvements in $z^L(v)$ are 28.7 percent, 18.8 percent and 35.3 percent, respectively. (These percentage improvement figures include the improvement due to final ascent; however, as indicated earlier, the final ascent phase did not improve the lower bound by more than 0.3 percent in any instance). The subgradient procedure consistently increased the Lagrangian lower bound by more for the largest problem size GEN5; however, for the other four problem sizes, there is no discernible relationship between this percentage improvement and the problem size.

Effectiveness of the Problem Reduction procedure:

The extent to which problems are reduced appears to be very sensitive to the absolute magnitude of the gap between the Lagrangian lower bound and the best upper bound, rather than the percentage gap. This result is to be expected since the cost coefficients are of the same magnitude for all instances and since the effectiveness of the reduction tests hinges on the absolute difference $(z^+ - z^L(v))$. On average, the Preprocessing routine eliminated 10.6 percent of the original 'flow' variables x_{ij}^{kr}. The number of variables eliminated is smaller for larger problems, even though the larger problems are sparser. To measure the effectiveness of the Lagrangian-based problem reduction phase, we use the following indicator:

$$PR = 1 - \frac{\text{percent of free flow variables at end of CCNFP alg}}{\text{percent of free flow variables after Preprocessing}}$$

This index was 48.5 percent, on average, over all problem instances. This procedure fixed all the variables for 2 out of the 25 problem instances, while it did not fix any variable for 1 instance.

Computational requirements:

The figures for CPU times in Table 8 show that the computational

requirements grow very rapidly as the problem size increases. Notice also that, for larger problems, the problem reduction phase requires more time than the other components of the algorithm. In retrospect, this time could have been reduced considerably, if for larger problems we had specified a maximum limit of 200 or 300 subgradient iterations in the intial run, rather than the 100 iteration limit that we used. For almost all the instances of problems GEN4 and GEN5, we had to run the composite algorithm three times before the improvement in the Lagrangian objective function value began to taper off. (The first two runs invariably terminated only because of the iteration limit and not because the step size became too small.) In all these cases, the improvement in the first two runs was insufficient to lead to any significant problem reduction. However, since the problem reduction phase is initiated once in every run, and since all the tests are performed at least once in this phase, the first two runs entailed substantial wasted effort. In addition, as we mentioned earlier, the first 20 or so subgradient iterations in each subsequent run (after the first run) did not usually improve the lower bound. Instead, if we had allowed the first run to continue for say 200 or 300 subgradient iterations, the total time for problem reduction could have been reduced by approximately 2/3rds, and the total time for subgradient optimization would also have decreased significantly. Additional savings could have been achieved by employing efficient sorting routines (required for solving the Lagrangian subproblems at each stage), special data structures and updating procedures, and an efficient shortest path subroutine.

Next, we discuss the results for the second problem type that we tested - Three-layer networks. Subsequently, we will compare the results for the two problem classes and attempt to justify some of the differences.

5.7.2 THREE-LAYER NETWORKS:

This problem type is a special case of the CCNFP in which

(1) nodes of the network are classified into 4 types - Source nodes, Consolidation points, Breakbulk points, and Destination nodes,

(2) transshipment is permitted only at consolidation and breakbulk points, and

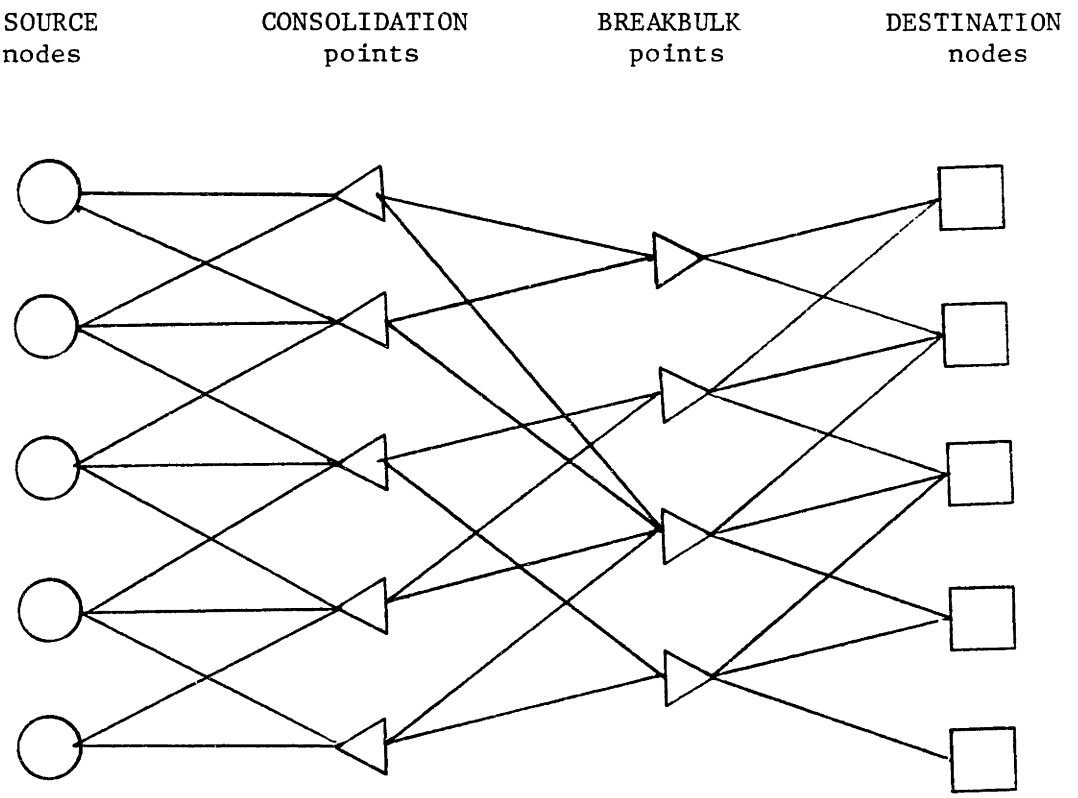
(3) the network contains only three categories of arcs -

Source-Consolidation arcs, Consolidation-Breakbulk arcs, and Breakbulk-Destination arcs.

Thus, the origins and destinations are distinct and do not serve as intermediate nodes. Every commodity must be transported across three 'layers' of the network. Figure 20 shows the configuration of a typical network of this type.

As we described earlier, consolidation points are the nodes at which incoming 'Less-than-Truckload' (LTL) shipments from the various sources are consolidated into truckloads (to exploit the economies of scale in transportation costs) before being dispatched to the breakbulk points. At the breakbulk nodes, incoming truckloads are 'broken', sorted destination-wise, and forwarded (perhaps as LTL shipments) to their respective destinations. The consolidation-to-breakbulk arcs are often referred to as 'Truckload lanes'. In our problems, we permit economies of scale (i.e., piecewise linear concave cost functions) on all arcs of the network; however, in many situations, such cost functions might apply only to the truckload lanes, with the transportation cost being linear on all other arcs. Note that, although only 3 types of arcs are permitted, it is possible to model other types of connections, such as direct source-to-Desination links, by introducing dummy consolidation and breakbulk nodes. This type of model is very useful when, for operational reasons, the load plan requires that no consolidated shipment is transshipped at intermediate points. If on the other hand, a maximum of say 2 transshipments per bulk load is permitted, we can use a similar five-layer network to model the problem.

Figure 20: Example of a Three-layer Network



Graph generation procedure:

We next summarize the differences between the random network generator for Three-layer problems vis-a-vis the procedure for General networks.

Parameters specified by the user:

For the Three-layer problems, the user must specify

(a) the number of source, consolidation, breakbulk, and destination nodes, denoted as n_S, n_C, n_B, and n_D respectively. The specified number of commodities must be between Max $\{n_S, n_D\}$ and $n_S * n_D$. This condition ensures that each source and destination node is utilized (i.e., is the origin and destination, respectively, for at least one commodity) and that all commodities have distinct origin-destination pairs; and

(b) the density of the source-consolidation, consolidation-breakbulk, and breakbulk-destination arcs, denoted as d_{SC}, d_{CB}, and d_{BD}, respectively.

Locating nodes on the 100 x 100 grid:

Within the 100 x 100 grid, the network generator randomly locates

- source nodes in the [0,20] x [0,100] rectangle,
- consolidation and breakbulk points in the [20,50] x [0,100] and [50,80] x [0,100] rectangles, respectively, and
- destination nodes in the [80,100] x [0,100] rectangle.

This choice ensures that, for all commodities, the minimum Euclidean distance between the origin and destination is 60.

Selecting the origin and destination for each commodity:

For the first n_{max} = Max $\{n_S, n_D\}$ commodities, the procedure chooses the origins and destinations in sequence (i.e., source node 1 and destination node 1 serve as the origin and destination, for commmodity 1, and so on); thus, all the source and destination nodes are utilized. For the remaining commodities, distinct origin-destination pairs are randomly selected.

Generating arcs of the network:

For each source node, the network generator identifies the ($d_{SC} * n_C$)

closest (in terms of Euclidean distance) consolidation points and includes the corresponding source-consolidation arcs in the network. Similarly, each destination node is connected to the ($d_{BD}*n_B$) closest breakbulk points. This choice reflects the characteristic of practical problems in which each source and destination is typically connected to a few of the closest End-of-Line (consolidation and/or breakbulk) terminals. The consolidation-breakbulk arcs are chosen randomly with probability d_{CB}.

Verifying problem feasibility:

Unlike the General network problems, Three-layer problems do not contain any direct origin-to-destination arcs. Hence, for each commodity, it is necessary to check if the current graph has at least one path from the commodity's origin to its destination, in order to ensure that the problem is feasible. If some commodity k does not have any path from O(k) to D(k), an appropriate consolidation-breakbulk arc is randomly added to ensure feasibility. (Note: Both O(k) and D(k) will always be connected to at least one consolidation and breakbulk point, respectively, because of our arc generation scheme.)

As before, a Preprocessing routine executed at the end of the graph generation procedure identifies, for each commodity k, all arcs that do not lie on any path from O(k) to D(k). This procedure reduces the upper limit of flow (and the number of ranges, if necessary) on all such arcs.

For our computational experiments, we generated problems in 5 different sizes, labeled LTL1 to LTL5. In each problem size, we generated 5 problem problem instances using different seeds for the random number generator. Table 9 specifies the problem sizes for each category.

All other problem generator parameters - for the Demand distribution, Variable cost structure, and Range widths - were identical to those used for generating the General CCNFP test problems.

Table 9 : Three-layer network problem size parameters

Problem Size	LTL1	LTL2	LTL3	LTL4	LTL5
No. of SOURCE nodes	4	5	6	8	10
CONSOLIDN nodes	5	10	12	15	20
BREAKBULK nodes	5	10	12	15	20
DESTN nodes	4	5	6	8	10
DENSITY of arcs:					
Source-Consolidn	0.5	0.5	0.5	0.5	0.4
Consolidn-Breakbulk	0.8	0.8	0.8	0.8	0.5
Breakbulk-Destn	0.5	0.5	0.5	0.5	0.4
Total no. of ARCS	42 to 47	131 to 141	190 to 207	309 to 312	358 to 372
No. of COMMODITIES	10	20	30	50	60
No. of RANGES	4	4	4	4	4

Solving the Three-layer test problems:

We used the general CCNFP algorithm, without any modifications for exploiting the special LTL structure, to solve the 25 test instances of the Three-layer problem. All the solution parameters in the CCNFP algorithm - the Maximum number of subgradient iterations per run, the initial value of the step size multiplier, the number of 'no improvement' iterations after which the step size multiplier is halved, the number of initial subgradient iterations for which the step size multiplier is not changed, and the frequency with which the intermediate heuristic procedure is applied - had the same values used for solving General networks.

Computational results for the Three-Layer problems:

Table 10 presents the summary statistics for the five problem sizes. (Appendix C gives the statistics for individual problem instances.) As before, we consider the performance of each component of the composite procedure in turn. All the improvements in the upper and lower bounds are evaluated in terms of percentages of the best final upper bound.

Quality of the Final Lagrangian lower bound:

The average value of the Final lower bound as a percentage of the best upper bound was 99.6 percent while the largest gap was 2.5 percent. In 19 out of the 25 problem instances, there was no gap between the final lower bound and the best upper bound, indicating that the optimal solution had been found in all these cases. (The number of such instances in each of the problem sizes LTL1 to LTL5 were 4, 5, 4, 3, and 3 respectively).

Effectiveness of the Lagrangian-based heuristic procedure:

In all but 3 instances, the Lagrangian-based heuristic improved upon the initial heuristic solution. As a percentage of the best upper bound, the Lagrangian-based heuristic improved the initial solution by an average of 6.8 percent; the maximum improvement was 24.8 percent.

Effectiveness of the Initialization procedure:

The average, minimum, and maximum values of the initial lower bound as a percentage of the best upper bound were 89.6 percent, 83.8 percent, and 95.4 percent respectively. For this class of problems, therefore, the multiplier initialization method seems to be very effective. We later discuss some of the likely reasons for this behavious.

Effectiveness of the multiplier adjustment method:

As before, the initial ascent phase gives significantly better improvements in the Lagrangian lower bound than the subsequent multipler adjustment phases. The average, minimum, and maximum values of the improvement in the lower bound as a percentage of the best upper bound are

TABLE 10
Summary statistics for test runs
on Three-layer Network problems

Problem class	SITR	IUB% BUB	ILB% BUB	LBIA% BUB	FLB% BUB	% Free Var Init	% Free Var Final
LTL1	71.2	103.7	92.6	96.7	99.8	30.4	5.9
LTL2	68.6	113.1	90.1	93.8	100.0	29.0	0.1
LTL3	100.4	107.6	87.4	92.6	99.6	26.8	5.1
LTL4	104.2	104.2	88.7	93.4	99.1	21.9	8.1
LTL4	134.4	105.4	89.0	93.5	99.5	13.4	2.9

Legend:
SITR = No. of Subgradient iterations
ILB = Initial Lower bound (after Step 1)
LBIA = Lower Bound after Initial Ascent (Step 2)
FLB = Final Lower bound
IUB = Initial Upper bound
BUB = Best Upper bound
% Free variables : expressed as a % of Total no. of possible flow variables.

(All figures are average values for 5 instances.)

Problem class	CPU time (secs) for Init Ascent	Subseq Ascent	Subgrad Opt	Prob Reducn	Heur.*
LTL1	1.2	0.3	11.4	1.1	0.5
LTL2	5.0	0.5	48.7	2.9	2.8
LTL3	12.7	2.9	144.1	17.5	9.0
LTL4	34.2	25.0	399.7	157.8	52.4
LTL5	43.3	42.4	582.7	102.9	72.2

CPU times in seconds on PRIME 850
* in terms of seconds per successful application of heuristic algorithm.

	Initial Ascent	Subsequent Ascent
Average increase	4.45	0.05
Minimum increase	1.11	0.0
Maximum increase	7.73	0.61

Effectiveness of the Subgradient procedure:

Again, larger problems required more subgradient iterations before termination. The average, minimum, and maximum increase in the Lagrangian lower bound (as a percentage of the best upper bound) due to the subgradient procedure were 5.6 percent, 1.7 percent, and 9.9 percent respectively. This percentage improvement does not seem to depend on the problem size.

Effectiveness of the Problem Reduction procedure:

On average, the Preprocessing routine eliminated 75.7 percent of the original flow variables. This routine eliminates a higher percentage of variables for larger problems. One likely reason for this higher reduction is that the larger problems are sparser; consequently, for each commodity k, the network probably contains a larger proportion of arcs that do not lie on any path from O(k) to D(k). The Lagrangian-based problem reduction phase fixed on average 80.4 percent of the variables remaining after preprocessing. This phase fixed all the variables for 16 out of the 25 problem instances. (Of these, 2, 4, 4, 3, and 3 instances, respectively, belonged to problem categories LTL1 to LTL5.)

Computational requirements:

Many of the comments that we made about the implementation in the context of the General Network problems are also applicable here. For instance, for larger problems, we should have specified a higher iteration limit for the subgradient procedure in the first run of the composite algorithm. In contrast with the General network problems, Three-layer problems required significantly lesser time required for problem reduction than for the subgradient phase. Also, all the problem instances required at most two runs of the composite algorithm before the lower bound converged. Finally, the CPU times might have been smaller if we had used a specialized code that exploits the structure of these networks. For instance, it is possible to identify shortest paths in Three-layer networks in O(m) time or less by suitably indexing the nodes and arcs.

5.7.3 Comparison of results for General and Three-layer problems:

The Three-layer problems that we tested were obviously easier to solve using our algorithm than the General network problems. For 19 out of the 25 Three-layer problem instances, the Composite algorithm reduced the gap between the upper and lower bound to zero compared to only 4 such instances for the General network problems. Furthermore, the average gap for LTL problems was only 0.4 percent compared to 1.67 percent for General network problems, and the CPU times required to solve the former were an order of magnitude smaller. The performance of every component of the algorithm was superior for the Three-layer problems. First, the Preprocessing routine was significantly more effective for these problems (75.7 percent reduction compared to 10.6 percent for General networks). This improved performance was probably due to the special structure of these problems. Because of the significant reduction in problem size at the preprocessing stage, the initial lower bound was on average 89.6 percent of the best upper bound, as compared to only 46.8 percent for General problems. (Recall that, besides eliminating flow variables, the Preprocessing routine also tightens the upper bounds on flow in each arc; since the initial values of the Lagrange multipliers are based on the cost per unit of flow at the upper bound, the tighter these bounds, the closer the initial multipliers will be to the 'optimal' set of multipliers). Since the percentage gap between the lower and upper bound is relatively small from the beginning, the Ascent and Subgradient phases did not improve the bounds as much for Three layer problems as for General Network problems. Also, since the gaps were smaller, more reduction was possible, and the Three-layer problems required fewer runs (and hence lesser CPU time). Finally, the Lagrangian-based

heuristic procedure also performed better for this class of problems (improving the initial heuristic solution by 6.8 percent compared to 2.3 percent for the General network problems). Since the final set of Lagrange multipliers were closer to the optimal values, the Lagrangian-based initial solutions might have been better than for General network problems. Thus, the effectiveness of each component of the composite algorithm contributes to increasing the effectiveness of subsequent phases.

We might expect the problems to be more difficult to solve as the percentage decrease in per unit cost from one range to the next increases. In our test problems, the demand for all the commodities was fixed at a constant value. Having different demands for each commodity, should not, in our judgement, increase the problem difficulty. Finally, in the Three-layer problem, if the cost functions on all but the consolidation-to-breakbulk arcs are linear, then the problem can perhaps be solved more efficiently using a Concave-cost Plant location algorithm. Next we discuss some variants of the Composite CCNFP algorithm.

5.8 Variants of the Composite Algorithm

So far, we have just described one implementation of the CCNFP algorithm; however, several alternative approaches can be adopted for each component of this algorithm. We outline some of these variants in this section.

5.8.1 The Lagrangian relaxation scheme:

Our Lagrangian relaxation scheme dualizes the flow conservation equations. Alternatively, if we dualize the forcing constraints (15.3) and the variable upper and lower bounding constraints ((15.4) and (15.5) respectively) of [CCNFP], the Lagrangian subproblems reduce to a series of Shortest path problems, one for each commodity. For this relaxation, however, the Lagrangian subproblem satisfies the Integrality property; hence, the best possible lower bound using this approach is the optimal value of the LP relaxation of [CCNFP]. (As we mentioned earlier, Geoffrion and McBride [1978] observed that dualizing the flow conservation equations often gives much better lower bounds for the Capacitated Facility location problem.) This method has the additional disadvantage of requiring a much larger number of Lagrange multipliers than our scheme; hence, the procedures for updating the multipliers might require more computational effort. As we noted in Section 4.2, several other Lagrangian relaxation schemes are possible; instead of discussing these alternatives, we describe an enhancement of our relaxation scheme.

Recall that, when we dualize the flow conservation equations (15.2) using multipliers v_i^k, the resulting Lagrangian subproblem [SP(v)] separates by arc and the optimal solution for each subproblem $[SP_{ij}(v)]$ can be determined by solving r^+ Knapsack problems $[SP_{ij}^r(v)]$, one corresponding to each cost range on arc (i,j). Suppose we 'strengthen' this relaxation by adding the constraints

$$\sum_j \sum_r M_{ij}^r y_{ij}^r \geq \sum_{k \in K(i)} R_k \qquad \text{for all } i \in N, \qquad (17.1)$$

where $K(i) = \{k \epsilon K : O(k) = i\}$, the set of all commodities with node i as the origin.

Constraint (17.1) specifies that the sum of the upper limits or capacities of the chosen ranges for all arcs incident <u>from</u> node i must exceed the total flow that must be routed from node i. Similarly, we could add the constraints

$$\sum_{j} \sum_{r} M^{r}_{ji} y^{r}_{ji} \geq \sum_{k \epsilon K'(i)} R_k \qquad \text{for all } i \epsilon N, \qquad (17.2)$$

where K'(i) is the set of all commodities with node i as the destination. In fact, if in the problem reduction phase we determine (using Reduction 3, for example) that commodity k must flow <u>via</u> node i, then this commodity can be added to both the sets K(i) and K'(i), thus strengthening the two inequalities (17.1) and (17.2) corresponding to node i. Let us now discuss how, for any given values of the Lagrange multipliers v^k_i, the Lagrangian subproblems containing these additional constraints can be solved.

We will consider the case when constraints (17.1) are added to the formulation. The method that we describe is also applicable when constraints (17.2) are included. For any node i, constraint (17.1) is not redundant in the Lagrangian subproblem only if the corresponding subset of commodities K(i) is nonempty. Consider one such node. Since (17.1) links the flow variables on all the arcs incident from node i, the Lagrangian subproblem no longer separates completely by arc. Instead, it now has two types of subproblems:

(a) subproblems $[SP_{ij}(v)]$ corresponding to arc (i,j) if K(i) is empty, and

(b) subproblems $[SP_i(v)]$ corresponding to nodes i for which K(i) is nonempty.

The 'arc' subproblems $[SP_{ij}(v)]$ can be solved as before - by solving r^+

Knapsack problems. To solve the 'node' subproblems, however, we must use the following Dynamic programming algorithm. For any given set of multipliers $\{v_i^k\}$, let $z_i(v)$ and $z_{ij}^r(v)$, respectively, represent the optimal values of the node subproblem $[SP_i(v)]$ and the Knapsack problem $[SP_{ij}^r(v)]$ described earlier. Assume, for convenience, that the original graph contains arcs (i,j) for all j=1,2,...,n. Then, the subproblem $[SP_i(v)]$ can be written as

$[SP_i(v)]$

$$\text{Minimize} \quad \sum_{j=1}^{n} \sum_{r} z_{ij}^r(v) y_{ij}^r$$

subject to

$$\sum_{j=1}^{n} \sum_{r} M_{ij}^r y_{ij}^r \geq R(i)$$

$$\sum_{r} y_{ij}^r = 1 \qquad \text{for all } j=1,2,\ldots,n$$

$$y_{ij}^r = 0 \text{ or } 1 \qquad \text{for all } j=1,2,\ldots,n,\ 1 \leq r \leq r^+,$$

where $R(i) = \sum_{k \in K(i)} R_k$.

For each $j^* = 1,2,\ldots,n$, and $0 \leq b \leq R(i)$, define the quantity $h_{j^*}(b)$ as follows:

$$h_{j^*}(b) = \text{Min} \sum_{j=1}^{j^*} \sum_{r} z_{ij}^r(v) y_{ij}^r$$

subject to

$$\sum_{j=1}^{j^*} \sum_{r} M_{ij}^r y_{ij}^r \geq b$$

$$\sum_{r} y_{ij} = 1 \qquad \text{for } j=1,2,\ldots,j^*$$

$$y_{ij}^r = 0 \text{ or } 1 \qquad \text{for } j=1,2,\ldots,j^*,\ 1 \leq r \leq r^+$$

Thus, by definition, the optimal value $z_i(v)$ of the subproblem $[SP_i(v)]$ is $h_n(R(i))$.

Claim:

$h_{j^*}(b)$ can be determined using the following Dynamic Programming recursion:

$$h_{j*}(b) = \begin{cases} \min\limits_{1 \le r \le r^+} \{ z^r_{ij*}(v) + h_{j*-1}(b - M^r_{ij*}) \} & \text{if } b \le \sum\limits_{j=1}^{j*} M^{r+}_{ij} \\ \infty \quad \text{otherwise.} \end{cases}$$

Using this recursion and the initial conditions

$$h_1(b) = \begin{cases} \min\limits_{r\,:\,M^{r+}_{i1} \ge b} z^r_{i1}(v) & \text{if } b \le M^{r+}_{i1} \\ \infty \quad \text{otherwise} \end{cases}$$

$$\text{for all } 0 \le b \le R(i),$$

we can find the optimal solution and value of the 'node' subproblem $[SP_i(v)]$. By embedding this Dynamic programming algorithm in an iterative procedure for appropriately modifying the multipliers, we might be able to derive tighter lower bounds than with our original method. However, note that the multiplier adjustment method that we used before must be modified for this tighter relaxation; since the subproblems are harder to solve now, the ascent method might also be more complex.

Another variant of our Lagrangian scheme is to dualize, for each commodity k, the flow conservation equations for just the 'intermediate' nodes (i.e., all nodes other than O(k) and D(k)). In this case, the Lagrangian subproblem would contain, for all commodities k, the additional constraints

$$\sum_j \sum_r x^k_{O(k)j} = R_k, \text{ and}$$

$$\sum_j \sum_r x^k_{jD(k)} = R_k.$$

Again, these constraints link the flow variables for all arcs that are incident from origin nodes and for those that are incident to destination nodes. The corresponding node subproblems are, however, more difficult to

solve than the subproblems $[SP_i(v)]$ that we just considered.

5.8.2 The Multiplier Adjustment method:

In the multiplier adjustment method outlined earlier, we attempt to change the Lagrange multipliers v_i^k so that

(1) $z_{ij}(v)$ remains unchanged for all arcs (i,j), and

(2) $v^k_{D(k)}$ increases for at least one commodity k.

For this purpose, we developed an underestimate u_{ij}^k of the maximum permissible decrease in $(v_i^k - v_j^k)$; we increased the v values by solving a shortest path problem that uses these estimates u_{ij}^k as arc lengths. While it is possible to derive a more accurate estimate of the maximum permissible decrease in $(v_i^k - v_j^k)$, the computational effort required for such a procedure would be prohibitive.

An alternative ascent strategy is to modify the Lagrange multipliers so that

(1) $z_{ij}(v)$ increases for at least one arc (i,j), and

(2) for all commodities k, $v^k_{D(k)}$ either increases or remains it at its current value.

For commodities k with $x_{ij}^k = R_k$ in the current Lagrangian solution, this strategy requires an additional estimate of the amount by which $(v_i^k - v_j^k)$ can be increased before it becomes optimal to set $x_{ij}^k = 0$.

5.8.3 Problem reduction:

Our problem reduction procedure attempts to identify, for all arcs

(i,j), commodities k that MUST or MUST NOT flow on (i,j) in the optimal [CCNFP] solution. In addition, we can devise Lagrangian-based methods to reduce the 'capacity' or upper limit M_{ij}^{r+} of each arc (i,j) as far as possible. As we pointed out earlier, lowering the limit could potentially decrease the computational time required as well as increase the lower bound obtained using the multiplier initialization method. One way to reduce the upper limit is by adding an 'opposite' constraint of the form

$$\sum_{k} \sum_{r} x_{ij}^{kr} \geq b \qquad \text{for some } b < M_{ij}^{r+}.$$

If the new Lagrangian objective function value (for the current set of multipliers) exceeds z^+ (the value of the current incumbent), then M_{ij}^{r+} can be reduced to b (and, if necessary, the number of ranges for arc (i,j) can be reduced). Subproblems with additional constraints such as this can easily be solved parametrically with respect to b using a variant of our original Lagrangian subproblem algorithm. Similarly, we can determine a lower limit for flow on arc (i,j) by adding a constraint similar to the last one, but with the sign of the inequality reversed. Indeed, a strategy that alternates between

(1) such a Range restriction phase, and

(2) a reinitialization of the multipliers based on the new upper limits M_{ij}^{r+},

might, by itself, result in a substantial increase in the Lagrangian lower bound.

5.8.4 Heuristic procedures:

The Lagrangian-based heuristic algorithm uses the current Lagrangian

solution to generate an initial feasible CCNFP solution. Starting with this initial solution, it makes routing improvements until a local optimum is found. In the procedure that we proposed earlier, any path from O(k) to D(k) that uses only arcs (i,j) with $x^k_{ij} > 0$ in the current solution is chosen as the initial route for commodity k. If no such path exists, commodity k is routed on the shortest path using the estimates u^k_{ij} as arc lengths. An alternative initialization method is the following: for each commodity k, route commodity k on the shortest path from O(k) to D(k) using $(v^k_j - v^k_i)$ as arc lengths, where $\{v^k_i\}$ is the current best set of Lagrange multipliers. This solution can be interpreted as the optimal subproblem solution for a Lagrangian relaxation scheme that dualizes all constraints of [CCNFP] except the flow conservation equations (15.2).

Based on some preliminary computational experiments, we found that, in most instances,

(1) this method derived initial solutions whose costs were significantly higher than the cost of the final local optimum; consequently, it required many more local improvement iterations than our earlier method, and

(2) the best upper bound obtained using this method was higher than both the cost of the 'stand-alone' heuristic solution and the best upper bound provided by our original method.

A different approach for initializing the local improvement procedure uses the problem reduction procedure iteratively. Given the current set of Lagrange multipliers $\{v^k_i\}$, the current best Lagrangian lower bound $z^L(v)$ and an upper bound z^+, the problem reduction procedure identifies arcs on which each commodity must or must not necessarily flow. The gap between $z^L(v)$ and z^+ determines the extent to which such reduction is possible; if

the gap is small, the procedure can fix more arcs as Open or Closed for each commodity. Suppose, starting with z^+ equal to the value of the current incumbent, we repeat the problem reduction procedure for a decreasing sequence of "hypothetical" upper bounds z^+. (In other words, we apply the problem reduction procedure with different values of the upper bound even though we have not identified primal feasible solutions with these costs). As z^+ decreases, the number of possible paths for each commodity in the residual network (after problem reduction) decreases. For some value of $z^+ \geq z^L(v)$,

either (1) the problem reduction procedure must deduce contradictory results; for example, it might indicate that some arc (i,j) must be both Closed and Open for some commodity k,

or (2) after reduction, there must be a unique path from O(k) to D(k) using arcs belonging to the set Open(k) U Free(k), for all commodities $k \in K$.

Then, the initial heuristic solution can be obtained from the 'reduced' graph corresponding to this value of z^+. (In case (2), this solution is unique). We have not tested such a heuristic initialization scheme.

5.9 Concluding Remarks

In this chapter we developed a composite procedure for finding good upper and lower bounds for a special case of the Network Design problem that is of considerable practical importance. The ascent method that we used for increasing the Lagrangian lower bound is similar in spirit to the dual ascent methods that we discussed in Chapter 4. By combining it with a subgradient procedure, a problem reduction phase, and a Lagrangian-based

heuristic algorithm, we were able to solve fairly large Concave-cost network flow problems. This Composite algorithm exploits the special structure of the CCNFP algorithm; therefore, we might expect it to outperform the general dual ascent procedure discussed in Chapter 4. (The latter algorithm is also applicable since the CCNFP can be modeled as a Network Design problem defined over an equivalent graph.) The computational results confirm the usefulness of a combined strategy with lower and upper bounding schemes that are based on the partial optimization of some related problems. Also, the performance of such algorithms seems to improve substantially when the networks have certain special structures.

The Lagrangian relaxation scheme that we used dualizes the flow conservation equations. Another problem for which such a scheme might be perform well is the Concave-cost network flow-with-gains problem that arises frequently in the production planning context. For instance, in assembly systems, the flow patterns do not satisfy the flow conservation equations (15.2); therefore, the coefficients in the left-hand side of constraints (15.2) in the IP formulation of this problem are not necessarily equal to 1. Since our scheme dualizes these constraints, the structure of the subproblems remain the same in spite of this modification. It might, therefore, be worthwhile to further explore if the multiplier adjustment phase can be suitably modified to solve this revised problem.

CHAPTER 6

CONCLUSIONS AND DIRECTIONS FOR FURTHER RESEARCH

The Network Design problem is of considerable theoretical and practical interest. It generalizes several well-known and extensively studied optimization problems and serves as a natural model for numerous planning problems that arise in government and industry. In addition, because of its special structure, Network Design is an attractive problem context for developing specially tailored theoretical and algorithmic results.

In this thesis, we have studied a wide range of issues that arise in the context of the Network Design problem. Our efforts were aimed at understanding the structure and properties of this problem and exploiting these properties to develop computationally attractive solution procedures. The thesis itself was, therefore, organized around these two objectives, with Chapters 2 and 3 dealing with the analytical and modeling issues and Chapters 4 and 5 emphasizing algorithmic developments and computational testing. Since the general Network Design problem is NP-hard, we focused on generating good lower and upper bounds for the problem, rather than finding optimal solutions. The linear programming relaxation plays a key role in determining the quality of the lower bounds for large-scale mixed-integer programs such as the NDP. We, therefore, began by studying the characteristics of this relaxation in the context of the NDP. In Chapter 2, we characterized the fractional extreme points for the LP

relaxation of the Path-flow formulation. We used this characterization to construct two classes of valid inequalities that eliminate some of the fractional LP extreme points. In the process, we were able to construct a systematic procedure for identifying such valid, violated inequalities, given any fractional extreme point. In contrast, in Chapter 3 we cast the NDP as a Set packing problem, which permitted us to specialize several cuts for the Set packing polytope that have been discussed in the literature. The emphasis in this chapter was on describing the general form of several classes of valid inequalities, rather than on devising procedures for identifying such cuts. We were also able to relate three seemingly diverse cut-generation techniques based, respectively, on the characterization of fractional extreme points, on the principle of constraint aggregation and coefficient reduction, and on the Set packing equivalent of the NDP. Assuming certain special structures in the given problem, we were also able to translate cuts for the Path-flow formulation into valid inequalities for the Arc-flow formulation. We did not investigate the tightness of these inequalities, for instance by establishing whether some of the cuts constitute facets of the UNDP polytope (even though most of the inequalities that we discussed in Chapter 3 are known to be facets of the corresponding Set packing polytope). Throughout Chapters 2 and 3, the Plant Location problem served as the prototype, and indeed most of our results are generalizations of results that have been derived for this problem. We found the Path-flow formulation to be analytically more tractable than the Arc-flow formulation; however, from an algorithmic point of view the latter formulation is more useful. Chapter 4 was devoted almost entirely to developing a general dual ascent framework and illustrating various implementation strategies for finding good

optimization-based lower and upper bounds on the optimal value of the NDP. Again, some of these techniques could be interpreted as logical extensions of similar methods used for solving the Plant Location problem. In this chapter, we emphasized different ways to exploit the special structure of the NDP. The limited computational results substantiate our belief that a combination of optimization-based upper bounding techniques, efficient lower bounding procedures, and related problem reduction methods is, perhaps, the best and most viable algorithmic strategy for solving large, structured problems like the NDP. In Chapter 5, we focused on a specific practical application of the NDP, namely, the Piecewise Linear Concave-cost, Multicommodity Network Flow Problem. Again, our approach was to construct a composite procedure that integrates several components in order to identify near-optimal solutions. Using this procedure, we were able to obtain relatively tight lower and upper bounds for fairly large problems. Our results also reinforced the need to devise specialized algorithms for problems with known special structures.

Directions for further research:

Further research might pursue two main lines of inquiry: to analytically study the structure and properties of the Network Design problem, and to develop and test solution procedures. We will indicate some specific topics that might be worth pursuing in each of these two areas.

While we have dealt only with properties of the Path-flow formulation of the UNDP in this thesis, understanding the structure of the integer

polytope of the Arc-flow formulation would be valuable from a computational point of view. Our LP extreme point characterization, however, does not seem to extend directly to the Arc-flow formulation; alternative approaches might, therefore, be necessary for this purpose. Consider, for instance, the following approach. We know that, since the cost coefficients are all non-negative, the Arc-flow formulation [UNDAF] of the UNDP has an optimal solution in which $y_{ij} = \text{Max}_k\ x^k_{ij}$ for all arcs (i,j). When we use this expression to subsitute for y_{ij} in the formulation [UNDAF], the LP relaxation becomes

$$\text{Minimize} \quad \sum_k \sum_{(i,j)} c^k_{ij} x^k_{ij} + \sum_{(i,j)} F_{ij}(\text{Max}_k\ x^k_{ij})$$

subject to

$$Ax^k = b^k \qquad \text{for all } k \epsilon K$$

$$x^k_{ij} \geq 0 \qquad \text{for all } k \epsilon K,\ (i,j) \epsilon A,$$

where A is the node-arc incidence matrix of the given graph, the vector b^k consists of a +1 in row O(k), a -1 in row D(k), and zeroes elsewhere, and x^k is the flow vector corresponding to commodity k. The objective function is no longer linear in the decision variables; however, as is easily shown, it is piecewise linear and convex. Notice that the set of constraints

$$Ax^k = b^k$$

defines a network flow polyhedron which, therefore, has all integral extreme points. Hence, if the LP optimum is fractional, it must necessarily not be an extreme point of this constraint set. Thus, if we can identify conditions under which the optimum solution to the piecewise linear, convex cost minimization problem (defined over a polyhedral set)

does not occur at an extreme point of the polyhedral set, then we can obtain a necessary condition for the optimal LP solution of [UNDAF] to be fractional.

This approach illustrates another issue that we have not addressed in this thesis, namely, conditions on the objective function coefficients for the LP optimum to be fractional. All our analysis of Chapters 2 and 3 has concentrated on the structure of the LP feasible region. However, it would be of greater interest to characterize circumstances under which the optimal LP solution occurs at a fractional extreme point. An approach that takes into account the relationship between the objective function and the feasible region would, therefore, be attractive. Finally, characterizing LP extreme points for the Capacitated Network Design problem is another useful area for investigation.

Another topic of interest would be to further study valid inequalities that eliminate some of the fractional extreme points. Apart from constructing more families of cuts for the Arc-flow formulation, we might explore the following issues in greater depth.

(1) Constraint Identification procedures:

In Chapter 3, we were mainly concerned with describing several classes of valid inequalities. For algorithmic development, however, procedures to systematically identify those inequalities that are violated by the current solution are necessary. We indicated a few Constraint Identification methods of this type in Chapter 2. Other approaches, such as those discussed by Crowder et al. [1983] and Martin and Schrage [1982], might involve solving a subsidiary optimization problem in order to determine such violated inequalities.

(2) Cuts for the CNDP:

Except for the Cutset inequalities that we outlined at the end of Chapter 3, our discussions were restricted to cuts for the

Uncapacitated Network Design problem. The polytope corresponding to the capacitated case, however, seems to have a very different structure, and many of the inequalities for the UNDP might not be applicable to the capacitated case if the flow variables are not restricted to be integer. Recall that, in the Constraint aggregation and Coefficient Reduction approach for instance, we assumed that all variables must have integer values in the optimal solution. We were, therefore, able to round down the fractional right hand side constants in the aggregated constraints, thus generating inequalities that were not redundant in the LP relaxation. Similarly, all the Set packing-based inequalities assume that the variables are all binary. Hence, for the capacitated problem in which the flows are not required to be integral, approaches similar to the one adopted by Van Roy and Wolsey [1984] might be necessary.

(3) Facial inequalities for the NDP:

Even for the Path-flow formulation, we were not able to establish that the facets of the corresponding Set packing polytope constitute facets of the original UNDP polytope. In contrast, Cornuejols and Thizy [1982] have proved, for the Plant Location problem, that almost all the known Set packing facets are also facets of the Plant Location polytope. The structure of the UNDP is substantially more complex than that of the Plant Location problem, and a direct extension of Cornuejols and Thizy's result does not seem possible. Establishing whether or not any particular inequality is a facet, however, is an important means for gauging the tightness of the inequality, and is worth pursuing in the context of the NDP.

(4) Inequalities for special cases:

Another area for future research is the identification of valid inequalities, preferably facets of the integer polytope, for special cases of the NDP, such as problems with a single source, problems with zero flow costs (for example, the Steiner Network problem), and problems with a 'complete' demand structure (i.e., demand from every node to every other node). Such problems arise frequently in practice, and insights into the facial structure of their feasible regions would improve algorithmic performance considerably.

While investigating valid inequalities, we should pay particular attention to the ways in which these inequalities affect the solution procedures for the problem. For instance, we found, in Chapter 4, that the dual ascent algorithm can handle additional inequalities that constrain the design variables alone (such as the Cutset inequalities) more easily than other types of cuts.

Finally, theoretical results about worst-case performance of both the upper and lower bounding procedures are of considerable interest and value. We have not attempted to derive such bounds in this thesis. The work of Cornuejols et al. [1977b] on worst-case bounds for a Lagrangian-based heuristic for the Plant Location problem, and the results derived by Wong [1980] for special cases of the Network Design problem might serve as the basis for additional work in this area.

Let us now examine some algorithmic and computational issues that remain to be resolved. First, we have tested only one version of the dual ascent algorithm, and only for a few Uncapacitated Network Design problems. To establish the usefulness of this method, we must

(1) perform more extensive testing on a wider range of problems, both capacitated and uncapacitated. By altering the method to handle undirected networks, we can compare the results obtained using this method with Magnanti et al.'s [1984] computational experience for uncapacitated, undirected Network Design problems;

(2) embed the composite algorithm in a branch-and-bound or enumeration procedure in order to solve the problems to optimality. We will then be able to gauge more accurately the quality of the heuristic solutions provided by the dual-based heuristic procedure. Furthermore, such a test would enable us to assess the usefulness and viability of the using the dual ascent procedure as a subroutine in situations that require optimal solutions; and,

(3) use the dual ascent procedure or variants/specializations of it to solve special cases of the Network Design problem. In particular, the performance of this algorithm for solving the Minimum-cost Multicommodity flow problem and the Network Design problem with a single source would be of considerable practical interest. The dual ascent technique can then be compared with other algorithms for the Multicommodity flow problem, for instance, that are currently being used.

In order for the computational comparisons to be meaningful, greater emphasis must be placed on using computationally efficient data structures, shortest path subroutines, and updating techniques.

In Chapter 4, we outlined a wide variety of strategies that can be adopted within the same basic dual ascent framework. In the absence of any theoretical results establishing the superiority of one method over another, it is necessary to evaluate the different implementations empirically and to draw some inferences regarding the circumstances under which one algorithm would outperform the others. Also, we mentioned in passing how the dual ascent technique can be altered to handle arbitrary side constraints. It might be worthwhile to devise a systematic procedure for handling such constraints, perhaps along the lines adopted by Guignard and Spielberg [1979] for the Plant Location problem. With such an enhancement, tighter lower bounds can be generated by iteratively strengthening the formulation with valid, violated inequalities. Also, the idea (that was briefly discussed in Chapter 5) of using the problem reduction procedure iteratively in order to find Dual or Lagrangian-based heuristic solutions merits further study and testing. Finally, extensions of the Concave-cost Network flow algorithm of Chapter 5 to handle problems with flow gains and losses at intermediate nodes would be of considerable practical importance, especially for production and transportation planning.

In this thesis, we have dealt exclusively with problems having a single, linear objective. However, models with nonlinear and multiple objective functions of the design and flow variables frequently represent practical situations more realistically. And yet, such problems are also highly intractable. Using optimization-based local heuristic methods seems to be the most viable problem-solving approach in most of these instances. Except for some very special cases, Multiobjective Network Design has not

been dealt with adequately in the literature, and seems to be a largely unexplored area with a vast potential for practical applications.

APPENDIX A

A GENERAL CUT-GENERATION PROCEDURE FOR THE UNCAPACITATED NETWORK DESIGN PROBLEM USING THE AUXILIARY GRAPH

In this Appendix, we extend the cut-generation procedure described in Chapter 2, to the case when all elements of the matrix R are either 0 or 1. Recall that, for any given LP extreme point solution (f,y) of the Path-flow formulation [UNDPF], the corresponding Auxiliary graph G_a contains one vertex, denoted as [p(k)], for every FLOW CARRYING path of the original graph; two vertices [p(k)] and [p'(k')] are adjacent in G_a if and only if the corresponding paths p(k) and p'(k') are LINKED with respect to the given solution, (i.e., if and only if these two paths contain a common arc (i,j) with $y_{ij} = f_{p(k)} = f_{p'(k')}$; in this case, we say arc (i,j) is tight for both the paths.). The element r_{km} in the k^{th} row and m^{th} column of the matrix R corresponding to the given solution (f,y) denotes the number of vertices of the Auxiliary graph G_a corresponding to commodity k that belong to the m^{th} component of G_a. In Chapter 2, we showed how a valid, violated inequality can be generated when at least one element r_{km} of the matrix R is greater than 1. The procedure involved finding a path in G_a between two vertices corresponding to the same commodity; the assumption about at least one element r_{km} being greater than 1 ensured that G_a contains at least one such path. The method that we develop in this section is similar. It identifies, in an Augmented graph (derived from the Auxiliary graph G_a) a cycle (instead of a path) containing two adjacent vertices corresponding to the same commodity. The violated cut is then formulated in terms of the flow variables corresponding to the vertices of this cycle

and some related design variables. We now describe the details of this general method.

In order to devise a procedure that generates a cut even when all the elements r_{km} of the matrix R are either 0 or 1, we define the AUGMENTED GRAPH G_a' as follows. This graph contains all the vertices and edges of the Auxiliary graph G_a for the given solution. In addition, it contains edges $\{[p(k)],[p'(k)]\}$ connecting every pair of vertices $[p(k)]$ and $[p'(k)]$ of G_a that correspond to the same commodity k. We refer to these additional edges of G_a' as SPECIAL edges. A special edge $\{[p(k)],[p'(k)]\}$ is said to be INCIDENT TO COMPONENT S_m if the vertex $[p'(k)]$ belongs to the m^{th} component S_m of G_a. We will assign a 'length' of v_m, the FLOW ASSOCIATED with component S_m, to each special edge of G_a' that is incident to component S_m. (Recall that, the flow associated with component S_m is the common flow value on all paths whose vertices belong to S_m.)

In order to identify a violated, valid inequality with respect to the given solution (f,y), we must find a CYCLE in G_a' that satisfies the following properties:

(1) the number of special edges, say r, in the cycle is odd,

(2) the total length of the special edges in the cycle is greater than (r-1)/2,

(3) special edges are not adjacent to each other in the cycle,

(4) no arc of the original graph is associated with more than one non-special edge of the cycle; (Recall that arc (i,j) is said to be associated with edge $\{[p(k)],[p'(k')]\}$ of G_a if it is tight for both the paths p(k) and p'(k').), and

(5) vertices corresponding to the same commodity, say $[p(k)]$ and $[p'(k)]$, are not adjacent to each other in the cycle unless the special edge $\{[p(k)],[p'(k)]\}$ also belongs to the cycle.

Thus, the cycle of interest in G_a' has the following structure:

$$[p(k)] - [p(1)] - [p(2)] - \ldots\ldots - [p(q_1)] = [p'(q_1)] - \ldots\ldots$$
$$\ldots\ldots [p(q_2)] = [p'(q_2)] \ldots = \ldots\ldots [p(q_r)] - [p'(k)] = [p(k)],$$

where, for convenience, we have indexed the commodities corresponding to the intermediate vertices from 1 to q_r, and '=' is a special edge, while '-' denotes a non-special edge of the Augmented graph G_a'.

It is easy to show that this procedure essentially traces an odd hole H of the Intersection graph G_I corresponding to the Set packing equivalent of the Path-flow formulation [UNDSPF] (see Chapter 3). While the simpler cut-generation procedure described in Chapter 2 identifies a 1-hole by tracing a path between two vertices of the Auxiliary graph, this method identifies general odd holes; each special edge of the cycle corresponds to an edge of the odd hole connecting two adjacent f-nodes [p(k)] and [p'(k)] (of the Intersection graph). The corresponding set K" (the set of all commodities with two corresponding f-nodes in the odd hole H) is, therefore,

$$K" = \{k, q_1, q_2, \ldots\ldots q_r\},$$

which is odd, since by assumption r is odd. (For 1-holes, r = 1; thus the simpler construction of Chapter 2 can equivalently be regarded as a procedure that generates a cycle satisfying the above assumptions and the additional restriction that the cycle contain exactly one special edge.)

Having identified a cycle of G_a' that satisfies the assumptions listed earlier, we can formulate the following valid inequality:

$$f_{p(k)} + f_{p'(k)} + \sum_{t=1}^{q_r} f_{p(t)} + \sum_{s=1}^{r-1} f_{p'(q_s)} + \bar{y}_{i0j0} + \sum_{t=1}^{q_r} \bar{y}_{itjt} \leq q_r + 1 + (r-1)/2$$

where (i0,j0) is the arc of G associated with edge $\{[p(k)],[p(1)]\}$ of G_a,

(it,jt) is associated with edge $\{[p(t)],[p(t+1)]\}$, for $1 \leq t \leq q_r$, and $p = q_1, q_2, \ldots, q_{(r-1)}$,

(it,jt) is associated with edge $\{[p'(q_s)],[p(q_s+1)]\}$, for $s = 1,2,\ldots r$, and $t = q_s$,

arc (iq_r, jq_r) is associated with edge $\{[p(q_r)],[p'(k)]\}$, and

$\bar{y}_{ij} = 1 - y_{ij}$.

This inequality is, in fact, identical to the Set packing inequality induced by the corresponding odd hole of the Intersection graph. (Refer Chapter 3.); hence, it is valid for [UNDPF]. Using the fact that the sum of the lengths of the special edges is greater than (r-1)/2 (and hence the sum of the flows v_m associated with the components S_m that the cycle spans is greater than (r-1)/2), we can easily show that the given solution violates this inequality. (Recall that, for any vertex [p(k)] belonging to component and any arc (i,j) of the original graph that is associated with one of the edges of component S_m,

$$f_{p(k)} = y_{ij} = v_m.)$$

Thus, this cut-generation procedure generalizes our earlier method and is essentially a method for identifying 'appropriate' odd holes (in the sense that the current solution violates the corresponding odd hole inequalities) of the Intersection graph. Finally note that, because of the

various conditions that must be satisfied, this procedure is not guaranteed to generate a violated inequality for any given extreme point.

APPENDIX B

DETAILED COMPUTATIONAL RESULTS FOR GENERAL CONCAVE-COST NETWORK FLOW TEST PROBLEMS

Upper/Lower Bounds and Problem Reduction

Problem Name	SITR	IUB % BUB	ILB % BUB	LBIA % BUB	FLB % BUB	% Free Var Init	% Free Var Final
GEN11	100	100.6	54.7	78.3	99.7	80.4	6.1
GEN12	100	100.0	55.5	77.1	100.0	80.0	5.5
GEN13	45	100.0	58.3	71.5	100.0	79.8	0.0
GEN14	64	100.0	51.1	72.2	100.0	83.8	1.9
GEN15	55	100.6	52.5	74.8	100.0	81.5	1.1
GEN1	72.8	100.2	54.4	74.8	99.9	81.1	2.9
GEN21	200	103.0	44.7	66.5	98.3	87.3	28.4
GEN22	197	102.7	52.2	79.9	98.8	87.9	24.6
GEN23	83	109.0	53.2	74.0	100.0	86.3	0.0
GEN24	200	100.6	44.9	71.6	98.5	86.7	27.9
GEN25	97	100.0	47.3	71.3	97.9	88.1	35.9
GEN2	155.4	103.1	48.5	72.7	98.7	87.3	23.4
GEN31	100	102.6	45.3	72.3	99.2	90.3	19.6
GEN32	200	102.8	44.7	69.4	98.8	90.8	30.1
GEN33	100	104.9	47.7	69.8	98.6	90.0	32.8
GEN34	195	103.5	44.9	69.4	98.3	89.8	46.0
GEN35	199	105.2	44.4	70.9	97.0	90.2	67.2
GEN3	158.8	103.8	45.4	70.3	98.4	90.2	39.1

Problem Name	SITR	IUB % BUB	ILB % BUB	LBIA % BUB	FLB % BUB	% Free Var Init	% Free Var Final
GEN41	299	103.2	44.6	65.5	96.2	93.7	91.8
GEN42	299	107.4	45.6	66.4	94.6	93.4	93.4
GEN43	299	102.4	43.7	66.0	96.0	93.5	93.0
GEN44	300	102.7	43.2	67.9	96.2	93.4	86.2
GEN45	199	103.9	46.0	66.4	98.0	93.4	70.7
GEN4	279.2	103.9	44.6	66.4	96.2	93.5	87.0
GEN51	200	100.9	40.4	63.1	98.4	94.9	79.0
GEN52	300	100.7	42.5	67.6	98.7	94.9	62.0
GEN53	300	100.7	40.5	63.5	98.6	95.1	71.0
GEN54	300	100.1	39.6	64.1	98.6	95.2	72.8
GEN55	298	100.5	41.7	61.2	98.0	95.0	84.4
GEN5	279.6	100.6	40.9	63.9	98.5	95.0	73.8

Problem naming convention:
Problem GENab is the b^{th} instance of General network problem size a.
For problem size a, average over 5 instances shown under GENa.

Legend:
SITR = No. of subgradient iterations
ILB = Initial Lower Bound (after Step 1)
LBIA = Lower Bound after Intial Ascent (Step 2)
FLB = Final Lower Bound
IUB = Initial Upper Bound
BUB = Best Upper Bound
% Free variables : expressed as a percentage of total number of possible flow variables

Appendix A (continued)

Computation times

Problem Name	CPU times (secs on PR1ME 850) for Init Ascent	Subseq Ascent	Subgrad Opt	Prob Reducn	Heur*
GEN11	3.5	0.6	23.4	2.2	0.6
GEN12	2.5	3.3	23.1	2.2	0.5
GEN13	2.2	0.3	21.6	1.8	0.4
GEN14	3.0	0.1	15.3	1.8	0.3
GEN15	4.4	0.6	23.9	1.9	0.4
GEN1	3.1	1.0	21.5	2.0	0.4
GEN21	24.0	12.0	153.1	106.2	3.5
GEN22	37.7	11.2	196.2	118.4	4.0
GEN23	21.8	0.3	86.0	12.3	1.6
GEN24	26.1	11.2	167.6	82.3	2.7
GEN25	20.6	4.0	91.6	41.6	1.9
GEN2	26.1	7.7	138.9	72.2	2.8
GEN31	148.5	48.8	299.2	168.8	15.6
GEN32	144.2	75.6	583.9	459.2	14.2
GEN33	138.2	35.2	293.1	224.5	13.3
GEN34	139.0	40.1	521.8	525.0	10.1
GEN35	133.8	109.3	521.8	456.8	17.9
GEN3	140.7	61.8	444.0	366.9	14.2

Problem Name	CPU times (secs on PRIME 850) for Init Ascent	Subseq Ascent	Subgrad Opt	Prob Reducn	Heur*
GEN41	1032.9	982.0	3559.5	5555.1	118.6
GEN42	1109.6	1832.5	3669.0	4681.6	209.2
GEN43	1014.5	1271.4	3458.8	5361.3	135.9
GEN44	1005.7	805.3	3481.8	6404.4	134.3
GEN45	967.7	564.1	2327.8	2876.6	100.4
GEN4	1026.0	1091.0	3299.4	4975.8	139.7
GEN51	1810.5	753.1	2932.7	6040.7	127.8
GEN52	1641.4	551.3	4282.9	10082.6	97.8
GEN53	1853.4	1239.3	4294.1	10791.3	123.8
GEN54	1945.9	1172.8	4366.0	8582.0	102.3
GEN55	1536.1	1576.8	4294.4	9319.2	99.3
GEN5	1757.5	1058.7	4034.0	8963.2	110.2

* CPU time for Heuristic phase expressed in terms of secs per successful application of local improvement procedure.

APPENDIX C

DETAILED COMPUTATIONAL RESULTS FOR THREE-LAYER CONCAVE-COST NETWORK FLOW TEST PROBLEMS

Upper/Lower Bounds and Problem Reduction

Problem Name	SITR	IUB % BUB	ILB % BUB	LBIA % BUB	FLB % BUB	% Free Var Init	% Free Var Final
LTL11	70	106.9	95.4	98.5	100.0	31.3	0.0
LTL12	44	100.0	94.2	96.9	100.0	26.5	2.6
LTL13	99	105.0	91.5	98.3	100.0	31.8	0.0
LTL14	99	103.0	93.4	96.1	100.0	31.7	3.8
LTL15	99	103.5	88.4	93.8	98.8	30.6	23.3
LTL1	71.2	103.7	92.6	96.7	99.8	30.4	5.9
LTL21	48	109.1	91.0	95.3	100.0	28.3	0.0
LTL22	75	124.8	94.5	95.6	100.0	29.0	0.0
LTL23	46	114.9	90.2	93.1	100.0	29.5	0.0
LTL24	98	101.8	85.9	91.3	100.0	28.9	0.0
LTL25	76	114.9	89.0	93.8	100.0	29.2	0.7
LTL2	68.6	113.1	90.1	93.8	100.0	29.0	0.1
LTL31	156	101.0	88.8	96.2	100.0	27.4	0.0
LTL32	84	113.7	88.9	93.7	100.0	26.0	0.0
LTL33	68	105.6	89.4	92.2	100.0	27.2	0.0
LTL34	98	113.2	84.4	90.1	100.0	26.6	0.0
LTL35	96	104.6	85.5	90.6	98.1	26.9	25.6
LTL3	100.4	107.6	87.4	92.6	99.6	26.8	5.1

Problem Name	SITR	IUB % BUB	ILB % BUB	LBIA % BUB	FLB % BUB	% Free Var Init	% Free Var Final
LTL41	196	100.0	86.2	88.7	97.5	21.9	21.2
LTL42	57	105.2	89.4	94.0	100.0	22.3	0.0
LTL43	100	103.3	87.2	93.3	100.0	21.9	0.0
LTL44	95	100.0	89.7	93.7	98.3	21.7	19.3
LTL45	73	112.5	91.2	97.2	100.0	21.7	0.0
LTL4	134.4	104.2	88.7	93.4	99.1	21.9	8.1
LTL51	177	115.3	90.4	92.8	100.0	13.5	0.0
LTL52	100	106.1	90.6	93.8	100.0	13.6	0.0
LTL53	197	103.3	83.8	91.5	97.6	13.4	13.4
LTL54	98	101.1	90.6	94.3	100.0	12.7	0.0
LTL55	100	102.2	89.6	95.1	99.7	13.9	1.3
LTL5	134.4	105.4	89.0	93.5	99.5	13.4	2.9

Problem naming convention:
Problem LTLab is the b^{th} instance of Three-layer network problem size a.
For problem size a, average over 5 instances shown under LTLa.

Legend:
SITR = No. of subgradient iterations
ILB = Initial Lower Bound (after Step 1)
LBIA = Lower Bound after Intial Ascent (Step 2)
FLB = Final Lower Bound
IUB = Initial Upper Bound
BUB = Best Upper Bound
% Free variables : expressed as a percentage of total number of possible flow variables

Appendix C (continued)

Computation times

Problem Name	CPU times (secs on PRIME 850) for Init Ascent	Subseq Ascent	Subgrad Opt	Prob Reducn	Heur*
LTL11	1.2	0.3	10.8	0.8	0.5
LTL12	1.0	0.4	7.5	1.0	0.4
LTL13	1.3	0.1	7.4	0.9	0.6
LTL14	1.2	0.2	15.9	1.0	0.8
LTL15	1.1	0.5	15.3	1.6	0.4
LTL1	1.2	0.3	11.4	1.1	0.5
LTL21	4.8	0.3	33.2	2.7	1.9
LTL22	4.6	0.3	52.3	2.6	2.3
LTL23	5.8	0.3	32.6	2.6	4.0
LTL24	5.2	1.2	68.1	3.6	2.4
LTL25	4.4	0.3	57.4	3.1	3.7
LTL2	5.0	0.5	48.7	2.9	2.8
LTL31	14.5	4.4	197.3	20.5	5.3
LTL32	12.2	0.4	116.3	5.2	4.7
LTL33	11.2	3.5	114.7	6.1	8.6
LTL34	14.0	1.4	146.0	6.0	15.1
LTL35	11.5	4.6	146.1	49.6	11.1
LTL3	12.7	2.9	144.1	17.5	9.0

Problem Name	CPU times (secs on PRIME 850) for Init Ascent	Subseq Ascent	Subgrad Opt	Prob Reducn	Heur*
LTL41	29.0	103.8	745.9	431.7	28.3
LTL42	30.9	1.0	224.0	13.6	97.1
LTL43	34.8	4.6	380.8	21.0	60.5
LTL44	29.8	13.2	358.8	306.0	29.2
LTL45	46.6	2.4	289.5	16.6	46.9
LTL4	34.2	25.0	399.7	157.8	52.4
LTL51	51.2	15.6	778.8	107.3	33.6
LTL52	34.0	140.8	425.1	21.6	105.2
LTL53	52.8	49.2	852.5	271.1	83.7
LTL54	34.8	2.8	431.7	15.8	55.5
LTL55	48.9	3.5	425.5	98.9	82.9
LTL5	43.3	42.4	582.7	102.9	72.2

* CPU time for Heuristic phase expressed in terms of secs per successful application of local improvement procedure.

REFERENCES

REFERENCES

A1. Akinc, V., and B. Khumawala, "An efficient branch-and-bound algorithm for the capacitated warehouse location problem", Management Science, Vol. 23, pp. 585-594. [1977]

A2. Assad, A., "Multicommodity network flows - A survey", Networks, Vol. 8, pp. 37-91. [1978]

B1. Balakrishnan, A., "Formulations and algorithms for the Steiner network problem", Unpublished manuscript, Sloan School of Management, M.I.T. [1982]

B2. Balas, E., "Facets of the Knapsack polytope", Mathematical Programming, Vol. 8, pp. 146-164. [1975]

B3. Balas, E., "Disjunctive programming and a hierarchy of relaxations for discrete optimization problems", Management Science Research Report No. 492, Graduate School of Industrial Administration, Carnegie-Mellon University. [1983]

B4. Balas, E., "On the facial structure of scheduling polyhedra", Management Science Research Report No. 496(R), Graduate School of Industrial Administration, Carnegie-Mellon University. [1984]

B5. Balas, E., and M. W. Padberg, "Set partitioning: A survey", SIAM Review, Vol. 18, pp. 710-760. [1976]

B6. Balas, E., and E. Zemel, "Critical cutsets of graphs and canonical facets of Set-packing polytopes", Mathematics of Operations Research, Vol. 2, No. 1, pp. 15-19. [1977]

B7. Balinski, M. L., "Fixed-cost transportation problems", Naval Logistics Research Quarterly, Vol. 8, pp. 41-54. [1961]

B8. Barany, I., T. J. Van Roy and L. A. Wolsey, "Uncapacitated lot-sizing : The convex hull of solutions", CORE Discussion paper No. 8314, Universite Catholique de Louvain, Belgium. [1983]

B9. Barany, I., T. J. Van Roy and L. A. Wolsey, "Strong formulations for multi-item capacitated lot sizing", Management Science, Vol. 30, No. 10, pp. 1255-1261. [1984]

B10. Barr, R. S., F. Glover and D. Klingman, "A new optimization method for large-scale fixed charge transportation problems", Operations Research, Vol. 29, No.3, pp. 448-463. [1981]

B11. Bazaraa, M. S., and J. J. Goode, "A survey of various tactics for generating Lagrangian multipliers in the context of Lagrangian

duality", European Journal of Operational Research, Vol. 3, pp. 322-338. [1979]

B12. Beasley, J. E., "An algorithm for Set covering problems", Working paper, Department of Management Science, Imperial College, London [1983]

B13. Benders, J. F., "Partitioning procedures for solving mixed variable programming problems", Numerische Mathematik, Vol. 4, pp. 238-252. [1962]

B14. Billheimer, J., and P. Gray, "Network design with fixed and variable cost elements", Transportation Science, Vol. 7, pp. 49-74. [1973]

B15. Boesch, F. T. (ed.), Large scale networks : Theory and design, IEEE Press, New York. [1976]

B16. Boffey, T. B. and A. I. Hinxman, "Solving the optimal network problem", European Journal of Operational Research, Vol. 3, pp. 386-393. [1979]

B17. Boorstyn, R. R., and H. Frank, "Large-scale network topological optimization", IEEE Transactions on Communications, Vol. COM-25, No.1, pp. 29-47. [1977]

B18. Boyce, D. E. (ed.), Transportation Research B, Vol. 13B, No.1. [1979]

B19. Boyce, D. E., A. Farhi and R. Weischedel, "Optimal network problem : A branch-and-bound algorithm", Environment and Planning, Vol. 5, pp. 519-533. [1973]

B20. Bradley, S. P., A. C. Hax, and T. L. Magnanti, Applied Mathematical Programming, Addison-Wesley, Reading, Massachusetts. [1977]

C1. Camerini, P. M., L. Fratta and F. Maffioli, "On improving relaxation methods by modified gradient techniques", Mathematical Programming Study 3, pp. 26-34. [1975]

C2. Chandy, K. M., and T. Lo, "The capacitated minimum spanning tree", Networks, Vol. 2, pp. 173-182. [1973]

C3. Cho, D. C., E. L. Johnson, M. W. Padberg and M. R. Rao, "On the uncapacitated plant location problem I: Valid inequalities and facets", Mathematics of Operations Research, Vol. 8, pp. 579-589. [1983a]

C4. Cho, D. C., M. W. Padberg and M. R. Rao, "On the uncapacitated plant location problem II: Facets and lifting theorems", Mathematics of Operations Research, Vol. 8, pp. 590-612. [1983b]

C5. Chu, W. W. (ed.), Advances in Computer Communications, Artech House, Dedham, Mass. [1977]

C6. Chvatal, V., "Edmonds polytopes and a hierarchy of combinatorial problems", Discrete Mathematics, Vol. 4, pp. 305-337. [1973]

C7. Chvatal, V., "On certain polytopes associated with graphs", Journal of Combinatorial Theory B, Vol. 18, pp. 138-154. [1975]

C8. Chvatal, V. and P. L. Hammer, "Aggregation of inequalities in integer programming", Annals of Discrete Mathematics, Vol. 1, pp. 145-162. [1977]

C9. Cooper, L., and C. Drebes, "An approximate solution method for the fixed charge problem", Naval Logistics Research Quarterly, Vol. 14, pp. 101-113. [1967]

C10. Cornuejols, G., M. L. Fisher and G. L. Nemhauser, "Location of bank accounts to optimize float", Management Science, Vol. 23, pp. 789-810. [1977a]

C11. Cornuejols, G., M. L. Fisher and G. L. Nemhauser, "On the uncapacitated location problem", Annals of Discrete Mathematics, Vol. 1, pp. 163-177. [1977b]

C12. Cornuejols, G., and J. M. Thizy, "Some facets of the simple plant location polytope", Mathematical Programming, Vol. 23, pp. 50-74. [1982]

C13. Crowder, H., "Computational improvements for subgradient optimization", Symposia Mathematica, Vol. 19, pp. 357-372. [1976]

C14. Crowder, H., E. L. Johnson and M. W. Padberg, "Solving large-scale zero-one linear programming problems", Operations Research, Vol. 31, pp. 803-834. [1983]

C15. Crowder, H., and M. W. Padberg, "Large scale symmetric traveling salesman problems", Management Science, Vol. 26, pp. 495-509. [1980]

D1. Davis, P. S., and T. L. Ray, "A branch-and-bound algorithm for the capacitated facilities location problem", Naval Logistics Research Quarterly, Vol. 16, No.3, pp. 331-344. [1969]

D2. Denzler, D. R., "An approximative algorithm for the fixed charge problem", Naval Logistics Research Quarterly, Vol. 16, pp. 411-416. [1969]

D3. Dijkstra, E. W., "A note on two problems in connexion with graphs", Numerische Mathematik, Vol. 1, pp. 269-271. [1959]

D4. Dionne, R., and M. Florian, "Exact and approximate algorithms for optimal network design", Networks, Vol. 9, pp. 37-59. [1979]

E1. Efroymson, M. A., and T. L. Ray, "A branch-and-bound algorithm for plant location", Operations Research, Vol. 14, pp. 361-368. [1966]

E2. Ellwein, L.B., and P. Gray, "Solving fixed charge location-allocation problems with capacity and configuration constraints", AIIE Transactions, Vol. 3, No.4, pp. 290-297. [1977]

E3. Elmaghraby, S. E., Some Network models in management science, Springer-Verlag, New York. [1970]

E4. Elmaghraby, S. E., Activity networks: Project planning and control by network models, John Wiley and Sons, New York. [1977]

E5. Erickson, R. E., C. L. Monma and A. P. Veinott, "Minimum concave-cost network flows", Working paper. [1981]

E6. Erlenkotter, D., "A dual based procedure for uncapacitated facility location", Operations Research, Vol. 26, pp. 992-1009. [1978]

E7. Essau, L. R., and K. C. Williams, "On teleprocessing system design", IBM System Journal, Vol. 5, No.3, pp. 142-147. [1976]

F1. Fisher, M. L., "The Lagrangian relaxation method for solving integer programming problems", Management Science, Vol. 27, pp. 1-18. [1981]

F2. Fisher, M. L., A. Greenfield, R. Jaikumar and P. Kedia, "Real-time scheduling of a bulk delivery fleet : Practical application of Lagrangian relaxation", Working paper No. 82-10-11, Deptt. of Decision Science, The Wharton School, Univ. of Pennsylvania, Philadelphia, PA. [Oct. 1982]

F3. Fisher, M. L., and D. S. Hochbaum, "Database location in computer networks", Journal of the ACM, Vol. 27, pp. 718-735. [1980]

F4. Fisher, M. L., R. Jaikumar and L. Van Wassenhove, "A multiplier adjustment method for the generalized assignment problem", Decision Sciences Working paper, University of Pennsylvania. [1980]

F5. Florian, M., G. Guerin and G. Bushel, "The engine scheduling problem in a railway network", INFOR, Vol. 14, pp. 121-138. [1976]

F6. Florian, M., and M. Klein, "Deterministic production planning with concave costs and capacity constraints", Management Science, Vol. 18, pp. 12-20. [1971]

F7. Ford, L. R., and D. R. Fulkerson, Flows in networks, Princeton University Press, Princeton, NJ. [1962]

F8. Foulds, L. R., "A multi-commodity flow network design problem", Transportation Research B, Vol. 15B, No.4, pp. 273-283. [1981]

F9. Francis, R., and J. White, Facility layout and locations : An analytical approach, Prentice-Hall, Englewood Cliffs, NJ. [1974]

F10. Frank, H., I. T. Frisch, W. Chou and R. Van Slyke, "Optimal design of centralized computer networks", Networks, Vol. 1, No.1, pp. 43-57.

[1971]

F11. Fulkerson, D. R., "A network flow computation for project cost curves", Management Science, Vol. 7, No. 2, pp. 161-179. [1961]

G1. Gallo, G., C. Sandi and C. Sodini, "Network flow problems with concave costs", Proceedings of AIRO National meeting, Urbino, Italy. [1978]

G2. Gallo, G., and C. Sodini, "Concave cost minimization on networks", European Journal of Operational Research, Vol 3, pp 239-249. [1979]

G3. Garey, M. R., and D. S. Johnson, Computers and intractability : A guide to the theory of NP Completeness, W.H.Freeman and Company, San Francisco. [1979]

G4. Garfinkel, R. S., and G. L. Nemhauser, Integer Programming, Wiley-Interscience, NY. [1972]

G5. Gavish, B., "On obtaining the best multipliers for a Lagrangian relaxation for integer programming", Computers and Operations Research, Vol. 5, pp. 55-71. [1978]

G6. Gavish, B., "Topological design of centralized computer networks - Formulations and algorithms", Networks, Vol. 12, pp. 355-378. [1982]

G7. Gavish, B., "Formulations and algorithms for the capacitated minimal directed tree problem", Journal of ACM, Vol. 30, No.1, pp. 118-132. [1983]

G8. Geoffrion, A. M., "Lagrangian relaxation for integer programming", Mathematical Programming Study 2, pp. 82-114. [1974]

G9. Geoffrion, A. M., "How can specialized discrete and convex optimization methods be married", Annals of Discrete Mathematics, Vol. 1, pp. 205-220. [1977]

G10. Geoffrion, A. M., and G. Graves, "Multicommodity distribution systems design by Benders decomposition", Management Science, Vol. 5, pp. 822-844. [1974]

G11. Geoffrion, A. M., and R. McBride, "Lagrangian relaxation applied to facility location problems", AIIE Transactions, Vol. 10, No.1, pp. 40-46. [1978]

G12. Gerla, M., and L. Kleinrock, "On the topological design of distributed computer networks", IEEE Transactions on Communications, Vol. COM-25, No.1, pp. 48-60. [1977]

G13. Goffin, J. L., "On the convergence rates of subgradient optimization methods", Mathematical Programming, Vol. 13, pp. 329-347. [1977]

G14. Golden, B., "A problem in network interdiction", Naval Research Logistics Quarterly, Vol. 25, No. 4, pp. 711-713. [1978]

G15. Granot, D., and I. Zang, "On the min-max solution of some combinatorial optimization problems", paper presented at the TIMS/ORSA Joint National meeting, Chicago. [1983]

G16. Gray, P., "Exact solution of the fixed-charge transportation problem", Operations Research, Vol. 19, pp. 1529-1538. [1971]

G17. Grotschel, M., M. Junger and G. Reinelt, "A cutting plane algorithm for the Linear Ordering problem", Operations Research, Vol. 32, pp. 1195-1220. [1984]

G18. Grotschel, M., and M. W. Padberg, "On the symmetric traveling salesman problem I : Inequalities", Mathematical Programming, Vol. 16, pp. 265-280. [1979a]

G19. Grotschel, M., and M. W. Padberg, "On the symmetric traveling salesman problem II : Lifting theorems and facets", Mathematical Programming, Vol. 16, pp. 281-302. [1979b]

G20. Guignard, M., "Fractional vertices, cuts and facets of the simple plant location problem", Mathematical Programming, Vol. 12, pp. 150-162. [1980]

G21. Guignard, M., and K. Spielberg, "A direct dual method for the mixed plant location problem with some side constraints", Mathematical Programming, Vol. 17, pp. 198-228. [1979]

G22. Guignard, M., and K. Spielberg, "Logical reduction methods in 0-1 programming", Operations Research, Vol. 29, pp. 49-74. [1981]

H1. Hammer, P. L., E. L. Johnson and U. M. Peled, "Facets of regular 0-1 polytopes", Mathematical Programming, Vol. 8, pp. 129-206. [1975]

H2. Handler, G., and P. Mirchandani, Location on networks : Theory and algorithms, MIT Press, Cambridge, Massachusetts. [1979]

H3. Held, M., and R. Karp, "The traveling salesman problem and minimum spanning trees", Operations Research, Vol. 18, pp. 1138-1162. [1970]

H4. Held, M., and R. Karp, "The traveling salesman problem and minimum spanning trees: Part II", Mathematical Programming, Vol. 1, pp. 6-25. [1971]

H5. Held, M., P. Wolfe and H. P. Crowder, "Validation of subgradient optimization", Mathematical Programming, Vol. 6, pp. 62-88. [1974]

H6. Hirsch, W. M., and G. B. Dantzig, "The fixed charge problem", Naval Logistics Research Quarterly, Vol. 15, pp.413-424. [1968] (originally published as a RAND paper in 1954).

H7. Hoang, H. H., "A computational approach to the selection of an optimal network", Management Science, Vol. 19, No.5, pp. 488-498. [1973]

J1. Jarvis, J., R. Rardin, E. Unger, R. Moore, and C. Schimpeler, "Optimal design of regional wastewater systems : A fixed-charge network flow model", Operations Research, Vol. 26, No.4, pp. 538-550. [1978]

J2. Jeroslow, R. G., and J. Lowe, "Modelling with integer variables", Report No. 83270-OR, Institute fur Okonometrie and Operations Research, University of Bonn. [March 1983]

J3. Johnson, D.S., J. K. Lenstra and A. H. G. Rinnooy Kan, "The complexity of the network design problem", Networks, Vol. 8, pp. 279-285. [1978]

K1. Kennington, J. L., "Multicommodity flows : A state-of-the-art survey of linear models and solution techniques", Operations Research, Vol. 26, pp. 209-236. [1978]

K2. Kennington, J., and E. Unger, "A new branch-and-bound algorithm for the fixed-charge transportation problem", Management Science, Vol. 22, No.10, pp. 1116-1126. [1976]

K3. Kershenbaum, A., "Computing capacitated minimal spanning trees efficiently", Networks, Vol. 4, pp. 299-310. [1974]

K4. Knuth, D., The art of computer programming: Volume 2, Seminumerical algorithms, Addison-Wesley, Reading, Massachusetts. [1969]

K5. Kuhn, H. W. and W. J. Baumol, "An approximative algorithm for the fixed-charge transportation problem", Naval Logistics Research Quarterly, Vol. 9, pp. 1-15. [1962]

L1. Lamar, B., "Less-than-truckload freight consolidation and routing strategies: A mathematical programming procedure", Project Report No. CTS/IU-83.2, Center for Transportation Studies, MIT. [1983]

L2. Lasdon, L., Optimization theory for large systems, Macmillan Company, London. [1970]

L3. LeBlanc, L. J., "An algorithm for the discrete network design problem", Transportation Science, Vol. 9, pp. 183-199. [1975]

L4. Leung, J., T. L. Magnanti, and V. Singhal, "Point-to-point delivery systems", paper presented at the ORSA/TIMS Joint National meeting, Dallas. [1984]

L5. Los, M., and C. Lardinois, "Combinatorial programming, statistical optimization and the optimal transportation network problem", Transportation Research B, Vol. 16B, No.2, pp. 89-124. [1982]

L6. Love, S. F., "Bound production and inventory models with piecewise linear concave costs", Management Science, Vol. 20, pp. 313-318. [1973]

L7. Luss, H., "A capacity-expansion model for two facility types", Naval Research Logistics Quarterly, Vol. 26, pp. 291-303. [1979]

M1. Magnanti, T. L., "Optimization for sparse systems", in Sparse Matrix Computations, J.R.Bunch and D.J.Rose (eds.), Academic Press, New York. [1976]

M2. Magnanti, T. L., "Combinatorial optimization and vehicle fleet planning: Perspectives and prospects", Networks, Vol. 11, pp. 179-214. [1981a]

M3. Magnanti, T. L., J. Shapiro and M. Wagner, "Generalized linear programming solves the dual", Management Science, Vol. 22, pp. 1195-1203. [1976]

M4. Magnanti, T. L., and R. T. Wong, "Accelerating Benders decomposition : Algorithmic enhancement and model selection criteria", Operations Research, Vol. 29, pp. 464-484. [1981]

M5. Magnanti, T. L., and R. T. Wong, "Network design and transportation planning : Models and algorithms", Transportation Science, Vol. 18, No. 1, pp. 1-55. [1984a]

M6. Magnanti, T. L., and R. T. Wong, Work in progress. [1984b]

M7. Magnanti, T. L., R. T. Wong and P. Mireault, "Tailoring Benders decomposition for uncapacitated network design", Working paper No. OR 127-84, Operations Research Center, MIT. [1984]

M8. Malek-Zaverei, M., and I. T. Frisch, "On the fixed-cost flow problem", International Journal of Control, Vol. 16, No.5, pp. 897-902. [1972]

M9. Mandl, C., "A survey of mathematical optimization models and algorithms for designing and extending irrigation and wastewater networks", Water Resources Research, Vol. 17, No.4. [1981]

M10. Manheim, M. L., E. R. Ruiter and K. U. Bhatt, "Search and choice in transportation systems planning : Summary report", Research report R68-40, Department of Civil Engineering, M.I.T. [1968]

M11. Martin, K., and L. Schrage, "Subset coefficient reduction cutting planes for 0/1 mixed-integer programming", Working paper, Graduate School of Business, University of Chicago. [1982]

M12. McKeown, P. G., "A vertex ranking procedure for solving the linear fixed-charge problem", Operations Research, Vol. 25, pp. 1183-1191. [1975]

M13. Meyer, R. R., "A theoretical and computational comparison of equivalent mixed integer formulations", Naval Research Logistics Quarterly, Vol. 28, pp. 115-131. [1981]

M14. Moder, J. J., C. R. Phillips and E. W. Davis, Project management with CPM, PERT and Precendence diagramming, Van Nostrand Reinhold Company, New York. [1983]

M15. Murty, K. G., "Solving the fixed charge problem by ranking the extreme points", Operations Research, Vol. 16, pp. 268-279. [1968]

N1. Nemhauser, G. L., and L. E. Trotter, "Properties of vertex packing and independence systems polyhedra", Mathematical Programming, Vol. 6, pp 48-61. [1974]

N2. Nemhauser, G. L., and L. E. Trotter, "Vertex packings: Structural properties and algorithms", Mathematical Programming, Vol. 8, pp. 232-248. [1975]

P1. Padberg, M. W., "On the facial structure of set packing polyhedra", Mathematical Programming, Vol. 5, pp. 199-215. [1973]

P2. Padberg, M. W., "Covering, packing and knapsack problems", Annals of Discrete Mathematics, Vol. 4, pp. 265-287. [1979]

P3. Padberg, M. W., and S. Hong, "On the symmetric traveling salesman problem : A computational study", Mathematical Programming Study, Vol. 12, pp. 78-107. [1980]

P4. Padberg, M. W., T. Van Roy and L. Wolsey, "Valid linear inequalities for fixed charge problems", CORE Discussion paper No. 8232, Universite Catholique de Louvain, Belgium. [1982]

P5. Peterson, B. E., "A cut-flow procedure for transportation network optimization", Networks, Vol. 10, pp. 33-43. [1980]

P6. Powell, W. B., and Y. Sheffi, "The load planning problem of LTL motor carriers: Problem description and a proposed solution approach", Transportation Research, Vol. 17A, pp. 471-480. [1983]

R1. Rardin, R. L., and U. Choe, "Tighter relaxations of fixed charge network flow problems", Technical Report # J-79-18, School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, Georgia. [1979]

R2. Richardson, R., "An optimization approach to routing aircraft", Transportation Science, Vol. 10, pp. 52-71. [1976]

S1. Sa, G., "branch-and-bound and approximate solutions to the capacitated plant location problem", Operations Research, Vol. 17, pp. 1005-1016. [1969]

S2. Schwartz, M., Computer-communication network design and analysis, Prentice-Hall, Englewood Cliffs, NJ. [1977]

S3. Scott, A. J., "The optimal network problem: Some computational procedures", Transportation Research, Vol. 3, pp. 201-210. [1969]

S4. Shapiro, J. F., "A survey of Lagrangian techniques for discrete optimization", Annals of Discrete Mathematics, Vol. 5, pp. 113-138. [1979a]

S5. Shapiro, J. F., Mathematical programming: Structures and algorithms, Wiley-Interscience, New York. [1979b]

S6. Singhal, V., "Point-to-point delivery systems", S.M. thesis, Department of Electrical Engineering and Computer Science, M.I.T. [1984]

S7. Soland, R. M., "Optimal facility location with concave costs", Operations Research, Vol. 22, pp. 373-382. [1974]

S8. Steenbrink, P. A., Optimization of transport networks, John Wiley, NY. [1974]

S9. Steinberg, D. I., "The fixed charge problem", Naval Logistics Research Quarterly, Vol. 17, No.2, pp. 217-235. [1970]

S10. Swoveland, C., "A deterministic multi-period production planning model with piecewise concave production and holding-backorder costs", Management Science, Vol. 21, pp. 1007-1013. [1975]

T1. Taha, H. A., "Concave minimization over a convex polyhedron", Naval Logistics Research Quarterly, Vol. 20, pp. 533-548. [1973]

T2. Tannenbaum, A. S., Computer networks, Prentice-Hall, Englewood Cliffs, NJ. [1982]

T3. Tansel, B. C., R. L. Francis and T. J. Lowe, "Location on networks : A survey, parts I and II", Management Science, Vol. 29, pp. 482-511. [1983]

T4. Trotter, L. E., "A class of facet producing graphs for vertex packing polytopes", Discrete Mathematics, Vol. 12, pp. 373-388. [1975]

V1. Van Roy, T. J., and D. Erlenkotter, "A Dual based procedure for dynamic facility location", Management Science, Vol. 28, pp. 1091-1105. [1982]

V2. Van Roy, T. J., and L. A. Wolsey, "Valid inequalities and separation for uncapacitated fixed charge networks", CORE Discussion paper No. 8410, Universite Catholique de Louvain, Belgium. [1984]

W1. Walker, W. E., "A heuristic adjascent extreme point algorithm for the fixed charge problem", Management Science, Vol. 22, pp. 587-596. [1976]

W2. Williams, H. P., "Experiments in the formulation of integer programming problems", Mathematical Programming Study 2, Vol. 5, pp. 180-197. [1974]

W3. Wolsey, L. A., "Facets and strong valid inequalities for integer programs", Operations Research, Vol. 24, pp. 367-372. [1976]

W4. Wong, R. T., "Worst-case analysis of network design problem heuristics", SIAM Journal of Algebraic and Discrete Methods, Vol. 1,

pp 51-63. [1980]

W5. Wong, R. T., "A dual ascent approach for Steiner tree problems on a directed graph", Mathematical Programming, Vol. 28, pp. 271-287. [1984]

Y1. Yao, A. C., "An O(|E| log log |V|) algorithm for finding minimal spanning trees", Information Processing Letters, Vol. 4, No.1, pp. 21-23. [1975]

Y2. Yaged, B., "Minimum cost routing for static network models", Networks, Vol. 1, pp. 139-172. [1971]

Z1. Zadeh, N., "On building minimum cost communication networks", Networks, Vol. 3, No. 4, pp. 315-331. [1973]

Z2. Zadeh, N., "On building minimum cost communication networks over time", Networks, Vol. 4, No. 1, pp. 19-34. [1974]

Z3. Zangwill, W. I., "Minimum concave cost flows in certain networks", Management Science, Vol. 14, No. 7, pp. 429-450. [1968]

Z4. Zemel, E., "Lifting the facets of zero-one polytopes", Mathematical Programming, Vol. 15, 268-277. [1978]