Modeling and Solving Network Flow Problems with Piecewise Linear Costs, with Applications in Supply Chain Management

by

Keely L. Croxton

B.S., Northwestern University (1992)
Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Operations Research

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 1999

Signature of Author .....
Department of Electrical Engineering and Computer Science
August 5, 1999

Certified by .....
Thomas L. Magnanti
Dean of Engineering and Institute Professor, MIT
Thesis Supervisor

Accepted by .....
James B. Orlin
E. Pennell Brooks Professor of Operations Research, MIT
Co-Director of The Operations Reserach Center, MIT

Modeling and Solving Network Flow Problems with Piecewise Linear Costs, with Applications in Supply Chain Management

by

Keely L. Croxton

Submitted to the Department of Electrical Engineering and Computer Science
on August 5, 1999, in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in Operations Research

Abstract

U.S. logistics costs account for nearly 30% of every sales dollar and mathematical modeling can play an important role in analyzing supply chains and implementing efficient and effective solutions. We address some generic modeling issues that arise in supply chain optimization, as well as other application domains, and then examine a specific application context called merge-in-transit.

Many supply chain applications are naturally modeled as network flow problems with complex cost structures. We address modeling techniques for problems with non-convex piecewise linear costs, and apply these techniques to several applications in supply chain management.

The underlying models are mixed integer programs. We first show that the linear programming relaxations of three classical textbook models are equivalent, each providing the lower convex envelope of the given cost function. We then investigate the use of variable disaggregation to tighten the formulation and attain better linear relaxation bounds. We show how to interpret the disaggregated models geometrically and examine their practical significance computationally on several networks with varying cost structures. In addition, we investigate the use of variable disaggregation to help solve a facility planning problem and the network loading problem. For each problem addressed, the disaggregated models provide significantly better lower bounds than the aggregated models. In fact, the objective value of the linear programming relaxation of the disaggregated formulation is frequently within 1% of the optimal objective value. In addition, if the resulting formulation is too large to solve directly with branch and bound, we can often use the linear relaxation solution in a rounding heuristic to attain quality integer, feasible solutions.

We apply these techniques to a specific application in supply chain management called merge-in-transit. Merge-in-transit is a two-echelon distribution system that might be appropriate when customer orders are composed of components produced at geographically dispersed locations. We develop a basic model that addresses merge-in-transit operations and then use disaggregation to improve the formulation. Because the models quickly become too large to solve using traditional approaches, we develop and implement an algorithm that combines row and column generation, rounding heuristics, and branch and bound to efficiently solve realistically sized problems.

Thesis Supervisor: Thomas L. Magnanti
Title: Dean of Engineering and Institute Professor, MIT

Acknowledgements

A year before submitting this thesis, I wrote these acknowledgements. In some ways, I feel it is one of the most important parts of my thesis, so I suppose it is fitting that I wrote it first. I did so after hearing the song "Kind and Generous" by Natalie Merchant on the radio. So to all those that have been so kind and generous I say:

You've been so kind and generous
I don't know how you keep on giving
For your kindness, I'm in debt to you
For your selflessness, my admiration
For everything you've done, you know I'm bound
I'm bound to thank you for it

You've been so kind and generous
I don't know how you keep on giving
For your kindness, I'm in debt to you
And I never could have come, this far without you
For everything you've done, you know I'm bound
I'm bound to thank you for it

Oh, I want to thank you for so many gifts you gave
The love, the tenderness, I wanna thank you
I want to thank you for your generosity, the love
And the honesty that you gave me
I want to thank you and show my gratitude
My love, and my respect for you, I want to thank you
Oh, I want to thank you, thank you, thank you, thank you

The first verse of this song expresses my gratitude to my advisor, Tom Magnanti. His support and unwavering encouragement kept me on track and plowing forward. My admiration is inexpressible. He stands as a role model for me to follow as a professor and a scholar. His dedication to his students is something that I hope to emulate throughout my career. In a similar role, Bernard Gendron was an immense help. He always took the time to help me think things through and in the process taught me a considerable amount about how to conduct research. So Tom and Bernard, I thank you for being so kind and generous.

The second verse I dedicate to my friends and colleagues who made both my academic life survivable and my social life enjoyable. My first year in particular was defined by the support and friendship of my fellow "first-yearers," especially Rebecca D'Amato, Leon Hsu, Tim Kniker, Joel Sokol, and Beril Toktay. For my sanity throughout this program I thank Megan White and Beth Kastner. They reminded me that there is life outside of MIT and it should be enjoyed with laughter and spontaneity. To all my friends everywhere, I thank you for being so kind and generous.

Finally, the third verse starts to put into words my gratitude to my family. The gifts my parents have given me are truly immeasurable. They helped me keep everything in perspective and not lose myself in the trials and tribulations of life. They celebrated with me those things that were to be celebrated and brought me out of the disappointments along the way. Likewise, I leaned on Jim, Dave, DaVaun and Claire who were always ready with advice and a listening ear. Their love, support, and laughter have kept my life balanced. And then there's Mickey and Kyle who have brought me the

most laughter and joy over the last five years. A better reminder of the real meaning in life exists nowhere. This thesis however, I dedicate to my grandfather, George Heinzelman, without whose engineering genes I wouldn't be writing this thesis. I know he would be so proud to call me "Dr. Croxton" and see me teaching at his alma mater. I love him and miss him. So to all my family, I want to thank you, thank you, thank you, for being so kind and generous.

Special thanks also go to my committee members, Steve Graves and Jim Orlin. I thank Jim Rice for his support with the ISCM and his financial acumen. In addition, my time at MIT was generously supported by the National Science Foundation, the Department of Transportation, and the Integrated Supply Chain Management Program (in particular Caliber Logistics). To my family and friends I am forever in debt, but without these sources of financial aid, I would be in a different kind of debt. So a special thank you to them.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Supply chain management is a vast field that touches upon a wide range of issues – from implementing integrative strategic management to developing precise operating models. Supply chains are important in practice, as reflected for example, in the U.S. logistics expenditure of almost $900 billion annually[1]. In fact, the logistics share of the gross domestic product in the U.S. is larger than that of health care, social security or defense. Moreover, logistics costs account for nearly 30% of the sales dollar in the U.S[2]. The design and operation of these supply chains raise many fascinating research issues. Formal models of supply chains are often complex and can be very large and difficult to solve. For example, supply chain optimization often leads to large-scale models with tens of thousands of variables and constraints and complex cost structures. This thesis addresses some generic modeling issues that arise in supply chain optimization, as well as other application domains, and then examines a specific application context called merge-in-transit.

This thesis is organized into five chapters. In Chapter 2, we provide an introduction to supply chain management and motivate the use of optimization techniques. We also highlight several companies that have successfully implemented optimization models to address a variety of supply chain issues. We can model many of the underlying logistics issues in supply chain management as network problems. Due to economies of scale inherent in the rates charged by transportation companies, the flows on the networks often have non-linear flow costs. We

[1] Delaney, R.V., 10th Annual "State of Logistics Report", 1999, Cass Information Systems, St. Louis, MO.
[2] Kasilingam, R.G., *Logistics and Transportation, Design and Planning*, Kluwer Academic, 1999.

will focus on situations with piecewise linear flow costs.

In Chapter 3 we discuss and compare modeling techniques for these problems. In particular, we examine three common models (multiple choice, incremental, and convex combination) and prove the equivalency of their linear programming relaxations. We show that each relaxation estimates the actual cost function with its lower convex envelope. In many situations, we are able to improve upon this estimate by disaggregating the variables to obtain tighter formulations. We provide a structural result stating that the disaggregated formulation approximates the actual cost function with its lower convex envelope in k dimensions, where k is the dimensionality of the disaggregation. We use one-arc, three dimensional examples to visualize the effects of disaggregation. For situations with concave flow costs, we are able to express the impact of disaggregation explicitly in closed form. In studying this disaggregation technique, we will apply it to five network problems: (1) a network flow problem with a single origin or destination, (2) a multi-commodity network flow problem, (3) a facility location problem with multiple capacity options, (4) the network loading problem, and (5) the merge-in-transit problem.

Chapter 4 examines the computational impact of the disaggregation techniques discussed in Chapter 3 in solving the linear programming relaxations for several networks with a variety of cost structures. As these results show, the disaggregated models provide much tighter relaxations than aggregated models. In fact, for network flow problems with a single origin or destination and concave costs, the solution to the relaxed disaggregated models are frequently integral. For other cost structures, the objective value of the disaggregated formulations are often within 3% of the optimal objective value, representing a substantial improvement upon the aggregated formulation. For example, on a particularly striking instance, the objective value of the linear relaxation of the aggregated formulation was over 500% from the optimal objective value while the objective value of the relaxation of the disaggregated model was within 1%. For large instances, we use an easily implemented rounding heuristic to obtain very good upper bounds. In addition to reporting on a set of random instances, we study the relaxation gap of the various formulations on a multi-commodity network with concave costs as we vary the initial fixed charge, the number of commodities, and the magnitude of the demands. This analysis provides a sense for how various network attributes effect

the performance of both aggregated and disaggregated formulations. We find that the fully disaggregated formulation is consistently tight (with objective values for the linear relaxations usually within 1% of optimal), but the relaxation gaps for the formulations with aggregated variables increase sharply as the value of the initial fixed charge increases. This result suggests that disaggregation is particularly useful on problems with large fixed charges. We obtain similar results as we vary the magnitude of the demands. We also examine computational results for the facility location and network loading problems, which both possess staircase cost functions. We again find that disaggregation can be very helpful in obtaining better lower bounds, however, solving the linear relaxation of instances with staircase costs can be very time consuming.

In Chapter 5 we introduce and examine the merge-in-transit problem, applying the previous analysis to develop strong formulations and good solution heuristics for this topical application in supply chain management. We describe the problem context and the specific operational issues that we wish to model. As we define and model it, the merge-in-transit problem is a multi-commodity, integer, generalized network flow problem. For realistically sized instances, even the most basic formulation contains over 60,000 constraints and 100,000 variables, mostly integer. We show how we can apply the disaggregation techniques discussed in Chapter 3 to develop several valid formulations for this problem. As we disaggregate, the formulations become larger (in most cases at least doubling the size of the basic formulation), but the linear relaxations become tighter. By employing dynamic column and constraint generation, we implement an algorithm that is able to solve realistically sized problems to within 5% of optimal in 8-10 hours.

Finally, Chapter 6 reviews the contributions of this research and provides a glimpse into potential future research directions.

Chapter 2

An Introduction to Supply Chain Management

2.1 Definitions of Supply Chain Management and Logistics

As articulated by Lambert et. al [21], *supply chain management* is "the integration of key business processes from end user through original suppliers that provides products, services, and information that add value for customers and other stakeholders." According to this definition, the field includes product development, procurement, manufacturing, demand management, customer relationship management, order fulfillment and distribution. Supply chain management attempts to cut-across organizational boundaries and manage inter-corporate processes.

Each company must determine the scope of its supply chain considerations as well as the type of relationships that it desires with each member of its supply chain. These considerations often lead companies to forge strong partnerships with key suppliers and/or customers. It is usually unnecessary to foster relationships with every member of the supply chain. For instance, a cereal company might desire a strong relationship with its box supplier, but not require a strong relationship, or likely any relationship at all, with the company that grows the trees used to make the pulp for these boxes. As each company examines its supply chain, it needs to assess the direct value of each member of its supply process and determine if it will manage the entire chain. Supply chain management, then, is about managing relationships

and processes.

Logistics is a key component of supply chain management. As defined by the Council for Logistics Management, *logistics* is "that part of the supply chain process that plans, implements, and controls the efficient, effective flow and storage of goods, services, and related information from point-of-origin to the point-of-consumption in order to meet customers' requirements." In this part of supply chain management, analytical modeling can play a particularly important role.

2.2 Supply Chain Networks

There are many ways to define and conceive of supply chains. As shown in Figure 2-1, supply chains typically have an underlying network structure that represents the flow of material, people, and financial resources from material/resource procurement, through product assembly/manufacture, and on to distribution centers and final customers. These entities are often competing for scarce or shared resources, including space, time, money, and people. The stylized network shown somewhat arbitrarily divides the overall supply chain into three interlocking components – a procurement/sourcing network, an assembly/manufacturing network, and a distribution network. The figure also shows parallel networks representing two products.

The complexity of the supply chain network will depend upon the underlying application context. The procurement network could have tens, hundreds, or even thousands of supply points representing both internal sources and outside vendors for raw material and components. The network might have its own assembly structure as the supply chain converts raw materials into intermediate components. This assembly/manufacturing network could represent tens of geographically dispersed plants. The distribution network could have tens or hundreds of warehouses and distribution centers and hundreds or thousands of dealers, retail outlets or customer zones. Moreover, the supply chain might be producing hundreds of related products and thousands or hundreds of thousands of stock keeping units (SKUs).

The efficiency of this network has a significant influence on the price and delivery time of the final product. Therefore, coordinating network activities is of primary importance for achieving good system performance. The coordination/optimization of the supply chain is complicated by several factors:

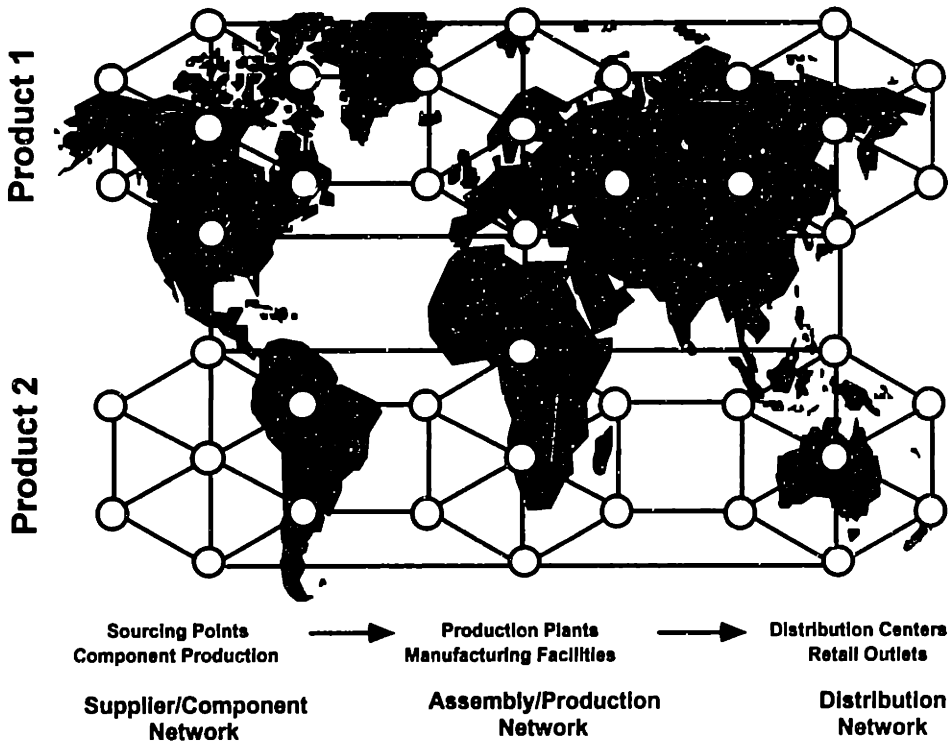

Figure 2-1: A Typical Supply Chain Network

- The various components, products, and distribution channels are sharing common resources (scarce production or distribution facilities, people, financial resources).
- The flows in the network are linked by precedence relationships, timing, and other "flow" interactions.
- The system is simultaneously attempting to optimize/balance multiple metrics including costs, asset utilization, customer responsiveness and service, flow time, and inventory.
- Typically, multiple organizations within a corporation (procurement, marketing, manufacturing, transportation, logistics, customer service) will have ownership and responsibility for various components of the supply chain.
- Global networks must include the effects and restrictions of international trade laws, duty and duty drawback, and international taxes.
- The supply chain is combinatorially complex, with an enormous number of options for

its design and operation.

2.3 Optimizing the Supply Chain

There are many opportunities for optimization within supply chains, ranging from a localized perspective (e.g., optimizing yield on any piece of equipment) to increasingly broader system-wide effects, for example, optimization within any node (a plant, a warehouse), optimization over some sub-network, and overall optimization of the entire network (often a daunting task). Models can answer a variety of issues that aid in coordinating within and across the supply chain. They also can range from high-level strategic decision-making to everyday operational issues. Some examples include:

- sourcing choices
- production scheduling
- plant loading
- technology/equipment choices
- inventory positioning and levels
- degree of outsourcing and vertical integration
- assignment of customers to distribution centers
- vehicle routing and crew scheduling
- location of plants and distribution centers
- transportation mode selection

The available literature and software has traditionally focused on small pieces of the supply chain network. The research and practitioner communities have successfully addressed many issues in the Distribution Network, with algorithms and software developed for making decisions regarding location of distribution centers and design of the distribution network including the assignment of warehouses to plants and customers to warehouses. The production scheduling and inventory control literature has addressed the Assembly/Production Network extensively. Less work seems to have examined optimization of the Supplier/Component Network, although there are examples of companies developing models for examining this piece of the supply chain. Figure 2-2 shows some examples of how and where some companies have used optimization.

As shown in Figure 2-2, several models address both the Assembly/Production Network and the Distribution Network. These models simultaneously answer questions concerning product allocation to plants and the design of distribution networks. In addition, two companies have created models to optimize across all three sub-networks. The GM model [7] focuses on product allocation and planning, but includes modules that allow simultaneous analysis of the

other components of the supply chain. It also has the capability to consider global issues. The DEC model [2], developed in collaboration with Insight, is the most comprehensive model that we have encountered. It is truly a global optimization model. The system recommends a combined vendor, production, and distribution network that minimizes cost or weighted cumulative production and distribution times, subject to meeting estimated demand and restrictions on local content, offset trade, and joint capacity for multiple products, echelons, and time periods.

Experiences vary across companies, but most report that with careful and proper implementation, optimization has provided improved solutions over other techniques, such as simulation and heuristic methods, and have provided means for substantial cost savings. For example, although it took years to develop, estimates indicate that the DEC/Insight model has helped save the company over $100 million. And although the Libbey-Owens-Ford model [23] has a much smaller scope, it has provided the company with annual savings of over $2 million. Similarly, DowBrands [27] attributes an annual savings in logistics costs of $1.5 million to its distribution model.

These problems are naturally large. In order to represent them with models that are of a solvable size, analysts commonly use aggregation and pre-selection. The Hunt-Wesson model [20], for example, aggregated hundreds of products and many thousands of customers into 17 product groups and 121 customer zones. It also eliminated thousands of variables by limiting possible plant/product group assignments, distribution center locations, and distribution center/customer zone assignments to those that were "reasonable." For example, the model does not create a variable representing the assignment of a customer on the east coast to a warehouse location on the west coast. Using aggregation and pre-selection, analysts reduced the Hunt-Wesson problem to a size that could be solved efficiently. Finding ways to aggregate variables and constraints without losing information is often an important consideration when developing a large scale model (note: this aggregation of data is different than the model aggregation we discuss in Chapter 3). Even with significant aggregation, these problems can be quite large. For instance, the Libbey-Owens-Ford model deals with four plants, over 200 products, and over 40 demand centers in a 12-month planning horizon. The resulting linear program has approximately 99,000 variables and 26,000 constraints and requires three to four hours to solve on a Sun Sparcstation. The DEC/Insight model typically solves mixed-integer

problems with 2,000-6,000 constraints and up to 20,000 variables.

The algorithms for these models use various advanced solution techniques. Many of them exploit the underlying network structure and use decomposition or relaxation techniques to efficiently solve them. The Hunt-Wesson model was the first to implement Bender's Decomposition [15], and some recent models still rely on this approach. The DEC/Insight model uses elastic constraints, row factorization, and scaling.

In some supply chain optimization efforts, like the DEC/Insight project, developing and implementing the appropriate model is a difficult task and, by itself, is a contribution to the literature. In other efforts, such as the Hunt-Wesson model, the algorithmic development requires the most insight and dedication. Regardless of the focus, there are many ways that optimizing parts, or all, of the supply chain can contribute to both the research and practitioner communities.

Although many companies do not use optimization as part of their supply chain toolkit, the use of software solutions is growing. Companies such as I2, CAPS and Manugistics are leading the industry with customizable software that uses optimization models and good-performing heuristics to solve many supply chain issues.

2.4 Summary

The field of supply chain management and logistics has grown substantially in the last ten years. Companies are increasingly looking to cross functional borders and find better ways to manage the entire supply chain, from procurement through distribution. A variety of software companies are attempting to profit from this growth through software solutions. In addition, there has been a steep increase in the number of third party logistics companies. In the last ten years, the number of contract logistics companies has grown from a few to hundreds[1].

As the need for faster, less expensive, and more effective logistics solutions has grown, the role of optimization has become more clear. Many companies have already developed optimization models to address a variety of issues, ranging from strategic decisions such as facility location, to operational decisions about production scheduling and vehicle routing.

[1] Yossi Sheffi, Director of the MIT Center for Transportation Studies, http://web.mit.edu/afs/athena.mit.edu/org/c/cts/www/education/ed_mlogwhystudy.html

These models are attempting to manage a very complex system, and as a result, they often require clever modeling techniques, as well as advanced algorithmic approaches.

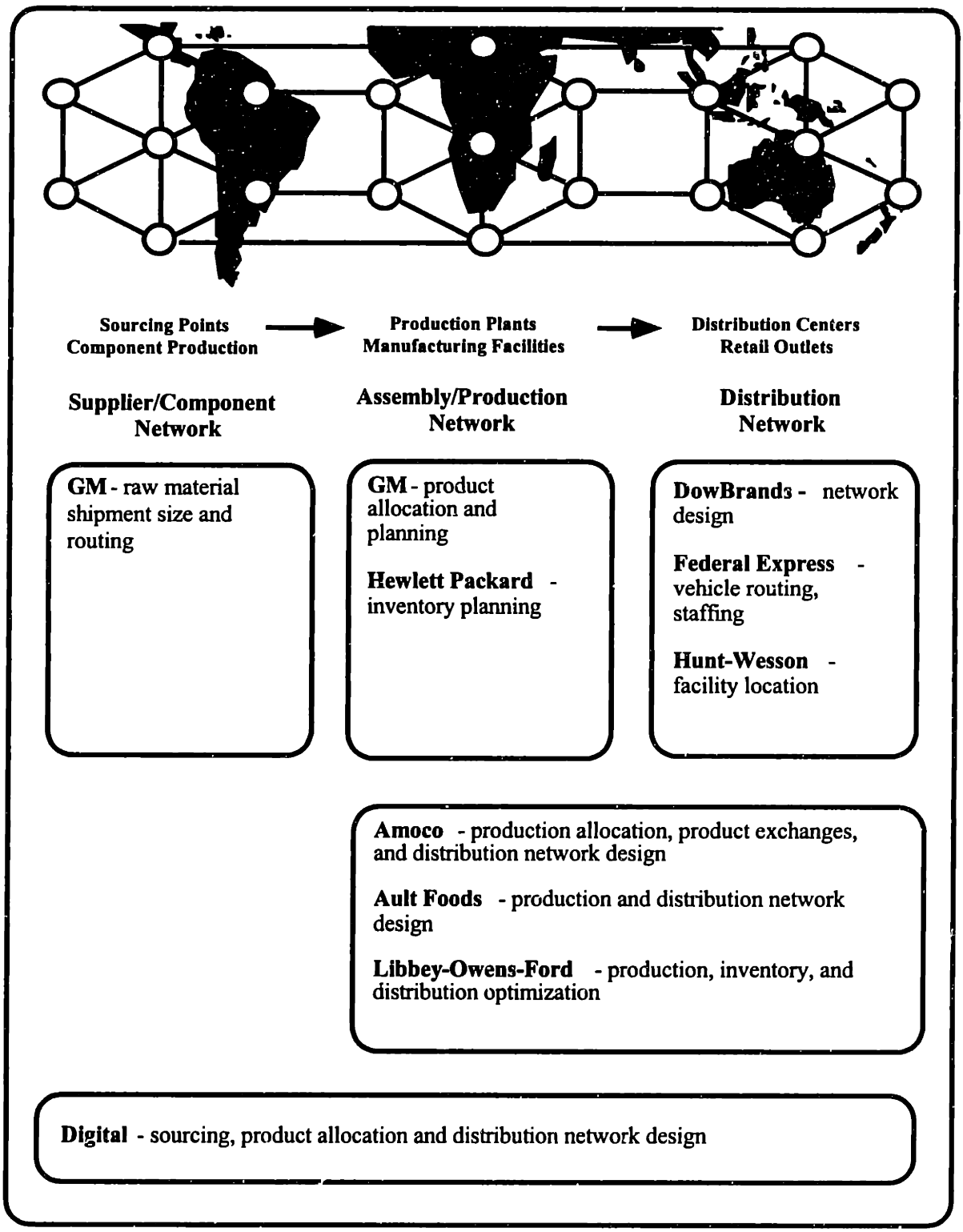

Figure 2-2: Examples of Optimization Applications

Chapter 3

Modeling Network Flow Problems with Piecewise Linear Cost Functions

3.1 Introduction

As indicated in the previous chapter, many supply chain and logistics problems can be viewed and modeled as networks. The cost structure of these systems are often complex, with fixed charges and possibly discontinuous piecewise linear flow costs. The importance of these problems motivates the study of more general network flow problems with piecewise linear costs. The insight gained from this investigation could be used in modeling and solving not only logistics problems, but also problems in other application domains such as transportation and telecommunications.

Within this chapter we will describe a generic problem, review relevant literature, and discuss modeling approaches. We will then examine the use of disaggregation to improve the linear programming relaxation of the problem and gain some structural insight into the effects of disaggregation. Along the way, we will examine disaggregation applied to several specialized problems.

3.2 The Problem

We are given a directed network, $G = \{V, A\}$, with a vector, d, of supplies and demands of a single commodity at the nodes. We consider the problem of finding the minimum cost flow when the cost, $g_e(x_e)$, on arc e is a piecewise linear function of x_e, the total flow on the arc, and might include fixed charges. Letting $g(x) = \sum_{e \in A} g_e(x_e)$, we can express the network flow problem as follows:

$$\text{Minimize } g(x) \tag{3.1}$$

$$\text{subject to: } Nx = d \tag{3.2}$$

$$x \geq 0. \tag{3.3}$$

Constraints (3.2) are the node balance constraints typical in a network flow formulation. N denotes a standard node-arc incidence matrix. For each node, $d_i > 0$ denotes a demand node with demand d_i, $d_i < 0$ denotes a supply node with supply $-d_i$, and $d_i = 0$ denotes a transshipment node. In our context, this formulation is, in general, nonlinear since the cost function can be nonlinear.

We will also consider multi-commodity versions of this problem where each commodity has a specific origin and destination. In this case, we will define x^k as the flow of commodity k, and let $x = \sum_k x^k$. In some applications we could consider, we might include capacity restrictions or other constraints. As we will see, we can add arc capacities through the definition of the cost function and do not require additional constraints beyond the basic formulation. Some problems, however, require other complicating constraints. For simplicity, we consider the most basic formulation. The techniques and analysis presented are applicable for many situations with a more complex constraint structure. The merge-in-transit application we consider in Chapter 5 provides an example.

We know the general version of this problem is NP-hard because a special case is the fixed charge network flow problem. In this variant of the problem, we incur a fixed charge for using an arc in the network. For example, in designing a telecommunications network, we pay a fixed charge for installing a line, but incur no marginal unit flow cost. This special

case is known to be NP-hard. Therefore, the more general problem we are considering is also NP-hard. We will examine methods of modeling the cost function and will seek formulations that provide strong linear programming relaxations.

3.3 Applications

Many network flow applications have piecewise linear flow costs. In Chapter 5, we examine one example, called the merge-in-transit problem, encountered in supply chain management. This problem can be defined as a generalized network flow problem. Two other applications, facility location and network loading, are often not modeled as network flow problems, but both can be interpreted within a network flow framework if we permit piecewise linear costs.

In the standard facility location problem, we need to determine which facilities to open to meet the demand of a given set of demand points. The cost associated with each facility includes a fixed charge for establishing the facility and often a linear production/transportation cost. Many real-world applications, however, possess a more complicated cost structure. Holmberg [18] and Holmberg and Ling [19] discuss situations in which both the fixed and variable costs vary for different production levels. This situation might arise if a facility can be opened with alternative capacities. The result is a staircase cost structure with breakpoints at each potential capacity, as in Figure 3-1. We can model this application as a network flow problem with piecewise linear flow costs, using a network like the one shown in Figure 3-2. The costs on the arcs from the dummy node into the facilities will have a structure shown in Figure 3-1, which includes both the set-up cost at the facility and any unit production costs. The arcs between facilities and customers will incur linear transportation costs.

For situations with no transportation costs, we can replace the set of customer nodes with a single destination node and include an arc from each facility to this aggregated node. We can then eliminate the facility nodes as well. As a result, this facility location problem can be reduced to a network with two nodes (the dummy node and a single aggregated node representing all the customers) and m parallel arcs, one for each facility. This case can be extended to the situation when for any fixed customer, transportation costs from all the facility locations is the same. In this case, the total transportation cost is constant for all solutions and we can, therefore, remove it from the objective function and reduce the problem to one

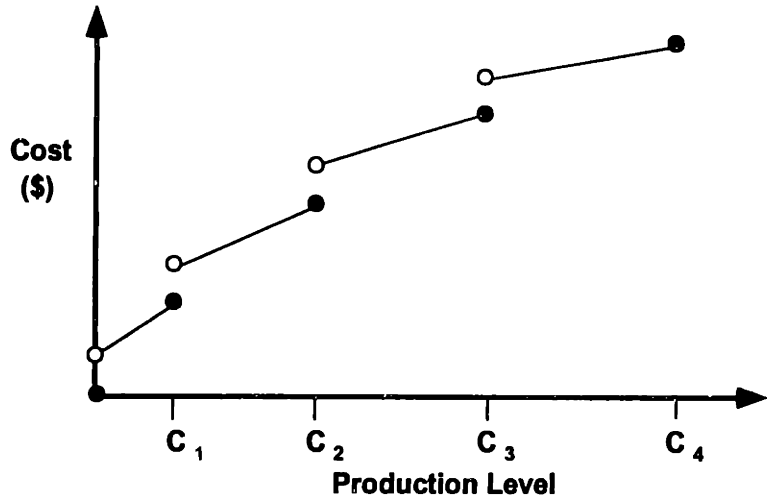

Figure 3-1: An Example Cost Function for the Facility Location Problem

with no transportation costs. As we will see in Chapter 4, we can solve the linear program of this special case using a greedy algorithm and use this solution to easily generate a feasible integer solution.

The network loading problem is another application that we can consider in the context of network flow problems. In this problem we need to install sufficient capacity on a network to simultaneously route a set of demands between pairs of nodes. We can install capacity in bundles of C units and often several types of capacity, with varying values of C, are available to be installed on each edge. This problem arises in telecommunications when we can install lines with varying capacities on each edge of the network. As an example, we can install two types of lines, OC1 or OC4 lines. The capacity of an OC4 is 4 times that of an OC1, but its cost is less than 4 times as much. Therefore, we incur economies of scale as flow values increase. The problem is to find a least cost combination of OC1 and OC4 lines to install on each edge of the network so that it has sufficient capacity on each edge to carry all the demand. For these telecommunications applications, we generally consider an undirected network so the edge capacity is shared by flow on both the directed arcs, (i, j) and (j, i). For any given flow value on an edge, some combination of OC1 and OC4 lines provides the most cost-effective loading. We can therefore consider the installation charge as a flow cost. The resulting cost function on each arc is piecewise linear and takes the form shown in Figure 3-3,

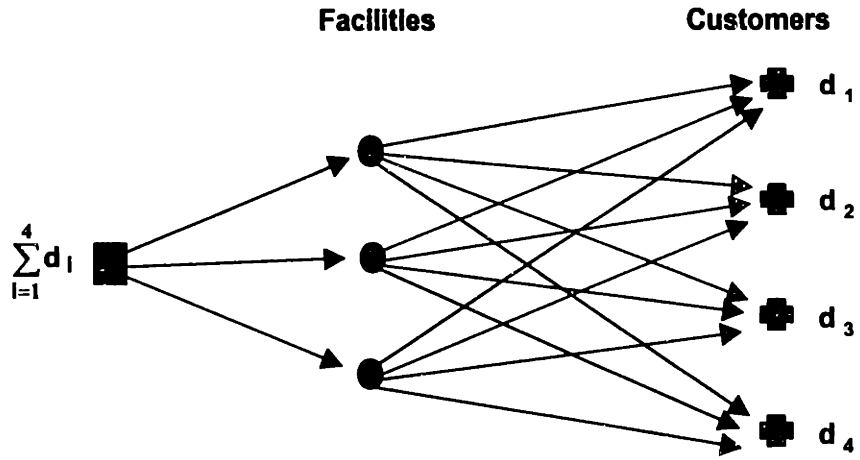

Figure 3-2: An Example of a Network for the Facility Location Problem

which corresponds to a situation where the cost of an OC1 line is three times as much as an OC4 line.

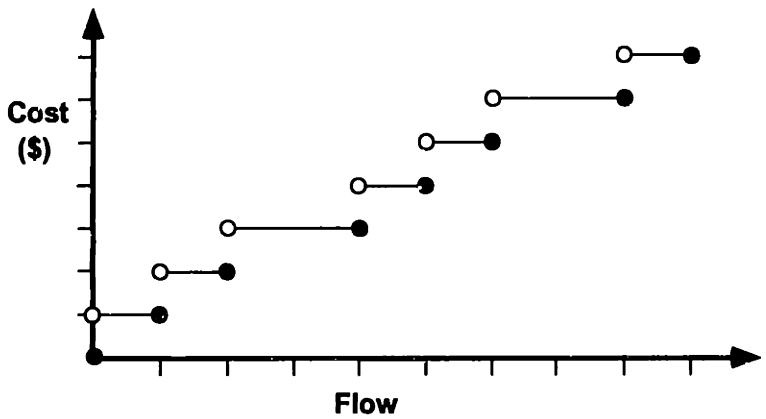

Figure 3-3: An Example Cost Function for the Network Loading Problem

Researchers most often discuss the network loading problem as an application in telecommunications, but it also arises in logistics. In this context, we are given freight that needs to be moved between pairs of nodes of a network. This freight can be shipped in containers of various sizes. The problem is to determine how many containers of each size will be required on each arc to simultaneously route all the demand at a minimum cost. This problem varies from the telecommunications application since the network is directed, and therefore the

capacity installed on arc (i, j) is not available to route freight on the reverse arc (j, i).

The merge-in-transit, facility location, and network loading problems are three examples of network flow problems with piecewise linear costs. Each application could benefit from the modeling techniques we discuss and the insight we gain in this chapter. We will examine the facility location problem and both the directed and undirected versions of the network loading problem more closely later in this chapter and in Chapter 4.

3.4 Literature Review

Although the literature on network flows is extensive, work on modeling problems with piecewise linear cost functions is limited. Balakrishnan and Graves [3] develop a Lagrangian-based algorithm for the uncapacitated network flow problem with piecewise linear concave flow costs. Although their approach is generic and can be implemented for the non-concave case, to our knowledge, the literature contains no computational experience for implementations of this algorithm on networks with non-concave costs. Cominetti and Ortega [10] solve the capacitated network flow problem with piecewise linear concave costs with a branch-and-bound-based algorithm. They exploit the concavity of the cost functions and employ sensitivity analysis to improve the cost approximations and obtain improved lower bounds. Motivated by a logistics application, Chan, Muriel, and Simchi-Levi [8] examine the multi-commodity version of the same problem. They derive structural results on a set-partitioning formulation, and then use this insight to develop a linear programming based heuristic.

Other relevant work includes Popken [26] who studies the multi-commodity flow problem and develops an algorithm that is valid for non-concave cost structures, assuming they are continuously differentiable. Holmberg [18] develops an algorithm for the facility location problem based on convex linearization and Bender's decomposition. Holmberg and Ling [19] use a Lagrangian heuristic for the same problem. In both papers, the cost functions are discontinuous, and thus non-concave, but the algorithms developed are specific to the facility location problem. The network loading application is covered extensively in [13].

Network design problems are related to this work. Magnanti and Wong [22] survey network design issues in transportation planning, and Balakrishnan, Magnanti, and Mirchandani [4] provide an annotated bibliography on general network design issues.

In this research, we study the effects of model disaggregation. A common approach in solving large logistics models is to use aggregation to decrease the size of the problems. Geoffrion [14] examines the issue of aggregating data in logistics planning models and develops error bounds. In two papers, [29] and [30], Zipkin studies the implications of both row and column aggregation and develops a priori and posteriori bounds on the resulting loss of accuracy. Mendelssohn [24] improves upon these bounds. In another study of aggregation, Hallefjord and Storoy [16] examine column aggregation of 0/1 programming problems and develop methods for improving the bounds through valid inequalities and exploiting primal degeneracy. Although the aggregation studied in these papers has a different flavor and purpose than the disaggregation we discuss, we can apply some of the insight we gain through studying disaggregation to interpret the effects of aggregation.

3.5 Modeling the Cost Function

We will discuss three standard textbook methods for modeling piecewise linear cost functions. All three introduce integer variables. We consider an arbitrary piecewise linear cost function like the one in Figure 3-4. The cost is a function of the total flow on the arc, with the unit flow charge and fixed charge varying according to the load on the arc. The function need not be continuous; it can have positive or negative jumps, though we do assume that the function is lower semi-continuous. That is, $g(x) \leq \lim_{i\to\infty} g(x^i)$ for any sequence x^i that approaches x. Without loss of generality, our models assume, through a simple translation of the costs if necessary, that $g(0) = 0$. As we will see, real-world applications often give rise to piecewise linear cost functions that are, in fact, neither convex nor concave.

To define a piecewise linear cost function, we need to describe its "pieces." Figure 3-5 illustrates the notation. On each arc e, each piecewise linear segment of the function has a variable cost, c_e^s (the slope), a fixed cost, f_e^s (the cost-intercept), and upper and lower bounds, b_e^{s-1} and b_e^s, on the flow of that segment.

Notice that we can model problems with arc capacities by defining the cost function only up to the capacity level on each arc. The bound b_e^s on the final segment will equal the capacity on the arc. This allows us to model capacitated problems through the definition of the cost function.

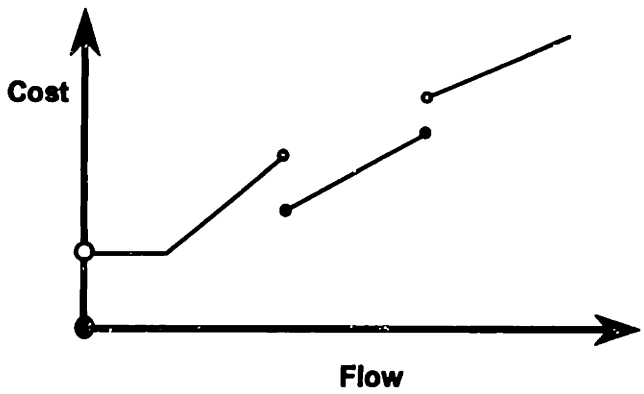

Figure 3-4: An Example Piecewise Linear Cost Function

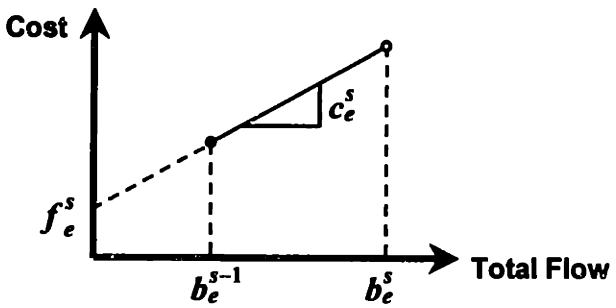

Figure 3-5: Notation for Each Segment

We now present three valid models for the network flow problem with piecewise linear costs.

3.5.1 Incremental Model

As reported in Bradley, Hax, and Magnanti [5] and several other basic textbooks, in the incremental model we define x_e^s as the amount of flow in the sth segment of arc e. The total flow on an arc, $x_e = \sum_s x_e^s$, is the sum of the incremental flows in each segment. To simplify our notation, we will write x_e^s as x^s, omitting the edge subscript. We will use this convention throughout this thesis for other variables and data as well, though we should remember that these variables and data are defined for each arc. In addition, the number of segments per arc might vary, so when we write an expression such as $\sum_s x_e^s$, we intend for the summation to be the sum over all segments on arc e.

To be feasible using this flow variable definition, the flow on segment $s+1$ must be zero unless segment s is "fully loaded." That is, $x^{s+1} > 0$ only if $x^s = b^s - b^{s-1}$. To model this requirement, we introduce binary variables, y^s, defined by the condition that $y^s = 1$ if segment s contains any flow, and $y^s = 0$ otherwise. In addition to manipulating the flow variables, the y variables also indicate whether we incur a fixed charge for that segment and, therefore, appear in the objective function. In this model, we define $\widehat{f}^s = (f^s + c^s b^{s-1}) - (f^{s-1} + c^{s-1} b^{s-1})$, the cost gap at the breakpoints between segments. In Figure 3-4, $\widehat{f}^1 > 0$ and equals the initial fixed charge at zero flow, $\widehat{f}^2 = 0$ because there is no gap at the low end of this segment, and $\widehat{f}^3 < 0$ and $\widehat{f}^4 > 0$ because the cost function jumps down and up, respectively, at these breakpoints. If the cost function is continuous, $\widehat{f}^s = 0$ for each segment s. We can now express the network flow problem given by (3.1)-(3.3) as a mixed-integer formulation as follows:

$$\text{Minimize} \sum_s c^s x^s + \widehat{f}^s y^s \tag{3.4}$$

$$\text{subject to:} \quad Nx = d \tag{3.5}$$

$$x = \sum_s x^s \tag{3.6}$$

$$(b^s - b^{s-1}) y^{s+1} \leq x^s \leq (b^s - b^{s-1}) y^s \tag{3.7}$$

$$x, x^s \geq 0, \quad y^s \in \{0, 1\}. \tag{3.8}$$

Again, the equalities (3.5) express the flow balance constraints. In (3.6) we define the flow on an arc as the sum of the segment flows. The two constraints in (3.7) assure that we properly assign the y variables. They assure that if $y^{s+1} = 1$, then $y^s = 1$ and $x^s = b^s - b^{s-1}$, so that the previous segment is fully loaded. On the other hand, if $y^{s+1} = 0$ and $y^s = 1$, then x^s must lie within the width of segment s, i.e., $x^s \leq b^s - b^{s-1}$. And finally, if $y^s = 0$, then $y^{s+1} = 0$ and $x^s = 0$. Constraints (3.8) define the flow variables to be nonnegative and the y variables to be binary.

3.5.2 Multiple Choice Model

The multiple choice model, used by Balakrishnan and Graves [3], uses an alternative definition of the flow variables. In this formulation, x^s equals the total flow on the arc if that flow lies in

segment s. For this definition, when a flow value of F lies in segment $\hat{s}$, $x^{\hat{s}} = F$ and $x^s = 0$ for all segments $s \neq \hat{s}$. As in the incremental formulation, $y^s = 1$ if segment s contains any flow, and $y^s = 0$ otherwise, but in this formulation at most one y^s will equal one. For this definition of the y variables, if $y^s = 1$, we want to include f^s in the objective function. The resulting formulation is:

$$\text{Minimize} \sum_s c^s x^s + f^s y^s \tag{3.9}$$

$$\text{subject to: } Nx = d \tag{3.10}$$

$$x = \sum_s x^s \tag{3.11}$$

$$b^{s-1} y^s \leq x^s \leq b^s y^s \tag{3.12}$$

$$\sum_s y^s \leq 1 \tag{3.13}$$

$$x, x^s \geq 0, \quad y^s \in \{0, 1\}. \tag{3.14}$$

The constraints in the multiple choice formulation differ from those in the incremental formulation in two ways. First, the addition of the inequality (3.13) assures that we choose at most one y^s to be positive on each arc. Second, the segment bound constraints, (3.12), are different than (3.7). They state that if $y^s = 0$, then segment s has no flow, i.e., $x^s = 0$. If $y^s = 1$, then the total flow on the arc must lie between the breakpoints of that segment, i.e., $b^{s-1} \leq x^s \leq b^s$.

3.5.3 Convex Combination Model

The third formulation we examine is a modification of the formulation presented in Nemhauser and Wolsey [25]. The formulation they present is not valid for discontinuous cost functions, so we modified it to handle arbitrary functions. This formulation makes use of the fact that in a piecewise linear cost function, the cost of a flow that lies in segment s is a convex combination of the cost of the two endpoints of segment s, b^{s-1} and b^s. If we define multipliers, μ^s and λ^s,

as the weights on these two endpoints, then the following formulation is valid:

$$\text{Minimize} \quad \sum_s \mu^s (c^s b^{s-1} + f^s) + \lambda^s (c^s b^s + f^s) \tag{3.15}$$

$$\text{subject to: } Nx = d \tag{3.16}$$

$$x = \sum_s (\mu^s b^{s-1} + \lambda^s b^s) \tag{3.17}$$

$$\mu^s + \lambda^s = y^s \tag{3.18}$$

$$\sum_s y^s \leq 1 \tag{3.19}$$

$$x,\ \mu^s, \lambda^s \geq 0,\ \ y^s \in \{0,1\}. \tag{3.20}$$

In this formulation, the y variables carry the same interpretation as in the multiple choice model. Constraint (3.19) assures that at most one of the y variables has value one on each arc. Constraint (3.18) assures that $\mu^s + \lambda^s = 1$ for the segment corresponding to the positive y^s variable, and that μ^s and λ^s are both zero for the other segments. Constraint (3.17) defines the total flow to be the convex combination of the two endpoints defined by μ and λ, and the objective function evaluates the appropriate convex combination of the cost of these two endpoints.

Note that in all three of these formulations, we can rewrite the flow balance constraints using the flow definition constraints, and thereby eliminate the x variables.

3.6 Comparing the Three Models

Given that all three of the previous formulations are valid, it is natural to ask if one is better than another. An important measure for assessing the quality of a mixed integer programming formulation is the strength of its linear relaxation. The following result characterizes the relaxations of these three formulations.

Proposition 1 *The linear programming relaxations of the incremental, multiple choice, and convex combination formulations are equivalent, in the sense that they each approximate the real cost function with its lower convex envelope.*

To establish this result, we need to show that for any arc e with a total flow $\widehat{x}_e$, the objective value of the linear relaxation obtained by optimally choosing the other variables is given by

the lower convex envelope of the cost function on arc e. As before, we omit the arc index e. We will consider each formulation separately.

Proof: Convex Combination Formulation

By relaxing the integrality restriction on the y variables, we can combine constraints (3.18) and (3.19) into $\sum_s(\mu^s + \lambda^s) \leq 1$ and we can eliminate the y variables. Any representation of $\widehat{x}$ as a convex combination of the breakpoints, therefore, provides a feasible solution. The cost minimizing convex combination is given by the lower convex envelope of the cost function. ■

Proof: Multiple Choice Formulation

1. We will prove the result by first showing that every extreme point of the linear relaxation is a convex combination of two endpoints of the piecewise linear segments. We first note two facts, (a) and (b).

(a). Every extreme point of the polyhedron $\widehat{P}=\{(x,y) \in \Re^{2s} : b^{s-1}y^s \leq x^s \leq b^s y^s \ \forall s, \sum_s y^s \leq 1, y \geq 0, x \geq 0\}$ is an endpoint of one of the segments of the piecewise linear cost function. To see this result, suppose we optimize some cost function $\sum_s c^s x^s + f^s y^s$ over the polyhedron $\widehat{P}$. Note that if $c^s \geq 0$, then $x^s = b^{s-1}y^s$ in some optimal solution of the linear program and if $c^s \leq 0$, then $x^s = b^s y^s$ in some optimal solution of the linear program. Therefore, we can express each x^s in terms of the y^s variables and eliminate the x^s variables and the $b^{s-1}y^s \leq x^s \leq b^s y^s$ constraints from the model. The resulting problem has a linear objective function and the single constraint $\sum_s y^s \leq 1$ in the nonnegative y variables. Since the problem has a single constraint, it has a solution with at most one $y^s = 1$ and all other y variables at value zero. Since any nonzero such point with either $x^s = b^{s-1}y^s$ or $x^s = b^s y^s$ is an endpoint of the sth segment of the piecewise linear cost function, and we have shown that such a solution is optimal for every choice of the cost coefficients, we conclude that any extreme point of $\widehat{P}$ corresponds to an endpoint of a segment. Conversely, any endpoint of a segment corresponds to an extreme point of $\widehat{P}$.

(b). We know that if $\widehat{Q}$ is a bounded polyhedron in $\Re^k$ and we let $wz = w^o$ be any linear equation in $\Re^k$, then every extreme point in the polyhedron $Q=\{z \in \widehat{Q} : wz = w^o\}$ is a convex combination of at most two extreme points of the polyhedron $\widehat{Q}$ (this is a special case of the more general Lemma 10 provided in Section 3.9.1).

The polyhedron $P=\{(x,y) \in \Re^{2s} : \sum_s x^s = \widehat{x},\ b^{s-1}y^s \leq x^s \leq b^s y^s\ \forall s,\ \sum_s y^s \leq 1,\ y \geq 0,\ x \geq 0\}$ describes the linear relaxation of the multiple choice model. The results (a) and (b) imply that every extreme point of this polyhedron is a convex combination of two endpoints of the piecewise linear segments.

2. Next we will show that if $\widehat{x}$ is a convex combination of two segment breakpoints, then the corresponding solution (that is the solution corresponding to the same convex combination of the two extreme points of $\widehat{P}$ corresponding to each breakpoint) is feasible in P. Let $\widehat{x}$ be a convex combination of two segment breakpoints, b^{s1} and b^{s2}, i.e., $\widehat{x} = \lambda b^{s1} + (1-\lambda)b^{s2}$ for some $0 \leq \lambda \leq 1$. Let z^1 be the extreme point solution of $\widehat{P}$ corresponding to b^{s1} and, without loss of generality, let $x^{s1} = b^{s1}y^{s1}$. That is $z^1 = (x^1, x^2, ...x^S, y^1, y^2 ...y^S) = (0,0,...,b^{s1},0,...0,0,1,0,...,0,0)$ and similarly define z^2 to correspond to b^{s2}. We will now consider $\widehat{z} = \lambda z^1 + (1-\lambda)z^2 = (0,0,...,\lambda b^{s1},...,(1-\lambda)b^{s2},...,0,0,...,\lambda,0,...,1-\lambda,...,0)$. We know $\widehat{z} \in \widehat{P}$ because it is a convex combination of two of its extreme points. In addition, $\sum_s x^s = \lambda b^{s1} + (1-\lambda)b^{s2} = \widehat{x}$. Therefore, $\widehat{z} \in P$.

3. Point (1) implies that if we evaluate the objective function at an extreme point of P, then we are taking convex combinations of the endpoints of two segments of the cost function. When minimizing, we will choose the convex combination with least possible cost. Point (2) assures that this least cost convex combination is feasible. In addition, since the interior points of a segment are convex combinations of the endpoints for that segment, we know the lower convex envelope of the true cost function is defined by the convex combinations of the segment endpoints. Therefore, as we vary the parameter $\widehat{x}$, the optimal objective value specifies the least cost convex combination of the endpoints which defines the convex lower envelope of the true cost function. ■

Because we have established that an extreme point of the linear programming relaxation of the multiple choice formulation is a convex combination of two points where the y variables have either the form $(0,0,...,0)$ (corresponding to the origin) or $(0,0,...0,1,0,...,0)$, we have established the following result:

Corollary 2 *At an extreme point of the linear programming relaxation of the multiple choice formulation, the set of y variables assumes one of two forms: one $y^s > 0$ (when one of the points corresponds to the origin), or two $y^s > 0$ and their sum is one.*

Proof: Incremental Formulation

We can establish this result using the same approach as with the multiple choice formulation. We must establish, however, an analog of (a) for this formulation; letting $\Delta^s = b^s - b^{s-1}$, we wish to show that every extreme point of the polyhedron $\widehat{P}=\{(x,y) \in \Re^{2n} : \Delta^s y^{s+1} \leq x^s \leq \Delta^s y^s \ \forall s,\ 0 \leq y \leq 1\}$ is an endpoint of one of the segments of the piecewise linear cost function. Note that if $c^s \geq 0$, then $x^s = \Delta^s y^{s+1}$ in some optimal solution of the linear program and if $c^s \leq 0$, then $x^s = \Delta^s y^s$. Therefore, we can express each x^s in terms of the y^s variables and we are left with the following constraints: $0 \leq y \leq 1$ and $y^s \geq y^{s+1}$, for all segments s. The fact that an extreme point solution to this system must have S, the number of segments, independent binding constraints implies that all extreme points of $\widehat{P}$ will be of the form, $(1,1,...,1,0,0,...,0)$. Using these values of y to find the values for x, we see that each such extreme point corresponds to an endpoint of the piecewise linear segments. We can then conclude that any extreme point of $\widehat{P}$ corresponds to an endpoint of a segment.

The other steps of the proof are the same as those in the proof for the multiple choice formulation so we can conclude that as we vary $\widehat{x}$, the objective value defines the lower convex envelope of the true cost function. ■

Since an extreme point of the linear programming relaxation of the incremental formulation is a convex combination of two points where the y variables have the form $(1,1,...,1,0,0,...,0)$, this proof establishes the following result:

Corollary 3 *At an extreme point of the linear programming relaxation of the incremental formulation, the set of y variables will either be of the form* $(\overline{y},\overline{y},...,\overline{y},0,0,...,0)$ *(when one of the endpoints is the origin) or* $(1,1,...,1,\overline{y},\overline{y},...,\overline{y},0,0,...,0)$ *for some constant* $0 \leq \overline{y} \leq 1$.

Another way of viewing the equivalency of these formulations is to provide a translation between feasible points of each formulation.

Proposition 4 *The linear programming relaxations of the incremental, multiple choice, and convex combination formulations are equivalent, in the sense that we can translate a feasible solution of one formulation into a feasible solution to the others with the same cost.*

Proof: See Appendix A. ■

To see how "good" these formulations are, as measured by the gap between the solutions to the linear programming relaxation and the integer program, we can examine the difference between a function and its lower convex envelope. In general, we cannot place upper or lower bounds on this gap. As in Figure 3-6, the linear programming relaxation could have a value of zero, and so the gap can be infinite. For other instances however, there could be no gap. For example, a convex function with no initial fixed charge coincides with its lower convex envelope, and there will be no gap between the integer program and its linear relaxation. This observation is further validated by network theory from which we know that a network flow problem with a convex function and no initial fixed charge will have an integral linear programming solution. Although we cannot place a priori bounds on this gap, for any problem instance, examining the difference between the function and its lower convex envelope will give us advanced insight into how tight the linear programming relaxations to these formulations will be.

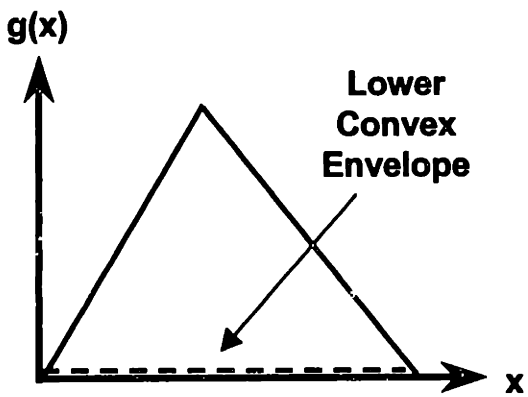

Figure 3-6: A Case With Potentially Infinite Gap Between $g(x)$ and its Lower Convex Envelope

Another measure of the quality of a MIP formulation is its size. If we let A, N, S denote the number of arcs, nodes, and segments respectively, we note the following:

Formulation	Variables	Constraints
Incremental	2AS	N+2AS
Multiple Choice	2AS	N+2AS+A
Convex Combination	3AS	N+AS+A

Although all three have the same number of binary variables, the convex combination formulation has more variables than the other two, but fewer constraints, and the incremental model has fewer constraints than the multiple choice model. The model size does not vary substantially however.

According to this measure, formulation size, there is a preference for either the incremental or the convex combination model. As we will see in the next section, however, the multiple choice model lends itself nicely to disaggregation. We will therefore focus our attention on this formulation.

3.7 Disaggregating the Multiple Choice Model

In some situations, we will be able to disaggregate each x_{ij} variable into a set of variables, x_{ij}^k, and place new bounds, M_{ij}^k, on each disaggregated variable. We then can rewrite the multiple choice formulation in a form with a stronger linear programming relaxation. For this disaggregation to be valid we need to be able to rewrite $Nx = d$ in a disaggregated form, $Nx^k = d^k$, for an appropriate choice of the demand vectors d^k. This disaggregation might take place at several levels. In the context of network flows, we can often associate the indices k with commodities and consider flows of individual commodities with bounds M^k. For instance, for a network with one source supplying several destinations, we can define a commodity by its destination and, if $d(k)$ denotes the demand at the destination used to define commodity k, set $M^k = d(k)$. Once we have defined a valid disaggregation, we can write the following

disaggregated version of the multiple choice model:

$$\text{Minimize} \quad \sum_{s,k} c^s x^{sk} + f^s y^s \tag{3.21}$$

$$\text{subject to:} \quad Nx^k = d^k \tag{3.22}$$

$$x^k = \sum_s x^{sk} \tag{3.23}$$

$$b^{s-1} y^s \leq \sum_k x^{sk} \leq b^s y^s \tag{3.24}$$

$$\sum_s y^s \leq 1 \tag{3.25}$$

$$x^k, x^{sk} \geq 0, \quad y^s \in \{0,1\}. \tag{3.26}$$

The advantage of this formulation is that we can add valid inequalities that tighten the formulation. We consider two forms of additional forcing constraints:

$$x^{sk} \leq M^k y^s \tag{3.27}$$

and

$$\sum_s x^{sk} \leq M^k \sum_s y^s. \tag{3.28}$$

Both constraints (3.27) and (3.28) are valid and redundant for the integer program, but can potentially strengthen the linear relaxation. We call the formulation consisting of constraints (3.21)-(3.26) and (3.27) the disaggregated formulation, and (3.21)-(3.26) and (3.28) the disaggregated/aggregated formulation (because the variables are disaggregated, but the forcing constraints are aggregated). Note that each inequality (3.28) is a sum over s of the (3.27) inequalities, and therefore, the disaggregated/aggregated formulation is not, in general, as strong as the disaggregated formulation. Both formulations, however, can improve upon the aggregated formulation given by (3.9)-(3.14).

When $M^k < b^s$, constraint (3.27) tightens the formulation. When $M^k \geq b^s$, constraint (3.27) does not improve the formulation because (3.24) already provides a tighter constraint and, therefore, constraint (3.27) is redundant. We cannot determine a priori if constraint (3.28) is redundant.

By disaggregating the convex combination formulation in a similar fashion, we will provide an equivalent formulation. We saw that the incremental model was smaller than the multiple choice model, so why wouldn't we choose to disaggregate that model instead? Although we can write a similar disaggregation, the forcing constraints might not result in as much improvement. The reason is that the disaggregated incremental model already contains the constraints $\sum_k x^{sk} \leq (b^s - b^{s-1})y^s$. Therefore, if $M^k \geq b^s - b^{s-1}$, the forcing constraints (3.27) will be weak. Problem instances with very small segments, relative to the demands, might have no strong forcing constraints to add. This limits the impact of disaggregation for the incremental model. We, therefore, concentrate on the effects of disaggregation on the multiple choice model while recognizing that disaggregating the convex combination model will provide equivalent results.

In order to disaggregate, the application must have an inherent decomposable structure. If in a general multi-commodity network flow problem, each commodity has a specific supply and demand node, the aggregated formulation given by (3.9)-(3.14) as a single commodity network flow problem is not valid. Therefore, we need to limit our discussion of aggregated formulations to networks where they are valid. In addition, for disaggregation to be applicable, the aggregated variables must possess a disaggregated form that can be bounded. As noted before, we can model a network problem with one source supplying many destinations, or visa versa, as an aggregated model using (3.9)-(3.14). Alternatively, we can define a commodity by its destination and then use the disaggregated formulation with $M^k = d(k)$, the demand at destination k. This disaggregation is also possible for the facility location problem.

Disaggregation might be possible at several levels. For example, in a multi-commodity network with many origins that each supply several destinations (as in the merge-in-transit problem discussed in Chapter 5), the aggregated formulation given by (3.9)-(3.14) is not valid, but we can write an aggregated formulation that defines a commodity by its origin and bounds the variable flow by the total possible demand provided by that origin. We can then consider a disaggregated model by defining a commodity by both origin and destination. These new variables are bounded by the particular demand for that origin-destination pair and, therefore, would provide a stronger formulation. We can even consider another level of disaggregation by a specific *path* through the network. This disaggregation, however, will be tighter only if

we can provide bounds on the path variables that are smaller than the bounds placed on the origin-destination variables.

3.8 An Application to the Network Loading Problem

3.8.1 The Undirected Case

As discussed in Section 3.3, we can view the network loading problem as a network flow problem with piecewise linear costs. For simplicity, we will consider the single facility network loading problem, although much of this discussion can be extended to the multiple-facility case. In the telecommunications version of this problem, we are given an undirected network $G = \{V, E\}$, with node set V and edge set E, and K, a set of commodities, each with a given origin $s(k)$, destination $t(k)$, and demand, $d(k)$. We need to install capacity on the arcs to simultaneously route the flow, but we can install capacities only in blocks of b units. We incur a positive fixed cost, f_{ij}, for each block of capacity we install on an arc, but no variable flow costs. The standard network loading formulation is as follows:

$$\text{Minimize} \sum_{(i,j)} f_{ij} y_{ij} \tag{3.29}$$

$$\text{subject to:} \quad \sum_{j:(i,j)\in E} x_{ij}^k - \sum_{j:(j,i)\in E} x_{ji}^k = \begin{cases} d(k) \text{ if } i = s(k) \\ -d(k) \text{ if } i = t(k) \\ 0 \text{ otherwise} \end{cases}$$

$$\sum_k (x_{ij}^k + x_{ji}^k) \leq b y_{ij}$$

$$x_{ij}^k \geq 0, \quad y_{ij} \in Z^+.$$

Alternatively, we can think of this problem as a network flow problem with a cost function like the one shown in Figure 3-3. In the single facility case, each segment will be of length b, so $b^s = sb$, and $f_{ij}^s = s f_{ij}$. We can consider both an aggregated formulation and a disaggregated formulation. The aggregated formulation will define a commodity by its origin. We use k to denote the original commodities and k' to denote these commodities defined by origin. Therefore, each commodity k' will have an origin $s(k')$, a vector of destinations $\overrightarrow{t}(k')$ and a demand $d(k', i)$, for each $i \in \overrightarrow{t}(k')$. The demand $d(k', i)$ is defined as $d(k', i) = d(\bar{k})$ if

$s(\overline{k}) = s(k')$ and $t(\overline{k}) = i$. We also define $D(k') = \sum_{k:s(k)=s(k')} d(k)$, the total demand of commodity k'. The formulation, then, is:

$$\text{Minimize} \sum_{s,(i,j)} f_{ij}^s y_{ij}^s \tag{3.30}$$

$$\text{subject to:} \sum_{j:(i,j)\in E} x_{ij}^{k'} - \sum_{j:(j,i)\in E} x_{ji}^{k'} = \begin{cases} D(k') \text{ if } i = s(k') \\ -d(k',i) \text{ if } i \in \overrightarrow{t}(k') \\ 0 \text{ otherwise} \end{cases}$$

$$(s-1)by_{ij}^s \leq \sum_{k'} (x_{ij}^{sk'} + x_{ji}^{sk'}) \leq sby_{ij}^s$$

$$\sum_s y_{ij}^s \leq 1$$

$$x_{ij}^{sk'} \geq 0, \quad y_{ij}^s \in \{0,1\}.$$

We can also disaggregate this model by destination; that is, we can define a commodity by both origin and destination. We can replace $D(k')$ by the individual demands, $d(k)$. The formulation becomes:

$$\text{Minimize} \sum_{s,(i,j)} f_{ij}^s y_{ij}^s \tag{3.31}$$

$$\text{subject to:} \sum_{j:(i,j)\in E} x_{ij}^{k} - \sum_{j:(j,i)\in E} x_{ji}^{k} = \begin{cases} d(k) \text{ if } i = s(k) \\ -d(k) \text{ if } i = t(k) \\ 0 \text{ otherwise} \end{cases}$$

$$(s-1)by_{ij}^s \leq \sum_{k} (x_{ij}^{sk} + x_{ji}^{sk}) \leq sby_{ij}^s$$

$$\sum_s y_{ij}^s \leq 1$$

$$x_{ij}^{sk} \geq 0, \quad y_{ij}^s \in \{0,1\}.$$

Proposition 5 *The solution to the linear programming relaxations of both the aggregated model (3.30) and the disaggregated model (3.31) have the same cost as the solution to the linear programming relaxation of the standard network loading formulation (3.29).*

Proof:

1. We first show that the models (3.30) and (3.31) are equivalent. Given a solution $(x_{ij}^{sk}, y_{ij}^{s})$ of model (3.31), if we let $x_{ij}^{sk'} = \sum_{k:s(k)=s(k')} x_{ij}^{sk}$, then $(x_{ij}^{sk'}, y_{ij}^{s})$ is a feasible solution to (3.30) with the same cost. In the solutions to each model, the summation of the x variables in the segment bound constraints are the same. For example, if the solution on arc (i,j) to the disaggregated model (3.31) contains 3 units of flow of a commodity originating at node A and destined for node B, and 4 units of a commodity originating at node A and destined for node C, we can solve the aggregated model (3.30) by setting the flow on arc (i,j) of the commodity originating at A to 7. Similarly, we can translate $(x_{ij}^{sk'}, y_{ij}^{s})$, a feasible solution of model (3.30), into $(x_{ij}^{sk}, y_{ij}^{s})$, a feasible solution of model (3.31) with the same cost by just splitting the flow of each origin-defined commodity into the flow of an origin/destination-defined commodity. With this one to one correspondence, we can conclude that these two formulations are equivalent.

2. We will now show that the linear relaxation of model (3.29) gives an optimal solution with the same cost as the optimal solution to the linear relaxation of model (3.31).

   (a). Consider an optimal solution $(x_{ij}^{sk}, y_{ij}^{s})$ to the linear relaxation of model (3.31). Let $x_{ij}^{k} = \sum_s x_{ij}^{sk}$ and $y_{ij} = \sum s y_{ij}^{s}$. Clearly, (x_{ij}^{k}, y_{ij}) satisfies the flow balance constraints of model (3.29). In addition, $b y_{ij} = b \sum_s s y_{ij}^{s} \geq b \sum_s \frac{\sum_k (x_{ij}^{sk} + x_{ji}^{sk})}{b} = \sum_k (x_{ij}^{k} + x_{ji}^{k})$. Therefore, (x_{ij}^{k}, y_{ij}) is feasible for the linear relaxation of model (3.29). Moreover, since $f_{ij}^{s} = s f_{ij}$, $\sum_{(i,j)} f_{ij} y_{ij} = \sum_{s,(i,j)} f_{ij}^{s} y_{ij}^{s}$, and so $(x_{ij}^{sk}, y_{ij}^{s})$ and (x_{ij}^{k}, y_{ij}) have the same objective value.

   (b). Consider an optimal solution (x_{ij}^{k}, y_{ij}) to the linear relaxation of model (3.29). For any edge $\{i,j\}$, let $\bar{s}$ equal the smallest integer satisfying the condition $0 \leq \frac{y_{ij}}{\bar{s}} \leq 1$. Now let $y_{ij}^{\bar{s}} = \frac{y_{ij}}{\bar{s}}$, $x_{ij}^{\bar{s}k} = x_{ij}^{k}$, and all other variables equal 0. Clearly, the vector $(x_{ij}^{sk}, y_{ij}^{s})$ satisfies the flow balance constraints and $\sum_s y_{ij}^{s} \leq 1$. In addition, for any edge $\{i,j\}$, $\sum_k x_{ij}^{\bar{s}k} + x_{ji}^{\bar{s}k} = \sum_k x_{ij}^{k} + x_{ji}^{k} \leq b y_{ij} = b\bar{s} y_{ij}^{\bar{s}}$ and for $s \neq \bar{s}$, $\sum_k x_{ij}^{sk} + x_{ji}^{sk} = 0$. Therefore, $(x_{ij}^{sk}, y_{ij}^{s})$ is feasible for the linear relaxation of model (3.31). Again, (x_{ij}^{k}, y_{ij}) and $(x_{ij}^{sk}, y_{ij}^{s})$ have the same objective value.

   (c). Since we can translate an optimal solution of model (3.31) into a solution of model

(3.29) with the same cost, and we can translate an optimal solution of model (3.29) into a solution of model (3.31) with the same cost, the linear programming relaxation of each formulation will provide optimal solutions with the same objective value.

3. Since models (3.30) and (3.31) are equivalent, and the linear programming relaxations of models (3.29) and (3.31) have optimal solutions with the same cost, we have established the proposition. ■

Note that we can solve the linear relaxations of each of these formulations by solving a shortest path problem for each origin/destination pair using arc costs $\frac{f_{ij}}{b}$.

We can now consider adding valid forcing constraints to the aggregated and disaggregated formulations. To the model (3.30) we can add the inequality

$$x_{ij}^{sk'} + x_{ji}^{sk'} \leq d(k')y_{ij}^{s} \tag{3.32}$$

and to the model (3.31) we can add the inequality

$$x_{ij}^{sk} + x_{ji}^{sk} \leq d(k)y_{ij}^{s}. \tag{3.33}$$

These forcing constraints are redundant for the integer program, but can strengthen the linear program. Therefore, to the extent that these forcing constraints tighten each model, both of these formulations can improve upon the standard formulation, and so we have the following result.

Proposition 6 *For the network loading problem, both the aggregated formulation, given by model (3.30) plus the inequality (3.32), and the disaggregated formulation, given by model (3.31) plus the inequality (3.33), are stronger formulations than the standard formulation (3.29).*

We will examine computational experiments on these formulations in the following chapter to see how much improvement can be expected.

3.8.2 Comparing The Directed and Undirected Cases

The directed case of the network loading problem arises in transportation applications when the capacity on the directed arc (i,j), in the form of a truck for instance, is not available to transport freight on the reverse arc (j,i). In this case we need to rewrite the network loading formulations to account for this difference. Assuming that whenever the arc set contains arc (i,j) it also contains the reverse arc (j,i), we now examine the standard formulation and the disaggregated formulation.

Using z_{ij} for the binary variables in the directed case, the standard formulation becomes:

$$\text{Minimize} \sum_{(i,j)} f_{ij} z_{ij} \tag{3.34}$$

$$\begin{aligned}
\text{subject to:} \quad \sum_{j:(i,j)\in E} x_{ij}^k - \sum_{j:(j,i)\in E} x_{ji}^k &= \begin{cases} d(k) \text{ if } i = s(k) \\ -d(k) \text{ if } i = t(k) \\ 0 \text{ otherwise} \end{cases} \\
\sum_k x_{ij}^k &\leq b z_{ij} \\
x_{ij}^k \geq 0, \quad z_{ij} &\in Z^+.
\end{aligned}$$

and the disaggregated formulation becomes:

$$\text{Minimize} \sum_{s,(i,j)} f_{ij}^s z_{ij}^s \tag{3.35}$$

$$\begin{aligned}
\text{subject to:} \quad \sum_{j:(i,j)\in E} x_{ij}^k - \sum_{j:(j,i)\in E} x_{ji}^k &= \begin{cases} d(k) \text{ if } i = s(k) \\ -d(k) \text{ if } i = t(k) \\ 0 \text{ otherwise} \end{cases} \\
(s-1) b z_{ij}^s \leq \sum_k x_{ij}^{sk} &\leq s b z_{ij}^s \\
\sum_s z_{ij}^s &\leq 1 \\
x_{ij}^{sk} &\leq d(k) z_{ij}^s \\
x_{ij}^{sk} \geq 0, \quad z_{ij}^s &\in \{0,1\}.
\end{aligned}$$

Propositions 5 and 6 hold for the directed case as well, and therefore, model (3.35) is

a stronger formulation than model (3.34). However, as we will verify with computational experiments in the following chapter, the relaxation gap for the standard formulation is smaller in the undirected case than it is in the directed case. To see this, note the following:

Proposition 7 *Assuming $f_{ij} = f_{ji}$, the solution to the linear programming relaxation of model (3.34) for the directed case has the same optimal objective value as the linear programming relaxation of model (3.29) for the undirected case.*

Proof:

1. Consider any feasible solution (x_{ij}^k, z_{ij}) to the linear relaxation of the directed formulation given by model (3.34). If we let $y_{ij} = z_{ij} + z_{ji}$, then (x_{ij}^k, y_{ij}) is a feasible solution to the linear relaxation of model (3.29) with the same objective value as (x_{ij}^k, z_{ij}). Since this transformation is possible for any feasible solution to the directed formulation, it is certainly possible for the optimal solution.

2. Consider (x_{ij}^k, y_{ij}), an optimal solution to the linear relaxation of the undirected model (3.29). Given that $f_{ij} > 0$, we know that at this optimal solution, $y_{ij} = \frac{\sum_k (x_{ij}^k + x_{ji}^k)}{b}$. If we let $z_{ij} = \frac{\sum_k x_{ij}^k}{b}$ for every directed arc (i,j), then (x_{ij}^k, z_{ij}) is a feasible solution of model (3.34). In addition, $y_{ij} = z_{ij} + z_{ji}$. Therefore, these two solutions have the same objective value.

3. Since we can translate an optimal solution from one problem into a solution to the other problem with the same objective value, the optimal solutions of both problems must have the same objective value. ■

We define the 'relaxation gap' of an instance as the difference, measured as a percentage, between the objective values of an optimal solution to the linear relaxation and an optimal integral solution. The previous proposition implies the following result.

Proposition 8 *Assuming $f_{ij} = f_{ji}$, the relaxation gap of the standard formulation for an undirected instance will be no larger than the relaxation gap of the standard formulation for a directed instance on the same network.*

Proof: The previous proposition tells us that the linear programming solutions for both models have the same optimal objective value. The optimal integral solution of the undirected problem, however, might have a smaller objective value than the optimal solution to the directed case. If (x_{ij}^k, z_{ij}) is an integer feasible solution to the directed model (3.34), then (x_{ij}^k, y_{ij}) with $y_{ij} = z_{ij} + z_{ji}$ is an integer feasible solution to the undirected model (3.29) with the same objective value. Therefore, the optimal objective value of the undirected model can be no larger than the optimal objective value of the directed model. Since the linear programming values are the same, but the integer programming value might be smaller for the undirected case, the gap between the linear program and the integer program solutions will be at least as large for the directed instance. ■

To develop more intuition, note that in order to have integral y or z variables, we might have to install excess capacity on each arc. In the directed case, we need to install this excess capacity on both arcs (i,j) and (j,i). In the undirected case, however, the flow in each direction can share capacity. Therefore, we might not need to install as much total excess capacity. For example, if the linear relaxation of the directed solution requires 2.3 units on arc (i,j) and 3.1 on arc (j,i), then the linear programming solutions to both the directed and undirected instance will require 5.4 units on the arc and the costs will be the same. Assuming the flow pattern is maintained, the integral solution for the directed formulation would require 3 units on (i,j) and 4 on (j,i) for a total of 7 units. The undirected formulation, however, would require only 6 units. As a result, the undirected model has a smaller relaxation gap.

We can provide a similar result if we consider the gap between the linear programming relaxations of the standard formulation and the disaggregated formulation. We express this difference as a percentage, which we call the 'improvement gap.'

Proposition 9 *Assuming $f_{ij} = f_{ji}$, the improvement gap for an undirected instance will be no larger than the improvement gap for a directed instance on the same network.*

Proof: As shown in Proposition 7, the linear programming relaxation of the standard formulation will provide solutions with the same optimal objective value for both the directed and undirected models. The linear programming relaxation of the disaggregated formulation, however, might yield a higher cost optimal solution for the directed case. The argument

is similar to the one provided in the proof of Proposition 8. If (x_{ij}^k, z_{ij}) is a feasible solution to the linear relaxation of the disaggregated directed model (3.35), then (x_{ij}^k, y_{ij}) with $y_{ij} = z_{ij} + z_{ji}$ is a feasible solution to the linear programming relaxation of the disaggregated undirected model, given by model (3.31) plus the inequality (3.33), with the same objective value. Therefore, the optimal objective value of the linear relaxation of the undirected model can be no larger than the optimal objective value of the linear relaxation of the directed model. Since the linear programming values for the directed and undirected standard formulations are the same, but the linear programming value of the disaggregated formulation might be smaller for the undirected case, the gap between the optimal objective value of the linear relaxation of the standard formulation and the optimal objective value of the linear relaxation of the disaggregated formulation will be at least as large for the directed instance. ■

To view this result intuitively, note that the forcing constraints in the disaggregated formulation force excess capacity to be added to the network when solving the linear relaxation, in the form of larger values for the y or z variables. It is this excess capacity that increases the total cost and 'tightens' the formulation. Again, the undirected network loading problem is able to share this capacity and therefore might not need to install as much excess. Therefore, the linear programming solution to the disaggregated formulation of the undirected case can be smaller than the linear programming solution to the directed case. As a result, the gap between the standard formulation and the disaggregated formulation is smaller for the undirected problem.

The practical implication of these results is that we can expect the use of the network flow formulations to be more effective for directed network loading problems, since both the relaxation gap for the standard formulation and the improvement we can expect from disaggregating is larger than in the undirected case. Our computational results in Chapter 4 will show that this is indeed the case.

Note that for either the directed or undirected case, the number of segments we need to consider in either of the network flow formulations equals the total flow on the network divided by b. Therefore, a large capacity corresponds to a small number of segments. Epstein [13] reports that the linear relaxation of the standard network loading formulation is tighter for lower capacities than higher capacities. One explanation for this result is the fact that if we

half the capacity of a particular instance, the linear programming solution doubles but the integer solution might not (it will double in the worst case). Therefore, the gap between the two will be smaller. To explain this result geometrically, we might examine two functions with the same convex lower envelope. Figure 3-7 shows two cost functions, one corresponding to a capacity of b and the other to a capacity of $b/2$. The linear programming relaxation of the standard formulation will approximate both by its lower convex envelope, shown by the dotted lines. It is clear from this figure that a function with more segments is better approximated by its lower convex envelope. We will see when we examine computational experiments in the next chapter that the standard formulation is indeed tighter when capacities are small. We will also see that in many cases, this property holds for the network flow formulations as well.

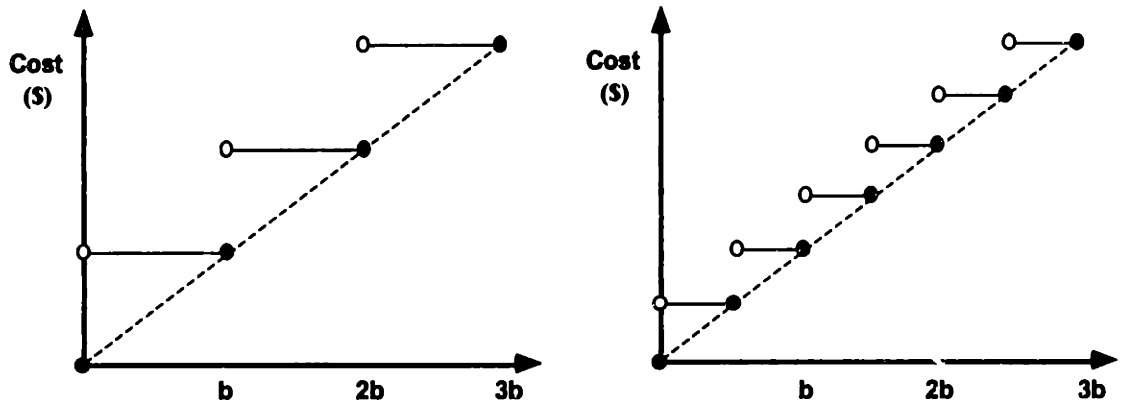

Figure 3-7: Network Loading: The Impact of More Segments on an Approximation

3.9 Effects of Disaggregation

We would like to better understand the implications of disaggregation. How do the forcing constraints improve the linear programming solution? How do the disaggregated and disaggregated/aggregated models approximate the cost function? We will first consider the disaggregated formulation and then examine the disaggregated/aggregated formulation.

3.9.1 The Disaggregated Model

Proposition 1 has already shown that the aggregated formulation approximates the actual cost with its convex lower envelope. For the disaggregated model, given by constraints (3.21)-(3.26) and (3.27), we develop a structural result and use the two-commodity case to help visualize and interpret it.

As an aid to understanding the effects of disaggregation, we will use linear programming techniques to establish the following result concerning extreme points.

Lemma 10 *Let* $Q = \{x \in \Re^n : Ax = b \text{ and } x \in \widehat{Q}\}$ *for some polytope* $\widehat{Q}$ *and system* $Ax = b$ *of* K *linear equalities. Then every extreme point of* Q *is a convex combination of at most* $K+1$ *extreme points of* $\widehat{Q}$.

Proof: Every extreme point of $\widehat{Q}$ is a convex combination of at most $n+1$ of its extreme points $x^1, x^2, ..., x^T$. Therefore, for any vector c, the linear program $min\{cx : x \in Q\}$ is equivalent to the "weighting" linear program $min\{\sum_{1 \le t \le T}(cx^t)\lambda^t : \sum_{1 \le t \le T}(Ax^t)\lambda^t = b, \} \sum_{1 \le t \le T} \lambda^t = 1,\ \lambda^t \ge 0 \text{ for all } t\}$in the sense that a point y solves the first linear program if and only if the point $y = \sum_{1 \le t \le T} x^t \lambda^t$ for some nonnegative weights λ^t summing to one, and this choice of weights solves the weighting linear program. But every extreme point of Q is the unique solution to the problem $min\{cx : x \in Q\}$ for some vector c. Therefore, we can represent every extreme point y of the polyhedron Q as $y = \sum_{1 \le t \le T} x^t \lambda^t$ for some optimal choice of the weights λ^t in the weighting linear program. But since this linear program has $K+1$ constraints, it has an optimal solution with at most $K+1$ positive weights λ^t. Therefore, every extreme point of the polyhedron Q is a convex combination of at most $K+1$ extreme points of the polytope $\widehat{Q}$. ■

For the disaggregated formulation, we can now establish the following result.

Proposition 11 *Let* K *denote the dimensionality of the disaggregation. Then the linear programming relaxation of the disaggregated formulation estimates a piecewise linear cost function with its lower convex envelope in* $K+1$ *dimensions.*

Proof: The proof of this is similar to the proof of Proposition 1.

1. Consider $\widehat{P}$=$\{(x,y) \in \Re^{SK+S} : b^{s-1}y^s \leq \sum_k x^{sk} \leq b^s y^s \ \forall s,\ x^{sk} \leq M^k y^s,\ \sum_s y^s \leq 1,\ x^{sk} \geq 0,\ y^s \geq 0\}$. Let K denote the number of commodities, i.e., the dimensionality of the disaggregation. First, we show that for all extreme points of $\widehat{P}$, $y^s > 0$ for at most one s and if one $y^s > 0$, then $y^s = 1$. We prove this result by contradiction. Assume the model has an extreme point (x,y) with $0 < y^r < 1$ and $0 < y^t < 1$. Without loss of generality, assume $r = 1$ and $t = 2$. The point, therefore, is $(x^{11},...,x^{1k},x^{21},...,x^{2k},0,...,0,y^1,y^2,0,...,0)$. We can express this point as $(1-y^1-y^2)(0,...,0)+y^1(\frac{x^{11}}{y^1},...,\frac{x^{1k}}{y^1},0,...,0,1,0,...,0)+y^2(0,...,0,\frac{x^{21}}{y^2},...,\frac{x^{2k}}{y^2},0,...,0,0,1,0...0)$. It is easy to see that these three points are in $\widehat{P}$ and, therefore, we have expressed (x,y) as a convex combination of three feasible points. It, therefore, cannot be an extreme point. The same argument can be extended to the case where three or more y variables are positive. Therefore, $y^s > 0$ for at most one s. When a single $y^s > 0$, we can express this point as a convex combination of two feasible points (by taking $y^r = y^s$ and $y^t = 0$ above), unless $y^s = 1$. Therefore, if at an extreme point of $\widehat{P}$, a single $y^s > 0$, it must equal one.

2. If at most one $y^s = 1$, and we denote it by $y^{\hat{s}}$, then we can substitute out the y variables and the polyhedra becomes $\widetilde{P} = \{x \in \Re^K : b^{\hat{s}-1} \leq \sum_k x^{\hat{s}k} \leq b^{\hat{s}},\ x^{\hat{s}k} \leq M^k,\ x^{\hat{s}k} \geq 0\}$. An extreme point must have at least K linearly independent constraints satisfied at equality. Only two kinds of points will satisfy this requirement. One is the set of points with $x^{\hat{s}k} \in \{0, M^k\}$ for all k. We refer to these points as corner points because they correspond to the corners of the feasible K-dimensional hypercube. The other is the set of points with $\sum_k x^{\hat{s}k} = b^{\hat{s}}$ or $b^{\hat{s}-1}$ and $x^{\hat{s}k} \in \{0, M^k\}$ for at least $K-1$ of the K terms. We refer to these as ridge points and they correspond to the extreme points of the K-dimensional face defined by $\sum_k x^{\hat{s}k} = b^{\hat{s}}$ (or $b^{\hat{s}-1}$) and $x^{\hat{s}k} \leq M^k$.

3. We have shown that the extreme points of $\widehat{P}$ consist of all corner and ridge points. Using Lemma 10, we can state that the extreme points of P=$\{(x,y) \in \widehat{P} : \sum_s x^{sk} = \hat{x}^k\}$ are convex combinations of at most $K+1$ of the extreme points of $\widehat{P}$.

4. As in the case of Proposition 1, every convex combination of the extreme points of $\widehat{P}$ is also feasible in P. Therefore, as we vary $\hat{x}^k$ and take the minimum, we will generate the

minimum cost convex combination of at most $K+1$ corner and ridge points. The corner and ridge points are the extreme points of the K-dimensional segments of the actual cost function and, therefore, we can express each point of the actual cost function as a convex combination of some subset of these points. Therefore, the minimum cost convex combination of all these ridge and corner points will define the lower convex envelope of the cost function in $K+1$ dimensions. ■

The Concave Case

To help understand the implications of Proposition 11, we will first examine the concave cost case. As shown in Figure 3-8, the approximation given by the aggregated formulation for any concave cost function is the line segment from the origin to the point of maximum flow.

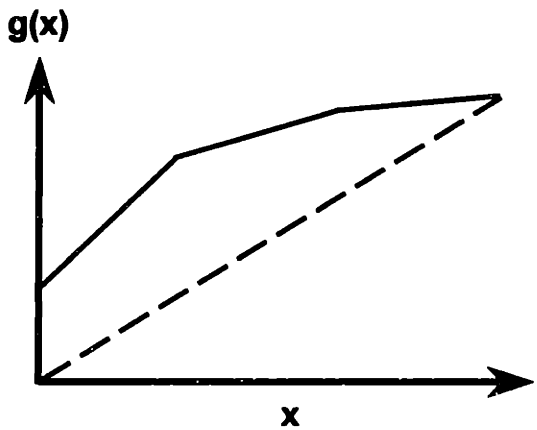

Figure 3-8: The Approximation of the Aggregated Formulation on a Concave Function

If we decompose the flow on an arc into K commodities, the total cost is now a concave function of the sum of K variables and can be represented in $K+1$ dimensions. We wish to understand the approximation that the linear programming relaxation of the disaggregated multiple choice formulation provides. Figure 3-9 illustrates the difference between the aggregated and disaggregated formulations with an example of the 2-commodity case. The aggregated approximation is a flat plane that lies below the actual cost function. The approximate and actual functions are equal at the origin and at the point of maximum flow, where

each commodity is at its maximum, but at no other points. The disaggregated approximation is formed by two planes that intersect along the diagonal. We can in fact write equations for these two planes. Each of these planes is equal to the actual cost function at three points: the origin, the point of maximum flow, and one of the two corner points where one commodity has zero flow and the other is at its maximum. This observation suggests that the disaggregated formulation provides the lower convex envelope of the actual concave cost function in both dimensions. Along the diagonal, the aggregated and disaggregated approximations are equal, but at all other points, the disaggregated approximation improves upon the aggregated one.

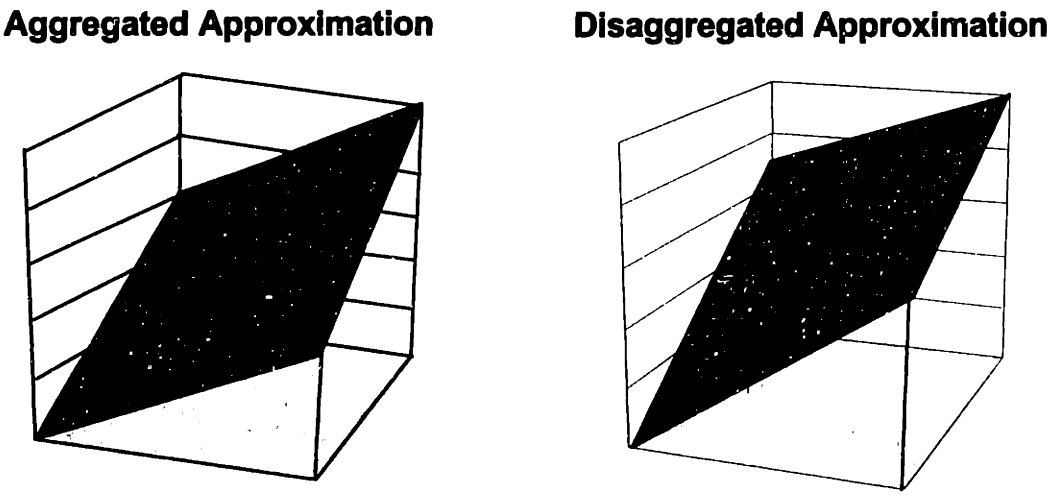

Figure 3-9: Concave Case: The Aggregated and Disaggregated Approximations

We can generalize these results for the K-commodity case. To determine the cost of a flow vector $(x^1, x^2, ..., x^K)$ in the disaggregated formulation, we first order the commodities so that $\frac{x^{(1)}}{d^{(1)}} \geq \frac{x^{(2)}}{d^{(2)}} \geq \frac{x^{(3)}}{d^{(3)}} \geq ... \geq \frac{x^{(K)}}{d^{(K)}}$. The disaggregated formulation then approximates the actual cost function, $g(x)$, with the function $\widehat{g}(x)$:

$$\widehat{g}(x^1, x^2, ..., x^K) = \sum_{i=1}^{K} a^{(i)} x^{(i)} \tag{3.36}$$

with

$$a^{(m)} = \frac{g(\sum_{i=1}^{m} d^{(i)}) - g(\sum_{i=1}^{m-1} d^{(i)})}{d^{(m)}}$$

Chan et. al [8] have derived equivalent equations independently. We can interpret $a^{(m)}$ as the slope of the segment between the cost of the sum of the first $m-1$ commodities and the cost of the first m commodities. For example, in the two commodity case, if the order of the commodities is 1 then 2, then $a^{(1)}$ is equal to the slope of segment 1 in the dotted line shown in Figure 3-10 and $a^{(2)}$ is equal to the slope of segment 2. The equation for the darker of the two triangles in Figure 3-10 is $\widehat{g}(x^1, x^2) = a^{(1)}x^1 + a^{(2)}x^2$.

Another way to view the improvement the disaggregated formulation provides is to generate the cost of the approximation by considering one commodity at a time. Figure 3-10 illustrates the impact for the two-commodity case. If we first flow one commodity up to its maximum and then begin to flow the second commodity, the resulting cost function is the dotted line in Figure 3-10. We can see that these costs improve upon the approximation of the aggregated formulation shown in Figure 3-9. In fact, we can interpret any flow vector one commodity at a time in this way, and see that for some permutation of the commodities, we will have a cost that improves upon the aggregated approximation (or at least as strong, as is the case along the diagonal). For instance, consider (x^1, x^2), the flow of two commodities. Suppose that $\frac{x^1}{d^1} \geq \frac{x^2}{d^2}$ so that (x^1, x^2) falls into the region under the darker triangle in Figure 3-10. Let m^1 be the slope of the first segment of the dotted line in Figure 3-10 and m^2 be the slope of the second segment. Then the cost of (x^1, x^2) for the disaggregated formulation will be $m^1x^1 + m^2x^2$, which is the same expression given by equation (3.36). In fact, $m^1 = a^{(1)}$ and $m^2 = a^{(2)}$. We can similarly extend this graphical interpretation to the k-commodity case.

These results provide insight into the structural effects of disaggregation. We can now begin to visualize the nature of the improvement resulting from disaggregating the formulation. Computational experience in the next section will quantify this improvement.

The Non-Concave Case

The geometric and graphic interpretations discussed for the concave case can be extended to non-concave cases, but we cannot provide closed-form expressions for the disaggregated approximation. We will, however, examine two 2-commodity examples to aid in the visualization of the impact of disaggregation.

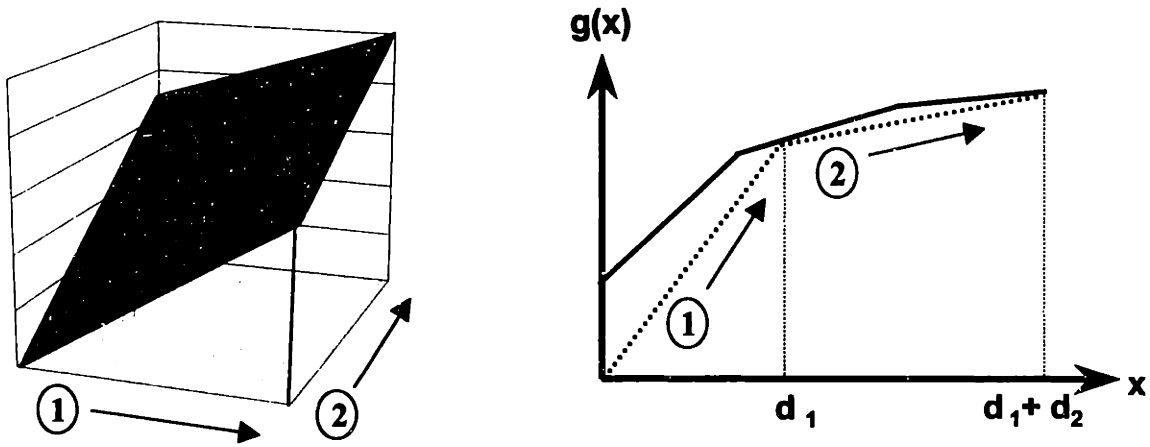

Figure 3-10: The Costs Resulting from Looking at Each Commodity Sequentially

Example 1: An LTL Cost Function The function we consider is shown in Figure 3-11. As in the concave case, the aggregated formulation approximates this function with a single plane that lies below it but is exact at the origin and at the top corner. The disaggregated formulation improves upon this approximation by taking the lower convex envelope of the actual cost function in each dimension. Figure 3-12 shows both the aggregated and disaggregated approximations. Again, the disaggregated approximation is composed of several planes that fold up around the diagonal. As in the concave case, the aggregated and disaggregated approximations are equal along the diagonal.

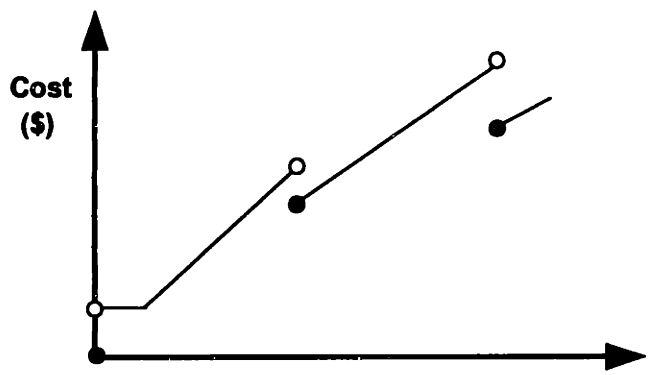

Figure 3-11: Example 1: An LTL Cost Function

We are also able to derive the formulas for each of these planes by examining the convex

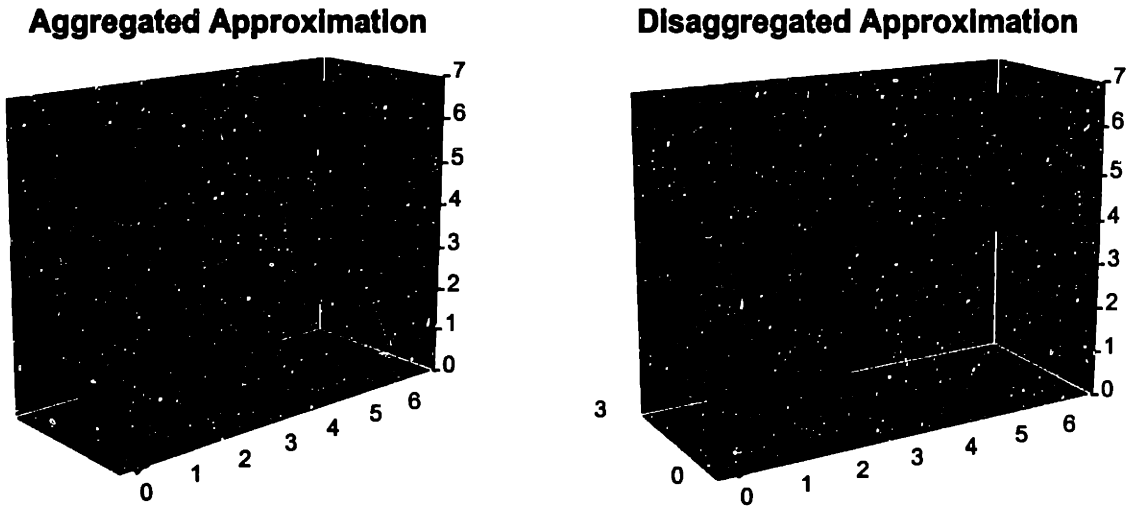

Figure 3-12: Example 1: The Aggregated and Disaggregated Approximations

envelope as we incrementally traverse each dimension. Figure 3-13 shows these two incremental envelopes, one as a dashed line and the other dotted. It is possible to derive the slopes of the four planes in Figure 3-12 from the slopes of the segments in these two dotted lines.

Example 2: A Truckload Cost Function In the second example, we consider a truckload cost function. This cost function is similar to the single-facility network loading problem discussed in Section 3.3. Figure 3-14 provides the example cost function, with the dashed line denoting its lower convex envelope. As we can see by the dashed line, the aggregated approximation will have two segments. Figure 3-15 illustrates this result along with the disaggregated approximation.

In this example the lower convex envelope in three dimensions varies very little from the lower convex envelope in two dimensions. The only improvement offered by the disaggregated formulation is in one corner of the function. Figure 3-16, which shows the two incremental convex lower envelopes, illustrates this result. For cost functions of this form, the convex envelope along each dimension will not vary substantially from the aggregated convex envelope. This observation might lead us to believe that we will not derive much improvement by disaggregating on cost structures of this form. As we will see, however, when we exam-

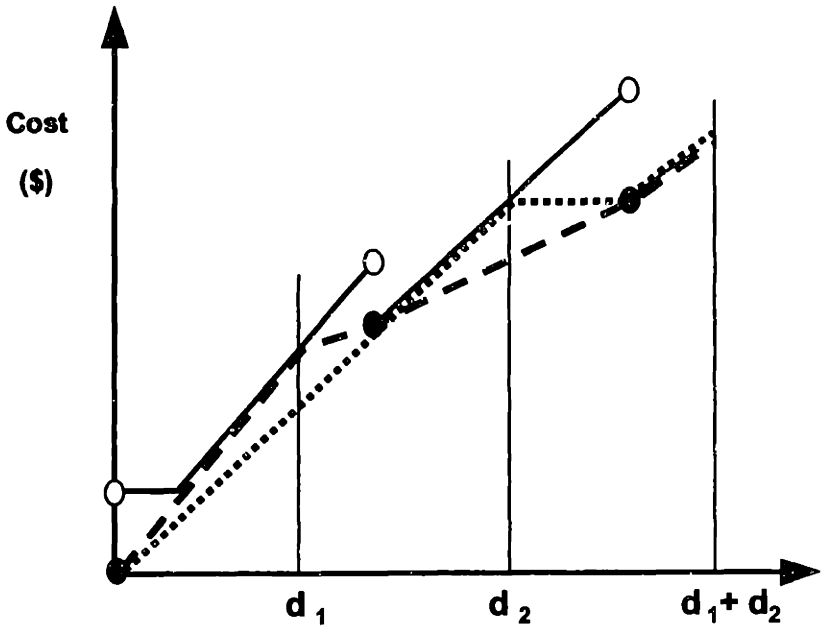

Figure 3-13: Example 1: The Incremental Convex Envelopes

ine computational results, this small improvement in the approximation can still have a large impact on the value of the linear programming relaxation.

3.9.2 Integrality of the Disaggregated Formulation

Although the disaggregated formulation provides a stronger (often quite strong) linear programming relaxation, it is not integral. We will see in the next section that the disaggregated formulation for networks with concave cost functions tends to provide integral solutions, but this is not always the case. We can, however, state the following result.

Proposition 12 *In a feasible solution to the linear programming relaxation of the disaggregated formulation in which each individual commoditiy flows on only one path through the network, this flow will not be split between segments and therefore the solution will be integral.*

Proof: Consider each arc of the network on which a subset K' of the commodities flows. For each $k \in K'$, $\sum_s x^{sk} = d^k$. Assume $b^{\hat{s}-1} \leq \sum_{k \in k'} d^k \leq b^{\hat{s}}$, i.e., the total flow falls into the sth segment.

1. For each $k \in K'$, $y^s \geq \frac{x^{sk}}{d^k} \Longrightarrow \sum_s y^s \geq \frac{\sum_s x^{sk}}{d^k} = 1$. But $\sum_s y^s \leq 1, \Longrightarrow \sum_s y^s = 1$.

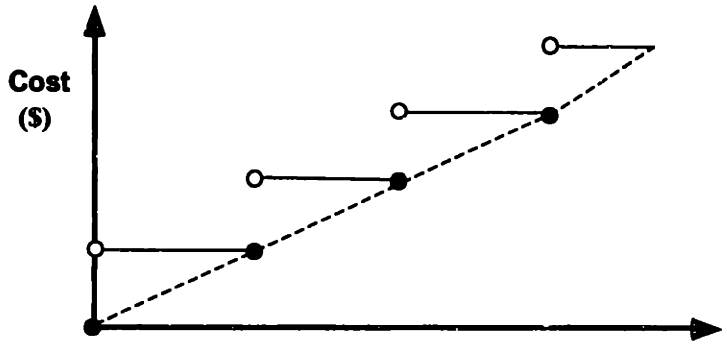

Figure 3-14: Example 2: A Truckload Cost Function

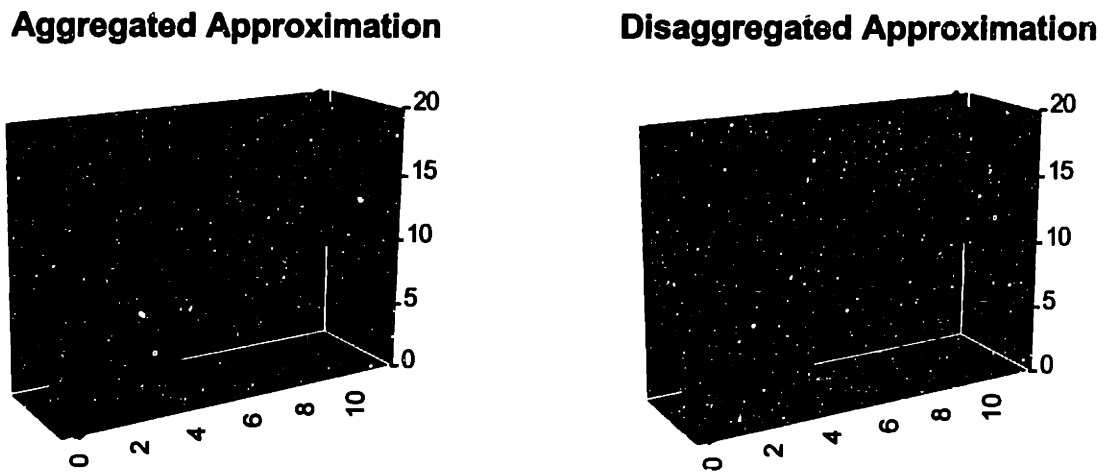

Figure 3-15: Example 2: The Aggregated and Disaggregated Approximations

2. Therefore, for each $k \in K'$, $y^s \geq \frac{x^{sk}}{d^k}$, $\sum_s y^s = 1$, and $\frac{x^{1k}}{d^k} + \frac{x^{2k}}{d^k} + ... + \frac{x^{sk}}{d^k} = 1 \Longrightarrow$ $y^s = \frac{x^{sk}}{d^k}\ \forall\ s$.

3. For all $s < \widehat{s}$, $\sum_{k \in k'} d^k > b^s$. Assume $y^s > 0$. Then $b^s y^s \geq \sum_{k \in k'} x^{sk} = y^s \sum_{k \in k'} d^k >$ $y^s b^s \implies$ contradiction $\Longrightarrow y^s = 0\ \forall\ s < \widehat{s}$.

   For all $s > \widehat{s}$, $\sum_{k \in k'} d^k < b^{s-1}$. Assume $y^s > 0$. Then $b^{s-1} y^s \leq \sum_{k \in k'} x^{sk} =$ $y^s \sum_{k \in k'} d^k < y^s b^{s-1} \implies$ contradiction $\Longrightarrow y^s = 0\ \forall\ s > \widehat{s}$.

4. Since $\sum_s y^s = 1$, $y^{\widehat{s}} = 1$. Therefore, $y^s \in \{0, 1\}\ \forall\ s$. ■

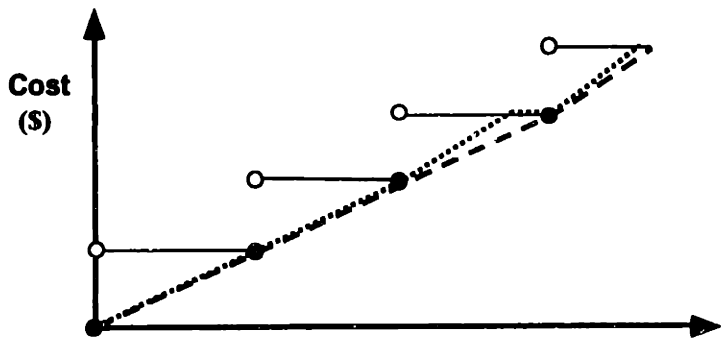

Figure 3-16: Example 2: The Incremental Convex Envelopes

Note that this proposition does not apply to the aggregated formulation. It also does not apply if a solution splits commodities between paths. Although concave cost functions encourage larger flows, it is not the case that its solution will never split flows between paths. There are, therefore, fractional cases of the disaggregated formulation, even with concave costs. Consider the facility location/assignment problem shown in Figure 3-17. In this problem, we need to assign each customer to a single facility. The cost of building each warehouse is 1. The transportation cost on the solid lines is 1/unit flow and on the dashed lines it is 2/unit flow. As discussed in Section 3.3, we can transform this problem into a single origin network flow problem with piecewise linear costs. In the transformed network, shown in Figure 3-18 with the associated arc costs, we need to route one unit from the dummy node to each of the three customer nodes. We can now model this problem with the disaggregated formulation. Because the cost functions have only a single segment and only one has a fixed cost, we need to define a binary variable only for each of the three arcs from the dummy node to the facilities. The flow variables have the form x_{ij}^k, where we define a "commodity" by the customer, a, b,

or c. The disaggregated formulation for this problem is:

$$\textit{Min.}\ \ x^b_{1b} + x^c_{1c} + x^a_{2a} + x^c_{2c} + x^a_{3a} + x^b_{3b} + 2x^a_{1a} + 2x^b_{2b} + 2x^c_{3c}$$

$$\begin{aligned}
\textit{Subject to}\ :\ & x^a_{D1} + x^a_{D2} + x^a_{D3} = 1 \qquad x^a_{1a} - x^a_{D1} = 0 \qquad x^a_{1a} + x^a_{2a} + x^a_{3a} = 1 \\
& x^b_{D1} + x^b_{D2} + x^b_{D3} = 1 \qquad x^b_{2b} - x^b_{D2} = 0 \qquad x^b_{1b} + x^b_{2b} + x^b_{3b} = 1 \\
& x^c_{D1} + x^c_{D2} + x^c_{D3} = 1 \qquad x^c_{3c} - x^c_{D3} = 0 \qquad x^c_{1c} + x^c_{2c} + x^c_{3c} = 1 \\
& 0 \le x^a_{D1} + x^b_{D1} + x^c_{D1} \le 3y_{D1} \\
& 0 \le x^a_{D2} + x^b_{D2} + x^c_{D2} \le 3y_{D2} \\
& 0 \le x^a_{D3} + x^b_{D3} + x^c_{D3} \le 3y_{D3} \\
& x^a_{D1} \le y_{D1} \qquad x^b_{D1} \le y_{D1} \qquad x^c_{D1} \le y_{D1} \\
& x^a_{D2} \le y_{D2} \qquad x^b_{D2} \le y_{D2} \qquad x^c_{D1} \le y_{D2} \\
& x^a_{D1} \le y_{D3} \qquad x^b_{D1} \le y_{D3} \qquad x^c_{D1} \le y_{D3} \\
& x^k_{ij} \ge 0,\ \ y_{ij} \in \{0,1\}.
\end{aligned}$$

The optimal solution to the linear relaxation of this formulation is: $y_{D1} = y_{D2} = y_{D3} = 0.5$, $x^b_{D1} = x^c_{D1} = x^a_{D2} = x^c_{D2} = x^a_{D3} = x^b_{D3} = 0.5$, and $x^b_{1b} = x^c_{1c} = x^a_{2a} = x^c_{2c} = x^a_{3a} = x^b_{3b} = 0.5$, which has a total cost of 4.5. Although the cost functions are concave, the optimal solution splits the flow of each "commodity" on two different paths, so Proposition 12 does not apply. The optimal solution to the linear programming relaxation of the disaggregated formulation of this instance is fractional. The optimal integer solution for this problem has a total cost of 5.

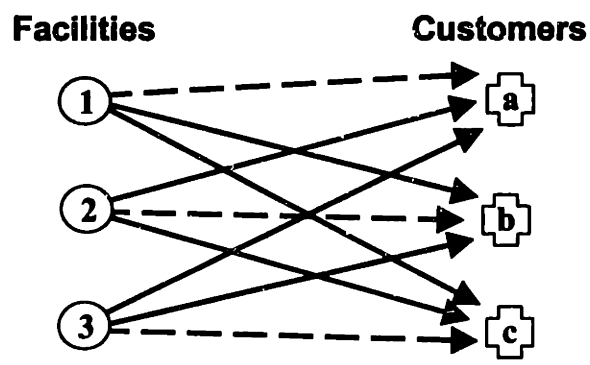

Figure 3-17: A [3,3] Facility Location Problem

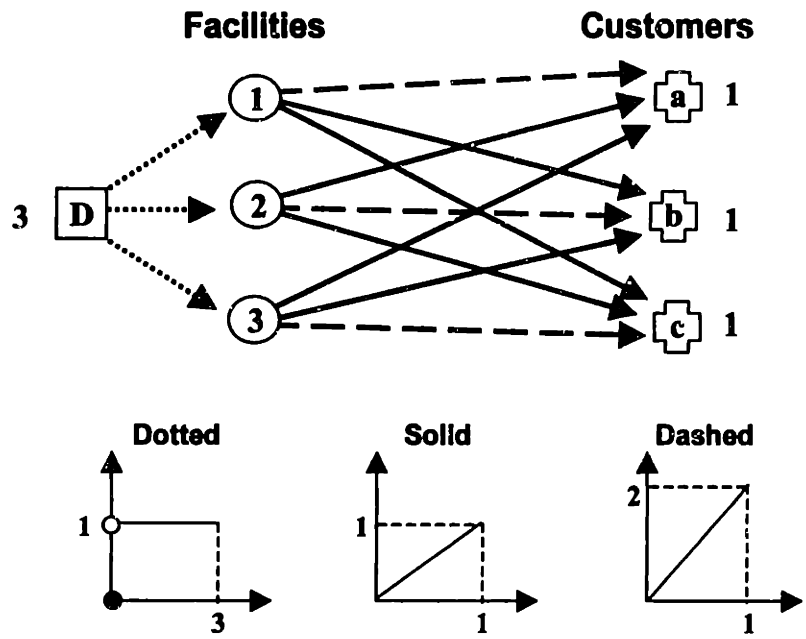

Figure 3-18: The Transformed [3,3] Facility Location Problem (with Arc Costs Below)

3.9.3 The Disaggregated/Aggregated Formulation

We already noted that the disaggregated/aggregated formulation is not, in general, as tight as the disaggregated formulation, but it can improve upon the aggregated formulation and it is not as large as the fully disaggregated model. We know the solution to the linear programming relaxation of this formulation will fall somewhere between the aggregated and disaggregated solutions, but we can also provide an analytical result on the effects of the aggregated forcing constraints (3.28). As a reminder, these forcing constraints are: $\sum_s x^{sk} \leq M^k \sum_s y^s$.

Proposition 13 *Let $(\widehat{x}^s, \widehat{y}^s)$ be the optimal solution to the linear relaxation of the aggregated formulation. This solution provides a lower bound for the integer formulation. Let (x^{sk}, y^s) be the equivalent solution with disaggregated flow variables, with $y^s = \widehat{y}^s$ and $\sum_k x^{sk} = \widehat{x}^s$. On each arc, the lower bound provided by the linear relaxation of the disaggregated/aggregated model increases by at most $f^1(1 - \sum_s \widehat{y}^s)$. In the case of concave costs, if S denotes the last segment, $F^k = \sum_s x^{sk}$, and $F = \sum_s \widehat{x}^s$, the per arc improvement is exactly equal to $f^1 \max_k\{0, \frac{F^k}{d^k} - \frac{F}{b^S}\}$,.*

Proof: Given $(\widehat{x}^s, \widehat{y}^s)$, and using the definition of the disaggregation, we can derive a feasible solution, (x^{sk}, y^s) to the formulation given by constraints (3.21)-(3.26), with $y^s = \widehat{y}^s$ and $\sum_k x^{sk} = \widehat{x}^s$. If this solution satisfies the inequality (3.28), then this solution satisfies all the constraints of the disaggregated/aggregated formulation and must be optimal. If not, then we need to change one or more of the y variables so that they satisfy this constraint. We can construct one feasible solution by increasing y^1 until we satisfy this constraint. In fact, we can increase y^1 without violating any other constraints because the model imposes no upper bound of the form $b^0 y^1 \leq \sum_k x^{1k}$ (since $b^0 = 0$). Therefore, if we increase y^1 until the solution satisfies inequality (3.28), then we have a feasible solution to the disaggregated/aggregated formulation. In the worst case, we increase y^1 until $\sum_s y^s = 1$; that is $y^1 = 1 - \sum_{s \neq 1} \widehat{y}^s$. We now have constructed a modified solution, (x^{sk}, y^s), that is feasible for the disaggregated/aggregated formulation. The cost of the original and new solutions on each arc are related by the following: $cost(x^{sk}, y^s) = cost(\widehat{x}^s, \widehat{y}^s) + f^1(y^1 - \widehat{y}^1) = cost(\widehat{x}^s, \widehat{y}^s) + f^1(1 - \sum_s \widehat{y}^s)$. In the case of concave costs, we know that for each arc with positive flow, the solution to the aggregated formulation is $\widehat{x}^S = F$ and $\widehat{y}^S = \frac{F}{b^S}$. Since $f^1 = \min_s\{f^s\}$, it is cheaper to increase y^1 than any of the other y's. Therefore, by letting $y^1 = \max_k\{0, \frac{F^k}{d^k} - \frac{F}{b^S}\}$, we satisfy the inequality (3.28) and have found the optimal solution to the disaggregated/aggregated formulation when we maintain the same arc flows. This solution increases the objective value by $f^1 \max_k\{0, \frac{F^k}{d^k} - \frac{F}{b^S}\}$. Therefore, since we are minimizing, the optimal linear programming objective function, and therefore the lower bound, increases by at most this amount. ■

We can interpret the concave case geometrically as we did in the previous section. Returning to the 2-commodity case and Figure 3-9, we note that along the diagonal, $\max_k\{0, \frac{F^k}{d^k} - \frac{F}{b^S}\} = 0$ and therefore the solutions to the linear relaxations of the aggregated and disaggregated/aggregated formulations have the same cost. Elsewhere, y^1 increases linearly along each dimension as F^k increases. Therefore, the approximation provided by the linear relaxation of the disaggregated/aggregated formulations is folded along the diagonal much like that of Figure 3-9, but with a fold that is not as high. Using the disaggregated/aggregated formulation, it is therefore possible to attain a portion of the improvement gained in the fully disaggregated formulation. In one important case, however, we will not see any improvement.

Corollary 14 *In the case of cost functions with no initial fixed charge (i.e., $f^1 = 0$), the*

aggregated/disaggregated formulation will not improve upon the aggregated formulation.

Proof: Proposition 13 shows that if $f^1 = 0$, we can increase y^1 without increasing the cost of the solution. Therefore, the disaggregated/aggregated formulation has a feasible solution with the same cost as the aggregated solution. Therefore, the disaggregated/aggregated formulation provides no improvement. ■

Using this result, we notice an interesting difference between the following two cases: (1) a cost function with an initial fixed cost of f^1, and (2) the same cost function with no initial fixed cost, but with a very steep and short initial segment whose endpoint is at f^1. These cost functions are approximately the same, and because the convex lower envelope of each is the same, the linear relaxations of the aggregated formulation with each cost functions will have the same optimal objective value. Similarly, the linear relaxation of the disaggregated formulations should also provide solutions with equal cost (the convex envelope in each dimension will also be the same for both cost functions). The disaggregated/aggregated formulations, however, will perform very differently. As a result of Corollary 14, the solution to the relaxation of the disaggregated/aggregated formulation for the second cost function with no initial fixed cost will be the same as the aggregated solution. As we will see when we examine computational results, however, the disaggregated/aggregated formulation for the first cost function might substantially improve upon the aggregated formulation. As this example shows, the performance of the disaggregated/aggregated formulation can vary greatly depending on how we represent the cost functions.

In situations when $f^1 \neq 0$, the disaggregated/aggregated formulation might be a good compromise. The formulation contains much fewer constraints than the fully disaggregated model and, therefore, will be easier to solve. In addition, particularly if the fixed costs are large, this formulation can be considerably tighter than the aggregated one. We will see in the following chapter how each of these disaggregated models performs on a variety of cost functions.

3.10 Deciding When and How to Disaggregate

The disaggregated formulation of the multiple choice formulation is quite a bit larger than its aggregated counterpart, both in the number of variables and the number of constraints. As we will see in subsequent sections, however, the disaggregated formulation can be substantially stronger than the aggregated formulation and, therefore, might be appropriate. Understanding the effects of disaggregation will help us understand how we can determine when to disaggregate.

There is one caveat to disaggregation. If we have one formulation with valid forcing constraints, we can often consider a further disaggregation, but this will not always improve the formulation, and might in fact weaken it. This can occur if we further disaggregate the flows, but cannot decrease the flow bounds effectively. For example, assume we are given flow variables x and the forcing constraints $x \leq My^s$. We can now consider the option of using disaggregated variables x^k (where $x = \sum_k x^k$) and forcing constraints $x^k \leq M^k y^s$. If $M^k \geq M$, the new forcing constraints are weaker than the old ones. By replacing the old ones, we weaken the formulation. In the situation where $M^k < M$, but $\sum M^k > M$, the tightest formulation is the one that includes both the aggregated and disaggregated forcing constraints.

As an example of this, consider the multi-commodity capacitated network flow problem. If we define a commodity by origin, and each origin has demand D^k, we can include forcing constraints with $M_{ij}^k = min(D^k, C_{ij})$, where C_{ij} is the capacity of arc (i,j). Now if we disaggregate commodities by destination, where each origin/destination pair has demand d^k, we can disaggregate the forcing constraints with $\overline{M}_{ij}^k = min(d^k, C_{ij})$. If we simply replace the aggregated forcing constraints with the disaggregated ones, we will weaken the formulation wherever $\overline{M}_{ij}^k = C_{ij}$. If $\overline{M}_{ij}^k = C_{ij}$, then $M_{ij}^k = C_{ij}$ and, therefore, we have not improved the bound on the constraints. In essence, we are replacing $\sum_k x^k \leq My^s$ with $x \leq My^s$ and the latter is weaker.

To avoid these problems and assure that we have a tight formulation, we should replace the aggregated forcing constraints with the disaggregated ones only when $\sum M^k \leq M$. If we cannot assure this inequality for each flow variable, but we still wish to disaggregate, we should retain the aggregated forcing constraints in the formulation so that we do not lose anything by disaggregating.

It might also be tempting to believe that we could further improve our formulation by disaggregating the commodities to ones with unit demands. For instance, a demand of d units between a specified origin and destination could be defined as d individual commodities, each with a demand of one. In our forcing constraints, we could then set $d^k = 1$. Therefore, in place of $x \leq dy^s$, we have $x^k \leq y^s$, which are a tighter set of constraints. However, this disaggregation will not improve our formulation. To establish this result, consider the following two sets of constraints.

$$\begin{aligned} \sum_{j:(i,j)\in E} x_{ij} - \sum_{j:(j,i)\in E} x_{ji} &= \begin{cases} d \text{ if } i = s \\ -d \text{ if } i = t \\ 0 \text{ otherwise} \end{cases} \\ x &\leq dy^s. \end{aligned} \tag{1}$$

$$\begin{aligned} \sum_{j:(i,j)\in E} x^k_{ij} - \sum_{j:(j,i)\in E} x^k_{ji} &= \begin{cases} 1 \text{ if } i = s \\ -1 \text{ if } i = t \\ 0 \text{ otherwise} \end{cases} \\ x^k &\leq y^s. \end{aligned} \tag{2}$$

The constraints in (1) model the flow of a single commodity, defined with a single origin $s(k)$ and destination $t(k)$, and a demand of d, and (2) represents this flow disaggregated into d commodities with unit demand. Note that if a solution $(\widehat{x}, \widehat{y})$ is optimal to (1), then the solution $(\frac{1}{d}\widehat{x}, \widehat{y})$ is feasible to (2) and has the same cost as (1). The issue here is that once a commodity is defined by an origin and destination, its flow in any feasible solution is on a set of paths between the origin and destination. If we decompose the demand of this commodity into two or more commodities, we can distribute the original flow proportionately so that we remain feasible in the disaggregated problem, but the total flow on the arcs does not change, and we therefore do not change the cost of the solution. Since we can always construct a feasible solution with the same cost, this disaggregation does not improve the linear programming relaxation of our problem. Graphically, we are able to construct a solution that lies on the diagonal of the function in Figure 3-9 (or any of the other functions similarly graphed), and along this diagonal, both the aggregated and disaggregated solutions are equivalent.

This same mechanism, however, does not operate when we disaggregate, for example, by destination. In this case, if we decompose a commodity traveling to two destinations, a feasible flow will contains arcs on which both of the new commodities do not flow, for example, the arcs leading into each destination. Therefore, we cannot proportionately distribute the flow as we did above, and still satisfy the flow balance constraints. In this case, the interaction of the flow balance constraints and the forcing constraints results in a stronger formulation. In other words, we are forced off the diagonal of Figure 3-9 and we obtain an improvement from the disaggregation.

These examples suggest that although disaggregation can tighten a formulation, we need to closely consider the circumstances under which we are disaggregating. We need to be sure that the forcing constraints do indeed tighten the formulation, and that we do not lose any strength by using them to replace the aggregated forcing constraints.

3.11 Summary

We started this chapter by examining three common approaches for modeling network flow problems with piecewise linear cost functions. Proposition 1 stated that these three formulations have equivalent linear programming relaxations and, in fact, each estimates the actual cost function with its lower convex envelope. We then examined a technique for disaggregating the variables by commodity. How we define a commodity is application specific, but in the examples we consider we define a commodity by its origin or destination. This disaggregation allows us to add forcing constraints to the multiple-choice formulation which, when appropriately used, can tighten the linear relaxation. We can apply this technique, for example, to the single origin or single destination network flow problem, the facility location problem with multiple capacities, and the network loading problem.

We are able to interpret this disaggregated formulation geometrically and analytically. Analytically, we are able to conclude that, as stated in Proposition 11, the disaggregated multiple-choice formulation approximates the actual cost function with its convex lower envelope in $K+1$ dimensions, with K dimensions corresponding to the flow of a commodity.

With this result, we examined the two-commodity disaggregation of concave, LTL and TL cost functions and gained a geometrical view of the impact of disaggregation on a single arc.

We will see how this impact plays out over an entire network in the next chapter.

Chapter 4

Computational Experience

In this chapter, we report on computational results for the standard network flow problem with piecewise linear costs, the facility location problem with multiple capacity options, and the network loading problem on an undirected network. We performed these computational experiments on a Dell PC with a 400 MHz processor running Linux and CPLEX 6.0.

We are primarily concerned with comparing the linear relaxations of the various formulations. To solve most of the linear relaxations, we used the dual simplex algorithm in CPLEX. We found that of the algorithms available in CPLEX, this one usually performed the best. CPLEX's network algorithm also performed well, but, was usually slightly slower than the dual simplex method. For some very hard problems, the barrier algorithm was quite effective, but for most problems it was inferior to the simplex method.

In addition to knowing how the formulations compare with each other, we also want to give a sense for how tight each of these formulations is as compared with the optimal mixed-integer solution. For small problems, we are able to use branch and bound to solve the mixed-integer problem for the true optimal solution. On larger problems, we used a rounding heuristic to find integer-feasible upper bounds on the optimal solution. This heuristic first solves the linear relaxation and examines the optimal values of the binary y variables. It then fixes all y variables that have values above a user-specified threshold to one. The upper bounds will be tighter for higher threshold values. We found that a threshold between 0.7 and 0.9 is usually best. Similarly, the heuristic fixes to zero all y variables whose value in the optimal linear relaxation solution is zero. With these variables fixed, it then performs branch and bound on

the resulting reduced problem. This reduced problem might not have a feasible mixed-integer solution, but if it does, this solution provides an upper bound. On problems where we could use branch and bound to solve for the optimal solution, we compared the heuristic solution with the optimal solution and found that this heuristic performs quite well.

For each problem instance, we provide the gap between the optimal linear programming objective value, LP, of each formulation and the known upper bound, UB. We calculate the gap as $[(UB - LP) \ / \ LP] * 100$. In most cases, we also provide the time, in CPU seconds, to solve the linear relaxations of the disaggregated formulations. For all these instances, the relaxation of the aggregated formulation solved in under a second, so we do not report these times for each problem.

4.1 Standard Network Flow Problems with Piecewise Linear Costs

To study the effects of disaggregation on network flow problem with piecewise linear costs, we conducted computational experiments on three networks and a variety of cost structures. Network 1, provided by Cominetti and Ortega [10], has 100 nodes, 195 arcs, and 70 commodities, with demands ranging from 2 to 100. Because this network is sparse, we also created a more dense network, Network 2, with 25 nodes, 200 arcs, and 20 supply nodes, whose supplies range from 15 to 95. Both these networks have a single commodity flowing from multiple origins to a single destination. This demand pattern allows us to define a commodity by its origin and to use the multiple choice formulation disaggregated by commodity, as discussed in Section 3.7. We can then compare the linear programming relaxations of the aggregated, disaggregated/aggregated (D/A), and disaggregated formulations.

The third network, Network 3, is a multi-commodity example with 40 nodes and 68 arcs in a grid pattern. The problem contains 6 commodities, each flowing from a single origin to multiple destinations. The demand for each origin-destination pair ranges from 5 to 30. On this network, the aggregated formulation given by (3.9)-(3.14) is not valid, since it does not discriminate between commodities. We can, however, consider several disaggregated formulations. We consider four formulations which we define by the variable definition and

by the form of the forcing constraints. We can choose to define a commodity by its origin or by its origin and destination. For each of these variable definitions, we can include forcing constraints of the forms of either (3.27) or (3.28). These two choices provide us with four formulations. This type of disaggregation is closely related to what we will see in the Merge-in-Transit application in Chapter 5.

4.1.1 Concave Cost Structures

We first compare the disaggregated model with the results of Cominetti and Ortega. They develop an algorithm for the capacitated network flow problem with concave piecewise linear costs. They formulate the problem using an aggregated model and then solve it with an algorithm that uses pre-processing and an LP-based branch-and-bound process that incorporates sensitivity analysis to improve the lower bounds. Their test problems used Network 1 (with 100 nodes, 195 arcs, and 70 commodities), and varied in the maximum number of segments per arc. Their arc costs were generated by taking the concave lower envelope of a function with n segments with randomly generated fixed and variable costs. Therefore, these problems might contain fewer than n segments per arc.

Because their test problems had a single destination node, we could model the problem with the disaggregated multiple choice formulation. Table 4.1 compares the results of their algorithm, which provided both a lower bound (LB) and an upper bound (UB), with the results of solving the linear relaxation (LP) of the disaggregated formulation. The "Int." designation indicates that the linear programming solution of the disaggregated formulation was integral and, therefore, optimal. The disaggregated formulations had roughly 20,000-30,000 variables and constraints. We should note that the times reported for the Cominetti and Ortega solutions are from runs on a Sun Sparc 10 machine with operating system Sun OS 4.1.3, not the Dell PC with Linux that we use to solve the linear relaxations.

The linear programming relaxation of the disaggregated formulation yields integral solutions for all five test problems. Moreover, solving the linear programming relaxation is significantly faster than the implementation of the algorithm developed by Cominetti and Ortega (even once we account for the possible differences in speed of the two machines). With these results as inspiration, we will now examine other computational results.

Problem	Max # Segs.	Cominetti & Ortega on the Aggregated Form. LB	UB	CPU (sec)	Solving the LP of the Disaggregated Form. LP Soln.		CPU (sec)
mtest0	1	128,110	130,012	1470	130,012	Int.	1
mtest1	2	90,239	93,026	1565	92,709	Int.	2
mtest2	3	96,345	100,501	1587	100,042	Int.	2
mtest3	4	63,291	67,587	1659	67,268	Int.	4
mtest4	5	48,958	52,420	1612	52,150	Int.	5

Table 4.1: Computational Comprison with Cominetti and Ortega Results

The results in Table 4.1 are not unusual. In all the single-destination instances we ran with concave costs, the disaggregated formulation solved with an integral solution. Tables 4.2 and 4.3 provide samples of such instances on Networks 1 and 2. In these problems we randomly generated a set of decreasing variable costs within the specified range for each arc. On Network 1, $b^s = \frac{s\sum_k d^k}{S}$ (where S is the number of segments), and so each segment has the same length, but on Network 2, $b^s = \frac{s^2\sum_k d^k}{S^2}$, so that the segment length increases as s increases. Given these variable costs, breakpoints, and f^1, the initial fixed charge, we can then calculate the appropriate fixed costs for the remaining segments so that the resulting cost function is concave.

On all three networks, when we include an initial fixed cost, we use a value of 1000. To give a sense for how large a fixed cost of 1000 is, relative to the variable costs, we note that for the instances on Network 1 with variable costs in [1,4], the optimal (integer) solution with an initial fixed cost of 1000 was, on average, 133% greater than the optimal (integer) solution with an initial fixed cost of 0. For the instances with variables costs in [1,10], this average difference was only 55%. On Network 2, the fixed costs dominated the variable costs even more. On instances with variable costs in [1,4], the difference was 414%, while it was 169% for instances with variable costs in [1,10]. Later we will see how the performance of the formulations are effected as we vary the value of the initial fixed cost.

.Because of the relatively small demands on Network 2, the instances with an initial fixed cost of 1000 provide results for a situation when fixed costs dominate the arc costs.

These results are a testimony to the strength of the disaggregated formulation. The computational results show not only that the disaggregated formulation solves with integral

#	Var. Cost		Init.	CPU (s)		% Gap from Optimal		
Segs	Min.	Max.	FC	D/A	Disagg	Agg.	D/A	Disagg.
3	1	4	0	7	5	16.0	16.0	0.0
3	1	4	1000	31	11	143.6	5.9	0.0
3	1	10	0	9	7	22.1	22.1	0.0
3	1	10	1000	22	8	79.4	12.7	0.0
4	1	4	0	12	8	24.5	24.5	0.0
4	1	4	1000	37	61	165.5	9.2	0.0
4	1	10	0	17	12	35.2	35.2	0.0
4	1	10	1000	57	63	102.5	20.5	0.0
8	1	6	0	65	394	57.2	57.2	0.0
8	1	6	1000	226	1570	191.2	21.3	0.0

Table 4.2: Computational Experience on Concave Cost Structures on Network 1

#	Var. Cost		Init.	CPU (s)		% Gap from Optimal		
Segs	Min.	Max.	FC	D/A	Disagg	Agg.	D/A	Disagg.
3	1	4	0	1	1	20.0	20.0	0.0
3	1	4	1000	1	3	336.0	3.4	0.0
3	1	10	0	1	1	30.5	30.5	0.0
3	1	10	1000	2	3	192.9	9.6	0.0
4	1	4	0	1	1	27.0	27.0	0.0
4	1	4	1000	3	49	354.5	4.2	0.0
4	1	10	0	1	1	39.5	39.5	0.0
4	1	10	1000	5	16	213.1	11.7	0.0
8	1	6	0	4	4	64.1	64.1	0.0
8	1	6	1000	11	201	437.1	8.3	0.0

Table 4.3: Computational Experience on Concave Cost Structures on Network 2

solutions, but also how weak the aggregated formulation is. Particularly when the problem has an initial fixed charge, the aggregated formulation is quite weak. We also see, however, that there is a price to be paid for the tighter formulation. As the number of segments grows, solving the disaggregated formulation becomes quite time consuming, while the aggregated formulation solves in under a second.

As a result of Corollary 14, the disaggregated/aggregated formulation does not improve upon the aggregated formulation when the initial fixed charge is zero. When the initial fixed charge is positive, however, this formulation improves significantly upon the aggregated formulation.

We now more closely consider the impact of the initial fixed cost. In Figure 4-1, we plot the relaxation gaps for each formulation versus the initial fixed cost, on Network 2. As we see,

the gap for the aggregated formulation increases as the fixed cost increases. This result is to be expected since the convex lower envelope of the function deteriorates as an approximation.

The relaxation gap for the D/A formulation, on the other hand, decreases as the fixed cost increases. If we analyze both the D/A LP solution and the optimal solution we can explain this phenomenon. For this network, the optimal solution for the D/A LP are the same for all values of the initial fixed cost. Similarly, the optimal integer solutions are also the same. Consider a single arc, and let (x^{sk}, y^s) be the optimal integer solution and $(\widehat{x}^{sk}, \widehat{y}^s)$ be the optimal LP solution to the D/A formulation. When we examine the solutions, we find that $\sum_s x^{sk} = \sum_s \widehat{x}^{sk}$; both solutions route the flow the same way and the solutions differ only by how the models allocate the flow among the segments. The difference between the cost of these two solutions equals $(c^s x^{sk} + f^s y^s) - (c^s \widehat{x}^{sk} + f^s \widehat{y}^s)$. Note that when we increase the initial fixed cost by Δ units, we increase each fixed cost by Δ as well (the entire cost function increases by Δ units). Clearly, if an arc has flow in the optimal integer solution, then $\sum_s y^s = 1$, and therefore an increase in the initial fixed cost by Δ, increases the optimal cost by Δ. We can also see that for an optimal solution to the linear program of the A/D formulation, if a particular commodity flows on a single path, then $\sum_s x_{ij}^{sk} = d^k$ and therefore $\sum_s y^s = 1$. Therefore, an increase in the initial fixed cost by Δ also increases the optimal cost of this LP by Δ. Indeed, for these instances, the linear programming solution sends each commodity on a single path. Since both the linear programming solution and the optimal solution are increased by the same amount, Δ, the difference between the cost of the two solutions remains constant as we vary the fixed cost. Therefore, when we calculate the gap using $[(Opt - LP) / LP]$, the numerator remains constant for all values of the initial fixed cost. The cost of the linear programming solution, however, increases as the fixed cost increases. Therefore, the relaxation gap for the D/A formulation decreases as the fixed cost increases.

This analysis relied on two observations particular to the solutions to this network and demand structure. The first was that the optimal integer solution and the linear programming solution of the D/A formulation did not change as the fixed cost changed, and in fact, the routing of the flows were the same in both solutions. We also used the fact that commodities were routed on single paths in the linear programming solution. This second fact is not uncommon for concave cost structures. We know the linear programming solution to

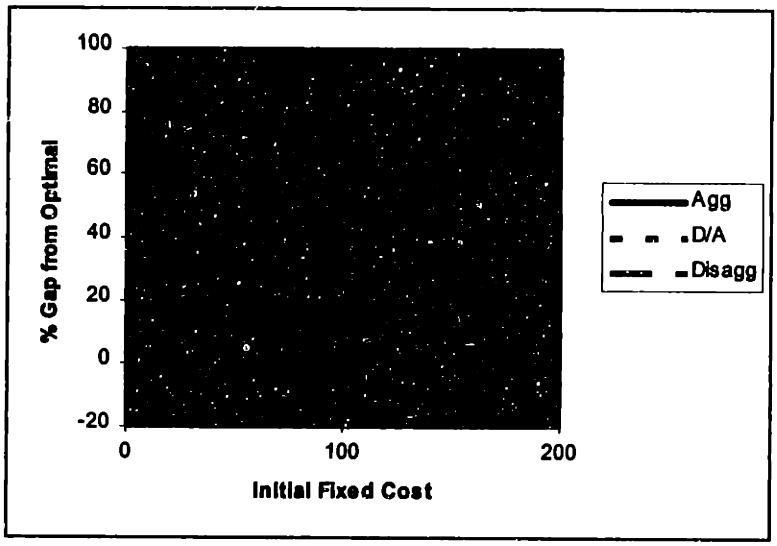

Figure 4-1: Optimality Gap on Network 2 as the Initial Fixed Cost is Varied

the aggregated formulation sends commodities on a single path because it simply linearizes the costs and uses the shortest path. In most cases, the optimal solution to the disaggregated/aggregated formulation maintains this same routing and simply increases y^1 to satisfy the forcing constraint (as shown in the proof of Proposition 13). As a result, the solution to the linear programming relaxation of the D/A formulation often satisfies this single path assumption.

Although other problems will not always exhibit these two features, they seem to be sufficiently present that the resulting trend often holds; that is, the relaxation gap for the disaggregated/aggregated formulation tends to decrease as the fixed cost increases. This plays an important role as we consider which formulation is best under varying circumstances.

The analysis thus far has examined concave cost structures on two single-destination networks. We will now examine the computational results on Network 3, a multi-commodity network. As mentioned, these results compare four models for the multi-commodity problem. We refer to the model by two letters. The first (A or D) denotes whether the variables are aggregated or disaggregated. Aggregated variables define a commodity by origin, while disaggregated variables define a commodity by both origin and destination. The second letter (also A or D) denotes whether the forcing constraints are aggregated or disaggregated, i.e., of

the form of (3.27) or (3.28). Table 4.4 provides a summary of the results. We do not include the solution times for the linear programming relaxations because all but one solved in under a second.

For instances with an initial fixed cost of 1000 on this network, the fixed costs dominated the variable costs overwhelmingly. The average difference between the optimal integer solutions with and without the initial fixed cost was 925% on instances with variables costs in [1,4] and 386% on instances with variable costs in [1,10].

Notice that when the problem has no initial fixed cost, the solutions to A/A and D/A are the same. The explanation for this is similar to the reasoning underlying Corollary 14. From a solution to the linear programming relaxation of A/A, we can derive a feasible solution to the linear programming relaxation of D/A by increasing y^1. When $f^1 = 0$, this solution will have the same cost as the solution to A/A. This property holds for all cost structures.

We also see that all but the D/A formulation have larger gaps when the problem has an initial fixed charge, but the fully disaggregated model is still quite tight and improves significantly upon the aggregated formulation. When the problem has no initial fixed cost, the solution to the fully disaggregated formulation is still integral, but we also notice a substantial improvement from just disaggregating the forcing constraints of the model with aggregated variables (i.e., comparing A/A and A/D). When the problem has an initial fixed cost, we do not obtain a significant improvement from just disaggregating the forcing constraints, but we realize a substantial improvement when we disaggregate the variables, even if we use aggregated forcing constraints. In summary, with concave cost structures, the fully disaggregated formulation is very tight. If, however, we want to use a smaller formulation, we should use a formulation with aggregated variables and disaggregated forcing constraints whenever the problem has no initial fixed charge. When the problem has an initial fixed charge, these results suggest that a formulation with disaggregated variables and aggregated constraints is most appropriate. As we will see, however, this can depend on the magnitude of the fixed costs.

One final observation from the data on all three networks is worth noting. As the number of segments increases, the gap between the aggregated and disaggregated formulations increases for cost structures with no initial fixed cost. This is because the way the cost functions were defined, a function with more segments was 'more concave.' The more concave the cost

#	Var. Cost		Init.	% Gap from Optimal			
Segs	Min.	Max.	FC	A/A	A/D	D/A	D/D
3	1	4	0	22.3	4.5	22.3	0.0
3	1	4	1000	73.0	69.1	2.8	0.9
3	1	10	0	31.6	6.0	31.6	0.0
3	1	10	1000	69.0	59.2	6.8	1.4
4	1	4	0	28.0	7.2	28.0	0.0
4	1	4	1000	73.3	69.3	2.4	0.4
4	1	10	0	42.1	10.0	42.1	0.0
4	1	10	1000	71.2	60.5	6.6	0.7
8	1	6	0	62.6	13.1	62.6	0.0
8	1	6	1000	77.8	69.2	4.7	0.0

Table 4.4: Computational Experience on Concave Cost Structures on Network 3

function is, the more improvement we gain from disaggregation.

We can also gain insight into these formulations by examining the trend in the gaps as we change the demand and cost structure. In Figures 4-2 – 4-5, we again consider a multi-commodity network with demand from several origins serving several destinations each (similar to Network 3).

In Figure 4-2, we see how the gaps change as we vary the magnitude of the initial fixed charge. We do not change the variable costs or the breakpoints; we just increase the fixed charge on each segment by a constant. As the fixed cost increases, the gaps for A/A and A/D increase. Although the relaxation gap for D/D was zero for each value of the fixed cost, for problems with smaller demands, the relaxation gap for this formulation also increased as the fixed cost increased. These increasing gaps are due to the fact that as the initial fixed cost increases, the approximation provided by the lower convex envelope deteriorates. While the gaps for the other formulations increase, however, we again see the gap for the D/A formulation decreasing as we did in Figure 4-1. As a result, for small fixed costs, the A/D formulation was superior to the D/A formulation, but for large fixed costs, the D/A formulation was tighter than the A/D formulation.

We might ask why the gap for the A/A formulation behaves differently than the gap for the D/A formulation, since both have the same constraint structure. The rationale goes back to the discussion of why the relaxation gap for the D/A formulation decreases. Recall that this result relied on an assumption that in the linear programming solution, each commodity

flowed on a single path so that $\sum_s x_{ij}^{sk} = d^k$ and, therefore, $\sum_s y^s = 1$. For this result to hold for the A/A formulation, we would need $\sum_s x_{ij}^{sk} = D^k$. Clearly, this cannot happen on the arcs into each destination, since each one requires only a portion of D^k. Moreover, since the destinations for each origin are likely to be spread throughout the network, we would not expect the flow to multiple destinations to "stick together" as they flow through the network. Therefore, we should not expect the linear programming solution to A/A to satisfy the property $\sum_s x_{ij}^{sk} = D^k$ on many arcs. As a result, the relaxation gap for A/A does not decrease as the fixed cost increases, as it did for D/A.

In Figure 4-3 we see how the gaps change as we vary the demand of each commodity. For these results, we vary d, a multiplier on the demand of each commodity. We then measure the gap between each formulation and the optimal solution. When the cost functions do not include an initial fixed cost, the gaps are constant for all d. When we introduce an initial fixed cost, however, we see the trends presented in Figure 4-3. Notice that these results follow the results shown in Figure 4-2, but in reverse. When generating these instances, we keep the fixed and variable costs constant as we vary the magnitude of the demand, but increase the breakpoints to keep the same number of segments. When the demand is small, then, the fixed cost represents a large portion of the total cost, but as the demand increases, the role of the fixed cost diminishes. As a result, increasing demand has the same effect as decreasing the initial fixed cost.

This figure, however, also provides us a clearer picture of the trade-off between disaggregating variables versus disaggregating forcing constraints. We see that as demand increases (or as the fixed cost decreases), the gain from disaggregating the variables decreases (i.e., the difference in the gaps between A/A and D/A and the difference between A/D and D/D decreases), while the value of disaggregating the forcing constraints increases (i.e., the difference in the gaps between A/A and A/D and the difference between D/A and D/D increases). This same trend is present in Figure 4-2, but it is not quite as clear that the solutions to A/A and A/D are approaching one another. This is an important consideration when choosing a formulation. In determining which formulation is the most appropriate, it is clear that we should consider the relative magnitude of the fixed costs.

In Figures 4-4 and 4-5, we vary k, the number of destinations that each origin serves. In this

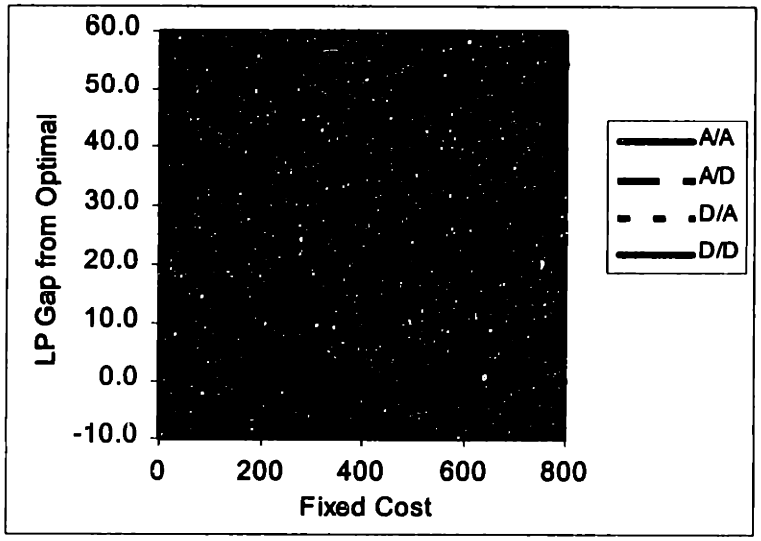

Figure 4-2: % Gap from Optimal as the Magnitude of the Fixed Cost Changes

case, we keep the expected supply out of each origin the same. Therefore, as we increase the number of destinations served, we decrease the expected demand for each origin/destination pair. In Figure 4-4 we use a concave cost function with no initial fixed cost. As expected for this case, the solutions to A/A and D/A are the same. In fact, the forcing constraints in A/A do not tighten the formulation. Therefore, this solution is the same as the solution to the formulation without any forcing constraints.

One thing we notice is that as we increase from one destination per origin to two, the A/A and D/A gap increases relatively significantly, but then stabilizes as we further increase the number of destination/origin pairs. Recall that the forcing constraints do not strengthen these formulations at all, so we can ignore them. We can then see that the reason for this initial jump is that as we move from one destination per origin to two, the total flow of the linear programming solution on any given arc tends to decrease significantly – in fact, often by a factor of two. When the total flow on an arc decreases, the segment bound constraints result in y variables with smaller fractions, and therefore weaker overall solutions. When k increases beyond two, however, there are more interactions between commodities and the total flow on a given arc stabilizes, resulting in more stable gaps.

We also see in Figure 4-4 that the A/D formulation also makes this initial jump, but then

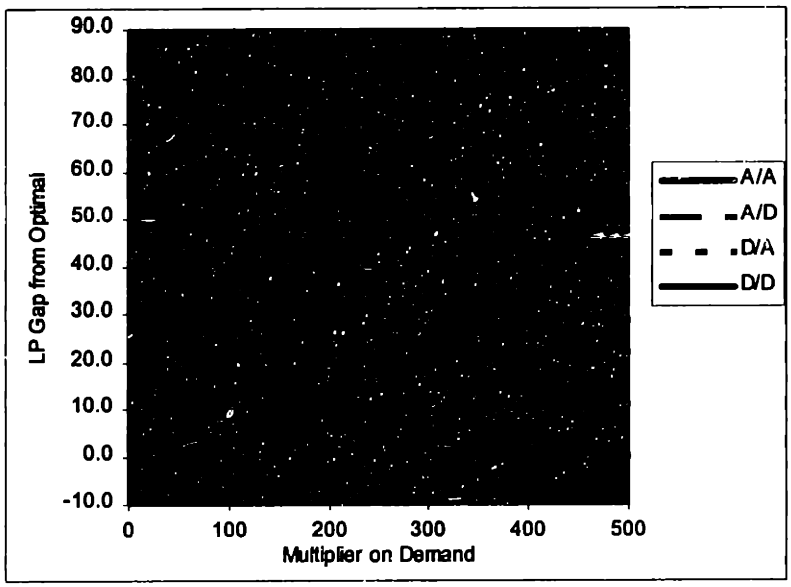

Figure 4-3: % Gap from Optimal as the Magnitude of the Demand Changes

continues to increase as k increases. Although the total flow on an arc might stabilize, the flow of each commodity continues its decreasing trend. Therefore, the forcing constraints in A/D, $x^{sk} \leq D^k y^s$, continue to produce smaller fractional y variables, and therefore, solutions with larger gaps.

Consider now Figure 4-5 where we again vary the number of destinations per origin, but this time include an initial fixed charge in the cost functions. The fixed charge increases the overall magnitude of the gaps, and there is also a difference between the solutions to A/A and D/A. However, these are not the only contrasts to Figure 4-4. We see that the gaps for A/A, A/D, and D/D increase as k increases, without any stabilizing. This is the reverse effect of what we saw in Figure 4-3. As k increases, the demand of the individual commodities decreases and, therefore, the gaps increase. The gap for D/A waffles, but remains fairly constant. As expected, the difference between A/A and D/A and the difference between A/D and D/D increase. This outcome reflects the intuitive notion that as the number of commodities increase, the gains from disaggregating by commodity increase.

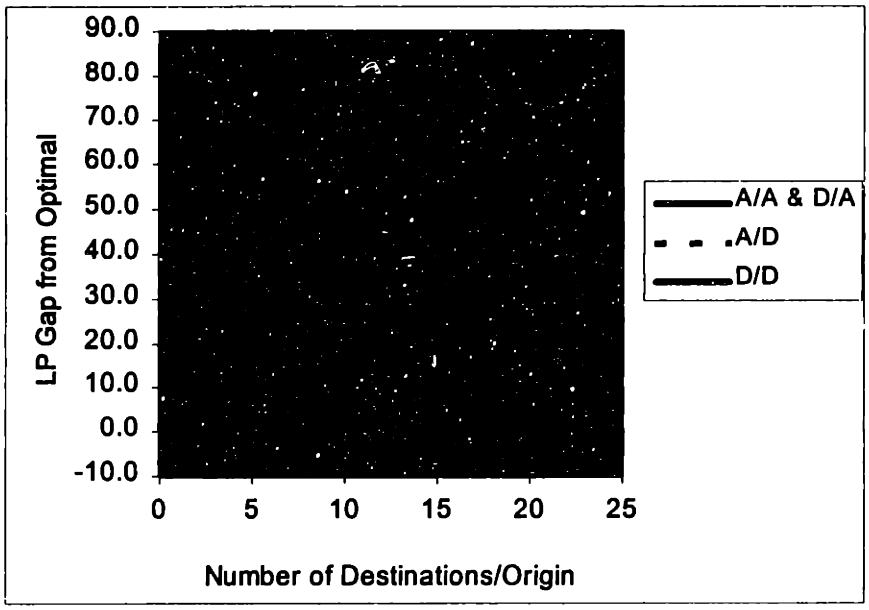

Figure 4-4: % Gap from Optimal as the Number of Destinations/Origin is Varied, with no Initial Fixed Cost

4.1.2 Non-Concave Cost Structures

When the cost functions are not concave, we find larger relaxation gaps for the disaggregated formulation, but we still see a remarkable improvement between the aggregated and disaggregated formulations. In the following computational runs, we have randomly (and uniformly) generated the costs within the specified range and the network and demand structure are again given by the previous specifications for Network 1, Network 2, and Network 3.

Tables 4.5, 4.6 and 4.7 show results for non-concave cost functions with fixed and variable costs for each segment generated randomly (uniformly) and independently within the ranges given for each instance. We can think of these as truly random piecewise linear cost functions, with no structure. We generated the upper bounds in Table 4.5 heuristically. In this table, as well as all others presented, the asterisk denotes a problem where the upper bound is, indeed, the optimal solution. The upper bounds shown in Tables 4.6 and 4.7 are the known optimal solutions.

We again notice that the disaggregated linear program solution is quite tight. When the instances are large it is more difficult to find good upper bounds, but for many of the instances

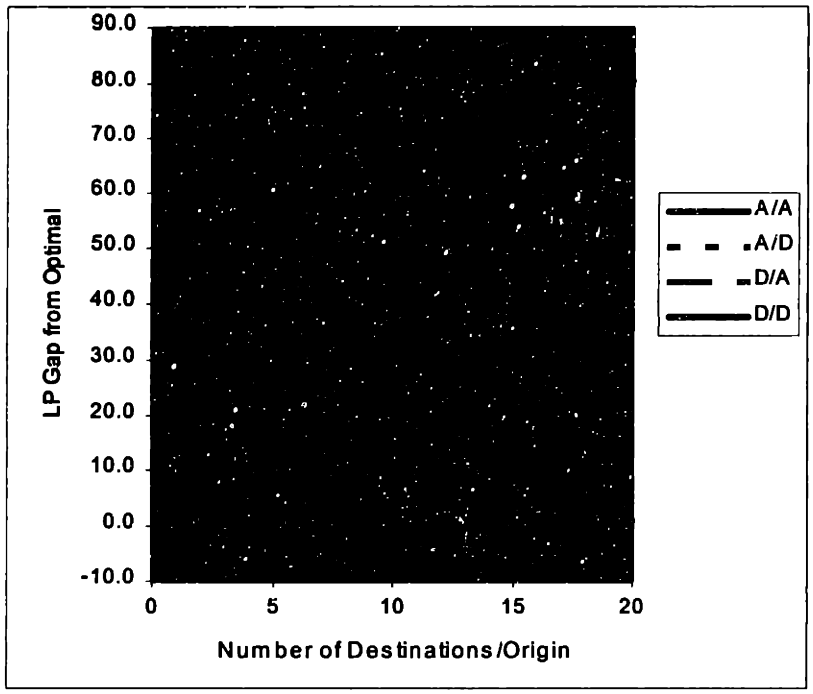

Figure 4-5: % Gap from Optimal as the Number of Destinations/Origin is Varied, with an Initial Fixed Cost

we see that the linear relaxation of the disaggregated formulation is very close to the upper bound. We also notice that the relaxation gaps for the disaggregated formulation tend to be slightly larger for Network 2 than for Network 1. One possible explanation of this difference uses Proposition 12. Recall that Network 1 is a less dense network, as measured by the ratio of the number of arcs to the number of nodes. As a result, the number of paths between any two nodes is likely to be smaller for Network 1 than for Network 2. Proposition 12 states that when commodities flow on a single path through the network, the solution to the linear relaxation is integral. The proof showed that on each arc, the linear programming solution solves with integral y variables. In a less dense network, commodities are more likely to flow on a single path because there are fewer paths from which to choose. As a result, more arcs will solve with integral y variables and the overall gap is expected to be smaller.

On Network 3, we have two instances where the gap between the fully disaggregated and optimal solutions is greater than 10%. This provides evidence that the disaggregated formulation is, on occasion, not very tight. Nonetheless, the disaggregated formulation still improves

#	Var. Cost		Fixed Cost		Time(s)		% Gap from UB			
Segs	Min	Max	Min	Max	D/A	Disagg.	Agg.	D/A	Disagg.	
3	1	4	0	0	3	108	28.0	28.0	0.0	*
3	1	4	0	100	4	126	37.5	25.3	0.1	*
3	1	10	0	0	3	133	42.9	42.9	0.0	*
3	1	10	0	100	4	136	48.2	40.4	0.1	*
4	1	4	0	0	3	424	33.5	33.5	1.4	
4	1	4	0	100	5	589	42.4	30.5	1.4	
4	1	10	0	0	4	1435	57.1	57.1	3.6	
4	1	10	0	100	6	1130	61.8	53.8	3.4	
5	1	4	0	0	4	371	42.9	42.9	2.0	
5	1	4	0	100	6	427	57.3	41.9	4.3	
5	1	10	0	0	6	945	77.5	77.5	4.4	
5	1	10	0	100	7	1578	86.4	75.0	5.9	

Table 4.5: Computational Experience on Non-Concave Cost Structures Network 1

#	Var. Cost		Fixed Cost		CPU (s)		% Gap from UB			
Segs	Min	Max	Min	Max	D/A	Disagg.	Agg.	D/A	Disagg.	
3	1	4	0	0	1	2	25.8	25.8	0.7	*
3	1	4	0	100	1	6	46.7	23.1	1.7	*
3	1	10	0	0	1	1	34.4	34.4	5.1	*
3	1	10	0	100	1	8	51.3	35.3	6.3	*
4	1	4	0	0	2	7	15.5	15.5	5.2	*
4	1	4	0	100	2	15	39.5	66.1	6.6	*
4	1	10	0	0	2	15	13.1	13.1	4.3	*
4	1	10	0	100	2	15	27.1	17.5	6.4	*
5	1	4	0	0	2	15	29.4	29.4	1.5	*
5	1	4	0	100	2	18	60.2	31.8	4.2	*
5	1	10	0	0	2	17	52.5	52.5	6.9	*
5	1	10	0	100	2	22	71.5	45.3	6.6	*

Table 4.6: Computational Experience on Non-Concave Cost Structures on Network 2

#	Var. Cost		Fixed Cost		CPU (s)	% Gap from UB				
Segs	Min	Max	Min	Max	D/D	A/A	A/D	D/A	D/D	
3	1	4	0	0	<1	24.6	12.9	24.6	0.5	*
3	1	4	0	100	6	102.1	71.3	40.3	13.3	*
3	1	10	0	0	<1	37.8	19.9	37.8	1.3	*
3	1	10	0	100	3	80.0	54.1	31.0	7.0	*
4	1	4	0	0	1	34.3	8.8	34.3	1.0	*
4	1	4	0	100	6	108.7	57.2	36.9	6.9	*
4	1	10	0	0	3	55.4	15.6	55.4	2.7	*
4	1	10	0	100	6	97.9	49.9	37.0	6.4	*
5	1	4	0	0	2	30.3	16.3	30.3	0.8	*
5	1	4	0	100	20	151.8	92.1	46.9	12.5	*
5	1	10	0	0	3	50.3	25.7	50.3	2.1	*
5	1	10	0	100	15	120.5	76.8	44.5	8.1	*

Table 4.7: Computational Experience on Non-Concave Cost Structures on Network 3

significantly upon the aggregated formulations. We also see that once again, A/D is tighter than D/A when there is no initial fixed cost, but D/A generally becomes tighter when we include an initial fixed charge.

Tables 4.8, 4.9, and 4.10 display results for LTL-like cost functions, which take the form shown in Figure 5-4 in Chapter 5. These functions contain no fixed costs on any of the segments and the variable costs decrease as the segment number increases. As we did for the concave cost instances, we assign breakpoints for each arc such that on Network 1 the segment lengths are constant but on Networks 2 and 3 the initial segments on any arc are shorter than subsequent ones. As a result of Corollary 14, the disaggregated/aggregated formulation will not improve upon the aggregated formulation. Therefore, we do not provide results for this formulation.

Like the results for the other cost structures, we notice large gaps between the aggregated and disaggregated solutions, and small gaps between the disaggregated solution and the known upper bound. Although not as small as for the concave case, the gaps between the disaggregated solutions and the known upper bound are within 1% for all instances on all three networks. These results are tighter than those presented for random non-concave functions. We also see again that the gap between the aggregated and disaggregated solutions increase as the number of segments increase. This is because the cost functions were generated so that instances with more segments had a more 'rounded' structure. As mentioned previously, the

#	Var. Cost		CPU (s)	% Gap from UB		
Segs	Min	Max	Disagg	Agg	Disagg	
3	1	4	103	28.6	0.0	*
3	1	10	118	43.1	0.1	*
4	1	4	330	45.0	0.5	*
4	1	10	907	73.2	1.6	
5	1	4	286	65.6	0.3	
5	1	10	1064	112.4	1.4	

Table 4.8: Computational Experience on LTL-Like Cost Structures on Network 1

#	Var. Cost		CPU (s)	% Gap from UB		
Segs	Min	Max	Disagg	Agg	Disagg	
3	1	4	2	23.2	0.6	*
3	1	10	5	36.7	0.4	*
4	1	4	3	26.9	0.7	*
4	1	10	2	41.8	0.1	*
5	1	4	9	37.1	0.1	*
5	1	10	7	65.1	0.7	*

Table 4.9: Computational Experience on LTL-Like Cost Structures on Network 2

less linear the costs, the more improvement we gain from disaggregating the variables.

On Network 3, we also conducted experiments varying k, the number of destinations served by each origin, for the LTL case. The results were very similar to those shown in Figure 4-4.

The final cost function we consider is that of a truckload cost structure, like that of Figure 3-14. In these problems, the only factor in determining the cost function is the capacity and the fixed cost of the trucks. The capacity and the total flow on the network determines the number of segments that need to be defined on each arc. For each arc, the gaps are

#	Var. Cost		CPU (s)	% Gap from UB			
Segs	Min	Max	D/D	A/A	A/D	D/D	
3	1	4	<1	20.9	8.4	0.4	*
3	1	10	<1	31.1	12.5	0.6	*
4	1	4	1	27.8	8.9	0.3	*
4	1	10	1	45.0	13.8	0.8	*
5	1	4	3	37.4	9.6	0.1	*
5	1	10	6	63.2	15.4	0.8	*
8	1	4	6	46.6	13.9	0.7	*
8	1	10	16	102.8	24.7	2.9	*

Table 4.10: Computational Experience on LTL-Like Cost Structures on Network 3

independent of the magnitude of the fixed cost. For example, if we double the fixed cost, both the cost of the linear program and the cost of the integer program on this arc doubles and the gaps remains constant. Therefore, the magnitude of the relaxation gaps should not be effected by the magnitude of the fixed cost.

The linear programming relaxations of instances with these cost structures are harder to solve than those with other cost structures. We found that with these cost structures, the primal simplex method performed much better than the dual simplex method. For larger problems, the barrier method was also quite efficient, in some instances outperforming the primal simplex method.

Tables 4.11, 4.12 and 4.13 provide results on each of the three networks for these truckload costs. There are several noteworthy observations to be made. The gaps between the aggregated and disaggregated models are even larger than for the previous cost structures, although the difference decreases as the number of segments increases. In fact, for an instance of Network 2 with 15 segments, the gap between the aggregated and disaggregated formulations was only 20%. This result suggests that the aggregated approximation improves as we increase the number of segments. This result contrasts with the other cost structures for which the gap increased as the number of segments increased. This is the same phenomenon that we discussed in Section 3.8 regarding the network loading problem, and showed pictorially in Figure 3-7.

We also notice that the solutions to the disaggregated/aggregated model are the same as for the fully disaggregated model in the case of Networks 1 and 2. This is often the case for problems with a small number of segments. As the number of segments increases, however, we find instances where there is a small difference between these two formulations. As seen more clearly in the results for Network 3, the improvement from disaggregating the constraints grows as the number of segments increases. This result is to be expected, since the aggregated constraints are indeed aggregated over segments.

4.1.3 A Partial Disaggregation

Another option when disaggregating networks is to only partially disaggregate. For example, rather than defining a commodity for each origin node, we can group the nodes and define

#	CPU (s)		% Gap from UB			
Segs	D/A	Disagg.	Agg.	D/A	Disagg.	
2	607	1430	527.9	0.6	0.6	*
3	1018	2649	337.2	1.2	1.2	*
4	941	7868	251.6	4.3	4.3	
5	1624	7017	189.8	3.3	3.3	

Table 4.11: Computational Experience on TL-Like Cost Structures on Network 1

#	CPU (s)		% Gap from UB			
Segs	D/A	Disagg.	Agg.	D/A	Disagg.	
2	7	22	409.5	2.0	2.0	*
3	54	144	249.0	1.0	1.0	*
4	46	85	173.7	1.8	1.8	*
5	42	188	130.6	3.9	3.9	*
8	145	976	66.5	6.4	6.3	*

Table 4.12: Computational Experience on TL-Like Cost Structures on Network 2

#	CPU (s)	% Gap from UB				
Segs	D/D	A/A	A/D	D/A	D/D	
2	<1	85.2	85.2	0.0	0.0	*
3	<1	91.4	91.4	8.4	8.0	*
4	<1	78.9	78.9	9.9	5.1	*
5	1	64.2	63.6	7.6	3.5	*
8	7	54.4	48.6	18.5	9.0	*
10	10	36.1	33.9	12.7	5.6	*

Table 4.13: Computational Experience on TL-Like Cost Structures on Network 3

a commodity for each group of n nodes. This approach will provide a formulation that is tighter than the aggregated formulation, but will not be as large as the fully disaggregated one. Figure 4-6 graphs the solution for different levels of aggregation on the Network 1 which has 70 supply nodes and a single destination node. Although the results presented in this figure correspond to an LTL-like cost structure with four segments, the results were similar for other cost functions. The dotted line plots the solution time in CPU seconds. The solid line shows the percentage of the original gap between the solutions to the aggregated and the fully disaggregated formulations that is realized under each level of aggregation. A disaggregation level with $n = 70$ implies that we have defined a commodity for each group of 70 nodes, and this, therefore, corresponds to the aggregated model and so we have the full gap. If we let the level of disaggregation equal 10, for example, we define a commodity for each group of 10 nodes, and we see from the figure that we gain 40% of the total improvement we would gain if we disaggregated entirely. At a disaggregation level of 1, we define a commodity for each node, so we have the fully disaggregated model and therefore we have closed 100% of the gap. This figure shows that "you get what you pay for." One might hope that a small disaggregation (corresponding to a high value of n) would give large benefits and the solid line in Figure 4-6 would be concave. Unfortunately, this is not the case and we have to disaggregate extensively in order to see striking results. Nonetheless, if performance time is more important than finding the tightest possible lower bound, it might be appropriate to use a partial disaggregation that improves upon the aggregated formulation without exploding the solution time.

4.1.4 Summary of Computational Results on Standard Network Flow Problems

The computational experiments discussed to this point in this chapter suggest the following results.

- Disaggregation can significantly improve linear programming lower bounds for network flow problems with piecewise linear costs.
- On instances with concave costs for networks with a single commodity flowing from several origins to a single destination, the linear programming relaxation of the disaggregated

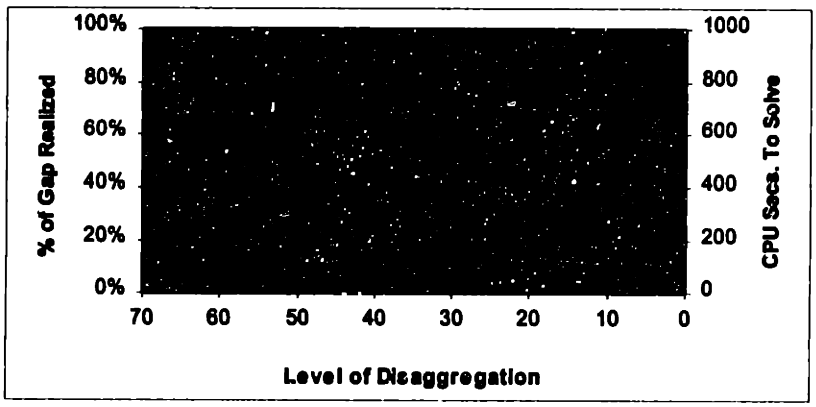

Figure 4-6: Level of Disaggregation vs. Solution Value and Time

formulation tends to solve with integral solutions.

- The aggregated formulation is particularly weak when the costs are either concave with large initial fixed costs, or have a truckload structure with a small number of segments.

- On multi-commodity networks with concave costs and each commodity flowing from a single origin to several destinations (or visa versa), a formulation with aggregated variables and disaggregated constraints (A/D) performs better than one with disaggregated variables and aggregated constraints (D/A) when there is a small (or no) initial fixed charge. As the fixed costs increase, however, the formulation with disaggregated variables and aggregated constraints performs better.

- As the magnitude of the fixed cost increases on a multi-commodity network with concave costs, the relaxation gaps for the two formulations with aggregated variables (A/A and A/D) increase, as does the relaxation gap for the fully disaggregated formulation (D/D). The relaxation gap for the formulation with disaggregated variables and aggregated constraints, however, generally increases.

- As the number of destinations/origin increases (while keeping the total demand per origin constant) on a multi-commodity network, the relaxation gap on formulations with aggregated variables (A/A and A/D) increases, while the gap on formulations with disaggregated variables (D/A and D/D) remains relatively constant.

#	Aggregated			D/A			Disaggregated		
Segs.	Consts.	Vars.	Bin.	Consts.	Vars.	Bin.	Consts.	Vars.	Bin.
3	1270	1170	585	21820	41535	585	41616	41535	585
4	1660	1560	780	22210	55380	780	63160	55380	780
5	2050	1950	975	22600	69225	975	70200	69225	975
8	3220	3120	1560	23770	110760	1560	119320	110760	1560

Table 4.14: Size of Formulations for Network 1

- The less linear the cost function, the more improvement can be gained from disaggregating the variables. The 'more concave' the cost functions are, the larger the gaps will be between the aggregated and disaggregated formulations.
- The fully disaggregated formulation is not as tight on instances with truckload style costs as it is for other cost structures.
- On instances with truckload costs, as the number of segments increases, the aggregated formulation provides a better approximation of the actual cost.
- Grouping commodities to achieve a partial disaggregation might be effective at limiting the size of a problem, but the rate at which the disaggregated model improves upon the aggregated model continues to increase as we further disaggregate.

We have seen that disaggregation can be a quite powerful technique for improving lower bounds on these types of problems. However, as the number of segments per arc grows, the problems quickly become so large that even solving the linear programming relaxation of the disaggregated formulation is quite time consuming, as seen by many of the solution times for the disaggregated linear programs, particularly on Network 1. Table 4.14 shows the relative size of these formulations on Network 1. So, although the disaggregated formulation appears to be tight, the problems can become prohibitively large. To solve large problems in practice, therefore, we need to develop more sophisticated approaches to solving the relaxations. The computational results in Chapter 5 illustrate this possibility.

4.2 The Facility Location Problem

In Section 3.3 we discussed the facility location problem. We showed how this problem can be transformed into a network flow problem with piecewise linear costs. Because this prob-

lem transformation results in a network with a single origin (see Figure 3-2), we can apply disaggregation to the formulation. In this case, we can define the flow to each customer as a commodity and then disaggregate the flow variables by commodity and add appropriate forcing constraints. Again, we can include forcing constraints of the form (3.28) to produce a disaggregated/aggregated formulation, or we can include constraints of the form (3.27), giving us a fully disaggregated formulation. We compare these two formulations against the aggregated one.

The data is generated with similar parameters used in Holmber and Ling [[19]], although we define the problems to be uncapacitated, i.e., each facility can be built large enough to satisfy all the demand. The demand per customer is random and uniformly distributed between 20 and 100. We define a constant $K = \frac{100n}{mq}$, where m is the number of possible facility locations, n is the number of customers, and q is the number of capacity options at each facility. We divide the total demand into q equal segments to define the segment bounds. This translates into a cost function with q equally sized segments on arcs from the dummy node to each facility. The incremental fixed cost is the same for each capacity level, and it is a random number between $100K$ and $200K$. We assume the production costs are zero, resulting in flat cost segments on arcs from the dummy node into the facilities. The transportation costs are linear with the euclidean distance between the facility and the customer, that is $c_{ij} = d_{ij}$.

We consider two physical networks. In the first, the number of possible facility locations is 10 and the number of customers is 20. In the second, the number of possible facility locations is 50 and the number of customers is 100.

Table 4.15 shows the relative gaps between the linear programming relaxation of each formulation and the optimal solution for four representative instances. We again find that the disaggregated formulation is quite tight and improves considerably upon the aggregated formulation. We also notice that the disaggregated/aggregated formulation also performs very well. Moreover, for the aggregated formulation, the smaller networks have smaller gaps and the problems with more facility sizes have smaller gaps. Note that the number of segments in the cost function is equal to the number of facility sizes. Therefore, this result is consistent with previous results for networks with these truckload style cost functions; that is, gaps are smaller when the cost functions have more segments.

Number of			LP CPU (s)		% Gap from Optimal		
Locs.	Custs.	Sizes	D/A	Disagg.	Agg.	D/A	Disagg.
10	20	2	<1	<1	12.9	0.0	0.0
10	20	6	<1	<1	3.1	1.0	0.8
50	100	2	2	7	75.7	0.0	0.0
50	100	6	3	13	30.4	0.0	0.0

Table 4.15: Computational Results for Facility Location Problem

In the case where the transportation costs are constant for each customer, we showed in Section 3.3 how we can reduce the problem to one with two nodes and m parallel arcs. In this case, disaggregation will not improve the formulation. The argument is equivalent to the one presented in Section 3.10, regarding disaggregation to commodities with unit demands. In this case, the disaggregation would define n commodities, but they would all have the same origin and destination. Therefore, it is possible to distribute the commodity flows such that we satisfy the forcing constraints without changing the cost of the solution.

In fact, when $CF = 0$, we can solve the linear programming relaxation of the aggregated formulation using a greedy algorithm. Since the linear programming relaxation estimates the actual costs with a convex function, the linear programming solution will start with the facility that has the least expensive initial segment cost, in the convex lower envelope, and saturate this segment. It will then look for the next cheapest possible segment - either continuing to add to this facility, or looking for another facility with a cheaper initial segment cost. We can continue in this manner until we have sent all the flow from the dummy node to the single destination node. The resulting solution will have at most one $y \notin \{0,1\}$. If one fractional variable exists, we can round it to one and have a feasible integer solution.

Because the majority of the arcs on these problems have linear costs, the relaxation gaps for the aggregated formulation vary substantially according to the relationship between the fixed costs at the facilities and the transportation costs. To examine problems with different relative transportation costs, we introduce a cost factor on the transportation costs. That is, $c_{ij} = CF * d_{ij}$. When CF is very large, the transportation costs dominate the problem; when $CF = 1$,we obtain the results in Table 4.15; when $CF = 0$, the problem has no transportation costs. Figure 4-7 graphs the gap for each formulation for varying cost factors on an instance with 50 locations, 100 customers, and 6 capacity options. For each instance, the dis-

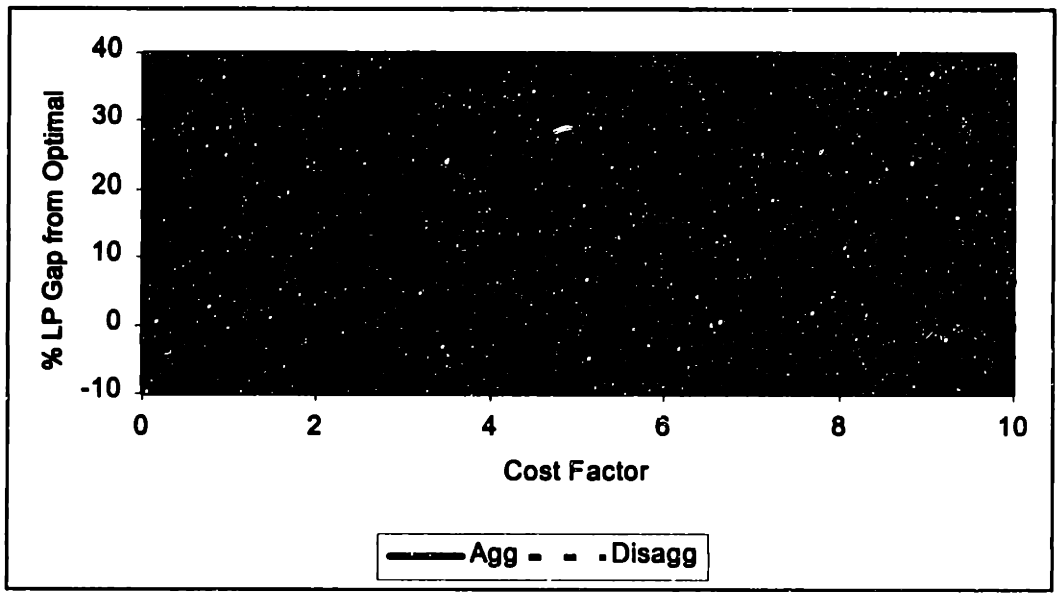

Figure 4-7: % Gap from Optimal as the Cost Factor is Varied

aggregated/aggregated formulation gave the same solution as the disaggregated formulation. As we see, the improvement gained by disaggregating is small when the transportation costs are either very small or very large, relative to the fixed costs at the facilities. As explained above, when the transportation costs are zero, we have no improvement at all. When the transportations costs are very large, the improvement from disaggregating on the arcs into the facilities is overshadowed by the linear transportation costs, resulting in only a small percentage improvement. As we see in this graph, however, there is a range for the cost factor in which we can still gain substantially by disaggregating.

We also should not underplay the impact of disaggregating, however, even when it might only improve 10% upon the aggregated formulation. The solution to the linear relaxation of the disaggregated formulation is very tight (within 1% for all instances tested), and therefore, the subsequent branch and bound can solve for the optimal solution quite efficiently. Starting with a linear relaxation that is, for example, 10% from optimal might lead to a much longer branch and bound process.

We learn from this computational analysis that modeling this facility location problem with a disaggregated network flow formulation provides tight linear programming relaxations.

However, the disaggregated/aggregated formulation is often just as tight, and the aggregated formulation also performs quite well for most cost parameters.

4.3 The Network Loading Problem

4.3.1 Undirected Case

In Section 3.8 we discussed three formulations for the network loading problem. We use computational results to compare the standard and disaggregated formulations for the undirected case, as seen, for instance, in telecommunications applications. For the directed case, the results for truckload functions shown previously are indicative. We label the standard network loading formulation, given by (3.29), as SNLF and the disaggregated network flow formulation, given by (3.31), as DNFF. As seen in the results for the network flow problem with truckload costs, the objective values of the solutions to the relaxations of the disaggregated/aggregated formulation and the disaggregated formulation are very close for these staircase cost functions. In fact, for most problems they yield the same objective value. The same was true when we tested the disaggregated/aggregated network loading formulation. We, therefore, do not report on these results. We solved the linear relaxations of the two network flow formulations with CPLEX's barrier algorithm. We found that this algorithm performed much better than either the primal or dual simplex method.

These problems are quite large in practice and both the disaggregated/aggregated and the disaggregated formulations can quickly become unwieldy. Although they result in better lower bounds than the standard formulation, they are often too large to solve to optimality with branch and bound. We, therefore, focus attention on the performance of rounding heuristics for finding good integer feasible upper bounds. In particular, we compare the performance of the rounding heuristic on the disaggregated formulation (DNFF-RH), with the performance of an edge rounding heuristic on the standard formulation (SNLF-RH). As described earlier, DNFF-RH first solves the linear relaxation of the disaggregated formulation. It then rounds to one any y variable whose value in the optimal linear programming solution exceeds a user-supplied threshold. It also rounds to zero any y variable with an optimal linear programming value of zero. With these variables fixed, the heuristic then performs branch and bound on the

remaining problem. If the resulting problem has a feasible and integral solution, it is feasible for the original problem and provides an upper bound on the optimal solution to the original problem. Alternatively, one can use this same procedure on the disaggregated/aggregated formulation. We found, however, that the network flow rounding heuristic performed slightly better on the solution to the disaggregated formulation then on the solution to the disaggregated/aggregated formulation. In many cases, the heuristic rounded more variables for the disaggregated solution, and therefore the branch and bound tree was smaller. For large problems, however, it might be advantageous to use the disaggregated/aggregated formulation since we can solve the initial linear relaxation more quickly.

Recall that in SNLF, the y variables are integer, not binary. SNLF-RH solves the linear relaxation of the standard formulation and then rounds each y variable up to the nearest integer. For example, if the linear programming solution installs 3.2 units on an edge, the heuristic solution will install 4 units. This solution is feasible and provides an upper bound on the optimal solution. Epstein [13] reports further results on this and other heuristics.

The instances reported here are generated using the network generator used by Epstein. He provides details on the nature of the network and demand structures generated and the methods used for generating them. K and S refer to the number of commodities (defined as an origin/destination pair with a nonzero demand) and segments, respectively. Tables 4.16 and 4.17 report results on a network with ten nodes and a uniform demand pattern. Each node of the networks in Table 4.16 has an edge degree between three and five. The networks in Table 4.17 are complete.

As expected, we see that the disaggregated model can provide much tighter bounds than the standard formulation. We also notice in these results that the rounding heuristic on the disaggregated formulation (DNFF-RH) performs very well. In over half of the instances, it solves with the optimal solution. In the worst case, it finds a solution within 2.4% of optimal. Recall, however, that these networks had only ten nodes. When problems are large, solving DNFF is very time consuming. In addition, solving the mixed integer portion of the rounding heuristic, DNFF-RH, is also too computationally difficult. In many cases, rounding the variables reduced the problem by at least 75%. Unfortunately, when the original problem is very large, even a 75% reduction in problem size is not sufficient to place it within the scope

		LP % from Opt.		Heuristic % from Opt.	
K	S	SNLF	DNFF	SNLF-RH	DNFF-RH
15	3	138.6	34.1	90.1	0.0
15	4	76.9	19.0	70.9	0.0
15	8	39.1	20.0	39.3	1.5
20	5	34.1	16.9	67.0	0.0
20	7	18.6	10.4	49.2	0.0
20	14	6.1	4.3	27.2	1.6
28	6	33.4	9.5	73.2	0.0
28	9	20.8	8.3	53.0	0.0
28	17	9.7	7.5	22.2	0.8
39	8	27.5	8.3	91.0	0.0
39	12	16.6	6.5	74.4	2.4

Table 4.16: Computational Results for a Network with Degree from 3 to 5

of branch and bound.

In both tables we notice that the relaxation gaps for the standard formulation decrease as the number of segments increase. This result was suggested in Section 3.8. In addition, the relaxation gaps for the disaggregated formulation tend to increase as the number of segments increase. Another way of viewing this is to plot the difference between the objective values of the two linear programming solutions versus the number of segments. As seen in Figure 4-8, problems with many segments tend to exhibit less improvement from disaggregation than those with fewer segments. The disaggregated formulation is more effective when the number of segments is small.

Another apparent observation is that although the disaggregated model produces better bounds than the standard formulation, we do not see the improvements that we saw for the standard network flow problem with truckload cost functions. Much of this can be explained by Propositions 8 and 9, as we will discuss below.

In conclusion, we find that although the disaggregated model improves upon the standard formulation, it still produces rather large relaxation gaps. Moreover, although the rounding heuristic works quite well on small problems, it is not efficient on large problems. For these reasons, we cannot solve the problem directly and there is still a need for more efficient methods for solving the network loading problem. Nonetheless, the improved formulation is still valuable and with further study, might be an important component of developing an approach for solving this problem.

		LP % from Opt.		Heuristic % from Opt.	
K	S	SNLF	DNFF	SNLF-RH	DNFF-RH
14	4	130.9	30.8	91.5	0.0
14	5	91.4	27.2	54.0	0.0
14	10	38.1	16.1	27.6	0.1
18	5	106.4	27.5	113.2	0.0
18	7	63.9	14.4	79.0	1.3
18	13	31.7	13.3	35.7	0.8
34	7	56.4	10.8	215.4	0.0
34	22	15.0	3.4	51.0	0.3

Table 4.17: Computational Results for a Complete Network

4.3.2 Comparison to the Directed Case

The directed case of the network loading problem is equivalent to the network flow problem with truckload style costs, so the results previously presented for this case are indicative of the results we would expect for the directed network loading problem. From Propositions 8 and 9, however, we expect the magnitude of the gaps to be different than in the undirected case. To illustrate this, Table 4.18 presents results for the directed case on some of the same instances reported in Table 4.16. Comparing these results provides a sense for the difference between the directed and undirected instances of the network loading problem. Indeed, the relaxation gaps for the standard formulation are larger for the directed instances, and the gain from disaggregating is also larger in the directed case. In addition, the relaxation gaps for the disaggregated formulation seem to be smaller for directed problems, although this difference is more prominent on smaller instances.

These examples provide evidence that using a disaggregated network flow formulation for the network loading problem is more effective when the problem we are solving does not allow capacity to be shared by flow traveling in each direction on an arc. Such applications arise in transportation when freight traveling from node i to node j cannot be moved on the same carrier as freight traveling from node j to node i, at least not simultaneously. In most telecommunications applications, we are installing capacity in the form of data transmission lines that can carry data being transmitted from node i to node j and from node j to node i simultaneously. In these latter problems, the disaggregated network flow formulation can still improve significantly upon the standard formulation, but just not to the same extent as

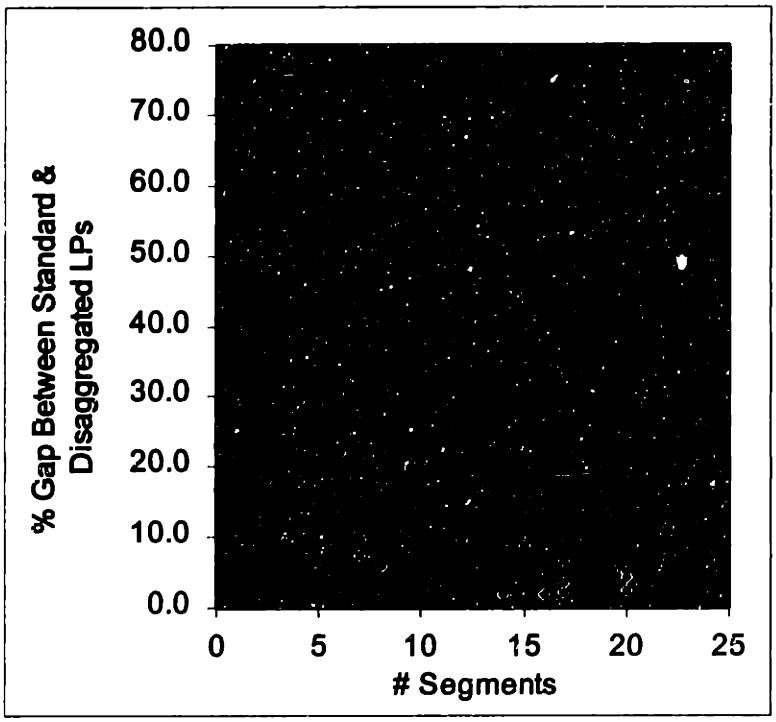

Figure 4-8: Difference Between Standard and Disaggregated Formulation vs Number of Segments

it does for directed problems.

4.4 Summary

In the previous chapter we examined the analytical and geometrical implications of disaggregation on a single arc. The computational results we reported in this chapter allow us to compare the solutions of the aggregated and disaggregated formulation, as well as to a known upper

		Undirected LP % from Opt.		Directed LP % from Opt.	
K	S	SNLF	DNFF	SNLF	DNFF
15	3	138.6	34.1	174.3	6.4
15	8	39.1	20.0	48.9	16.4
20	5	34.1	16.9	60.1	16.5
20	7	18.6	10.4	40.3	16.8

Table 4.18: Comparing the Undirected and Directed Results

bound. We saw that the aggregated formulation was comparatively weak, with typical gaps between the aggregated and disaggregated formulations ranging from 20% to 200%. However, the disaggregated formulation was frequently within 2% of the optimal solution, and in the case of concave costs, it was always integral, and therefore optimal, for the single-destination problems. The multi-commodity problem had small gaps – usually within 1%. These results provide computational evidence that disaggregation can be quite powerful when applied to network flow problems with piecewise linear costs.

The computational experiments also showed that disaggregation can be an effective tool for modeling the multiple-capacity facility location problem and the network loading problem. The effectiveness, however, is dependent on the size and cost structure of the problem. For the step functions inherent in these problems, the linear relaxations are computationally difficult to solve, so the trade-off between solution time and improved tightness of the formulation requires close examination.

Chapter 5

An Application of Non-Concave Costs: The Merge-in-Transit Problem

We now examine an application of the results in the previous chapters to a current real problem in supply chain management. In this chapter we describe the problem, present a model and a solution algorithm, and discuss computational results. This work has been conducted jointly with Professor Bernard Gendron of the University of Montreal.

5.1 Background

The 90's has been a decade that has revolutionized the way manufacturing firms think about customer service. It is not enough to produce a quality product - now it must be a quality product delivered to the right place at the right time. A significant part of accomplishing this is through good management of the distribution process. Firms that want to compete in today's market must find cost effective ways to meet their customers' needs.

In the computer industry, among others, this process can be complicated by products that are composed of several components, produced at geographically dispersed locations. A customer might place an order for computer X, which translates into an order for a monitor, a CPU unit, a printer and a keyboard - all of which might be produced at different sources.

The customer does not want to receive four shipments. It wants one shipment to arrive on its dock on the requested day.

The manufacturer can meet this need in several ways. It can maintain large warehouses with inventories of all the components. When a customer places an order, the manufacturer takes these components off the shelf and repackages them into a single shipment. As an alternative, it can use merge centers that act as in-transit consolidation points. It can ship components from plants to a merge center, where they are consolidated into the final product requested by the customer. These merge-in-transit centers are not intended to be inventory centers. Therefore, the firm needs to coordinate the component shipments so that they arrive simultaneously (or nearly so) and can be bundled and shipped immediately to the final customer for arrival on the due date.

Several companies are adapting this merge-in-transit concept and have found it to be an appropriate way to meet customers' needs. Compaq, for one, has implemented a merge-in-transit system as part of a complete, and very successful, overhaul of their logistics process. In addition to the computer industry, merge-in-transit might be appropriate for consumer goods companies like 3M. As with computers, a typical 3M customer order is composed of several components produced around the country. Lucent Technology has also considered merge-in-transit. Their business is primarily build-to-order and their components are expensive. These attributes make them a good candidate for a merge system.

Although merge-in-transit can be an efficient and effective system, it is also a complex one to design and manage. A firm first has to decide if this is an appropriate distribution strategy for their business. If the answer to that question is in the affirmative, then there are a host of tactical questions regarding the number and location of merge centers. Even with these issues settled, the daily management of the system offers substantial challenges.

In the following sections, we describe a model that addresses operational issues of a merge-in-transit system. The model was developed to address issues faced by Caliber Logistics. Caliber, based in Cleveland, Ohio, is a third party logistics firm that provides transportation services to manufacturing companies. Many of its clients are asking Caliber to more actively manage their supply chains. Several of these companies, particularly in the high tech industries, would like to implement a merge-in-transit system for their distribution process and are

asking Caliber to help with the strategic and operational planning.

5.2 Problem Description

A merge-in-transit program uses a two echelon distribution system. Figure 5-1 depicts the underlying network. Components move from source points to merge centers, where they are consolidated into products and shipped to customers. In general, a firm might source multiple components from one plant, or source one component from multiple plants. In the computer industry, manufacturers usually produce a component at a single plant. For example, Source 1 might produce Monitor 1 and Monitor 2. Source 2 might produce CPU1; Source 3 might produce CPU2, and Source 4 could produce Printer 1.

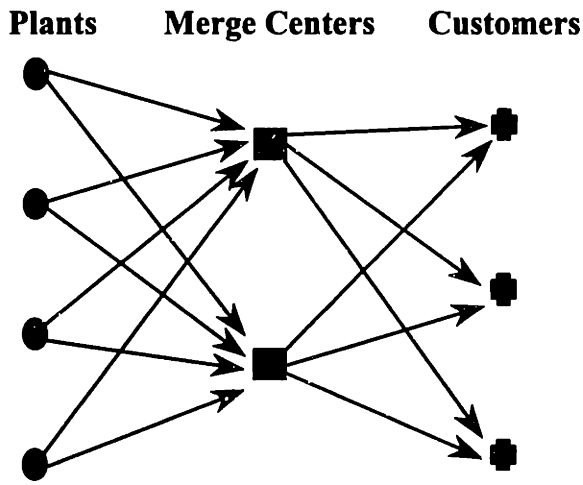

Figure 5-1: A Merge-in-Transit Network

Each product is composed of several components as described by a "recipe" of its components. For example, Product A might require Monitor 1, CPU1, and Printer 1. Product B might require Monitor 2, CPU2, and Printer 1. Each time a customer orders a product, all of its components must arrive at one of the merge centers before the entire product can be shipped to the customer. The merge centers are not intended to be inventory centers; they should hold inventory only when it is cost effective to do so (we describe such circumstances later). Consequently, we must coordinate the shipments so that they arrive nearly simultaneously so that we can ship them on to the customer with minimal delay.

In addition to merging components into products, the system needs to merge products into orders. To fulfill a customer order of 10 units of Product A and 20 units of Product B, the system must merge both products at the same merge center and ship them via the same mode. This assures that the customer receives its entire order in a single shipment.

Not only do we need to be concerned about the timing (when to ship) and routing (which merge-center to use) of each shipment, we must also select a mode of transportation. In general, when choosing a transportation mode, we face a trade-off between the shipment cost and transit time. We consider four modes: Air, Truckload, Small Package, and Less-Than-Truckload (LTL). Air is the quickest, but most costly. It might be the required mode for high value or fragile items. Truckload shipments are cost effective for large shipments. Small Package and LTL shipments are the most cost effective for small shipments, but are the slowest modes. In the following section, we will further discuss the cost structures for each mode.

We can state the merge-in-transit problem we consider as follows: **Given a set of orders (product type, customer location, and delivery date) over a short term planning horizon, find the optimal routing, mode, and timing for sending components from plants to merge centers, and products from merge centers to customers in order to meet demand.**

Several key features complicate this problem:

- choosing among the transportation modes to assure that products arrive "simultaneously" while minimizing costs,
- because of economies of scale, it might be cost effective to send components earlier than they are needed (resulting in short term inventory),
- the selection of a merge center might require a trade-off between the best choices for different components, and
- due to economies of scale, it might be cost effective to consolidate freight and send items to a merge center that in isolation might not be optimal.

The second point addresses the issue of inventory. Although the merge centers are not intended to be warehouses for inventory, they are usually designed to hold limited levels of

stock. Two situations might drive a company to hold inventory at a merge center. The issue raised by the second bullet leads to short-term inventory. If Merge Center 1 needs 10 monitors on day 4 and 1 monitor on day 5, it might be cost effective to ship all 11 monitors to arrive on day 4. The trade-off cost is the one day inventory of one monitor. If the economies of scale in the shipping rates justify this inventory cost, the optimal solution will ship all 11 units at once.

Another situation might create longer term inventory. If the product recipe includes a small generic item, we might stock rather sizable quantities of it rather than shipping it as customers place each order. An example might be a printer cable or user manual, items that will probably be bundled with each computer. Rather than attempting to coordinate the arrival of such items, it is probably beneficial to stock them at each merge center. Our model does not consider this type of long term inventory. If this inventory is desired, we would determine its optimal replenishment frequency and size independently using traditional inventory analysis like the Economic Order Quantity (EOQ) model.

The third bullet addresses an issue of compromising between components. While one merge center might be "ideal" for one component, for example, it is in a direct line between the plant and the customer, it might be less costly to merge the order at a less ideal merge center because it is, for example, closer to the source for two of the other components. The model needs to consider these trade-offs.

The fourth point is very important. Due to economies-of-scale, we need to consider all orders collectively. In general, the cost of transportation favors large shipments, since the per unit transportation cost decreases as the size of the shipment increases. Therefore, from the perspective of each order in isolation it might appear optimal to send a component for two orders to two different merge centers, but it might actually be more cost effective to consolidate the orders and send the components to the same merge center.

The problem description makes the following assumptions:

- Merge center locations are given and fixed.
- Demand is known for the time horizon being considered.
- Orders can be routed through any merge center, i.e , customers need not be sourced from

a single merge center.

- The merge centers are capacitated.
- Orders must arrive at the customer ON the delivery day – not early and not late.
- Orders must arrive in a single shipment, i.e., on the same mode via the same merge center.
- We consider components as arriving to a merge center "simultaneously" if they arrive on the same day.
- Product is available to leave the merge center the same day that all necessary components are available at that merge center.
- All locations are domestic (so we need not worry about issues of duty or local content).
- The sources have unlimited supply.

We expect that a firm would run this model on a daily basis using the most recent demand data. Although an appropriate time horizon is dependent on the business and the nature of the demand patterns, we consider five to seven days as a typical time horizon. On a given day, the firm would exercise the model and then plan on executing the schedule for the next day as the model solution suggests. The following day it would update the demand data and re-run the model. This execution with a rolling calendar allows the model to make use of demand data as it enters the system, but the final schedule is determined on an as-needed basis.

To solve this problem, we developed a mixed integer linear program. Before addressing the specifics of the model, however, we will review the relevant literature and examine the cost functions for the four modes we consider.

5.3 Literature Review

To the best of our knowledge, there is very little existing literature on the merge-in-transit problem. Hastings [17] and Dawe [12] describe the business issues underlying merge-in-transit. Cole [9] develops a basic optimization model and GIS-based support system that addresses design issues, including the number and location of merge centers.

Although merge-in-transit literature is limited, several related problems are well studied, including applications in transportation logistics and assembly systems. The body of literature on transportation logistics is large and broad in scope. Bramel and Simchi-Levi [6] and Crainic and Laporte [11] provide thorough reviews of much of the research in this area and discuss key strategic, tactical, and operational issues.

The merge-in-transit problem is a multi-commodity integer generalized network problem. As such, it is closely related to the applications in assembly systems. For a review of the literature in this area, see the survey by Shapiro [28]. Although some recent books like Ahuja, Magnanti, and Orlin [1] discuss multi-commodity networks and single-commodity generalized networks, to our knowledge, there are very few references on multi-commodity generalized networks.

5.4 The Costs for Each Mode

In the transportation industry, a "lane" is defined by an origin, destination, and mode. Using this terminology, we will describe the merge-in-transit problem as defining and loading the lanes in the distribution system. The cost of a shipment will depend on its size and the lane on which it travels. Our model includes four modes of transportation: Air, Small Package, Less-than-truckload (LTL) and Truckload. Each mode has its own cost structure, but each is piecewise linear. In some cases, good approximations will help simplify modeling. The following discussion of cost structures is based on our conversations with Caliber Logistics.

5.4.1 Truckload

When sending a shipment via truckload, the shipper is paying for a dedicated truck between a single origin and a single destination. The cost structure includes a fixed charge and a per mile charge, which might vary from lane to lane. The shipment charge is independent of weight and therefore the cost per truck is constant for each lane. There are, however, weight and volume capacities. Therefore, we will require additional trucks for shipments whose weight or volume exceeds these capacities. If we want to consider the option of multiple trucks, we will have to choose to consider either the weight or the volume capacity. This is usually not a restriction in practice because in many industries, trucks either cube-out or weigh-out and

we need to consider only one of the capacities. Because the fixed charge is large, shipping by truckload is not economical for small shipments. Figure 5-2 shows a typical cost curve for a truckload shipment. In this figure, c denotes the weight capacity of a truck.

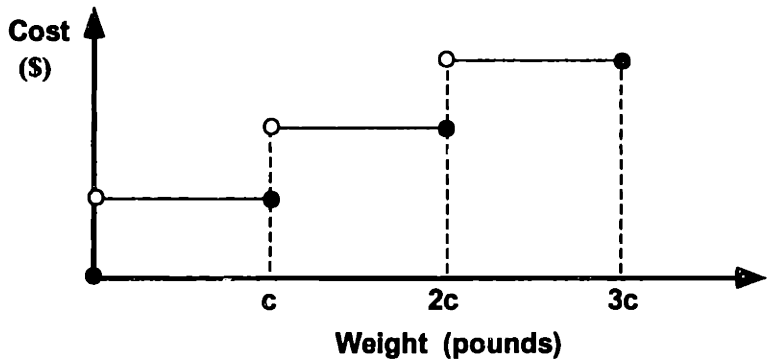

Figure 5-2: A Typical Truckload Cost Function

Often several truck sizes are available; for simplicity, we assume a single truck size in our model. To consider multiple truck sizes, we could define the cost function with an appropriately sized segment for each combination of the trucks. We calculate travel times using a rule of thumb that a truck can cover 600 miles per day. A simple calculation then gives us a transit time for each truckload lane.

5.4.2 Air

For our purposes, we assume that every point-to-point demand can travel via air in one day. Prices for air transportation are generally quoted with a minimum shipment cost and then a per pound cost with price breaks for increasing shipment size. The cost is usually independent of distance. Figure 5-3 shows a typical cost curve for air transportation.

Because air travel is the most expensive form of transportation, a shipper will only use it when time constraints permit nothing else, or when the component is fragile or valuable and it is the required mode. Therefore, the choice of air transportation is governed by issues other than cost. This property provides us more flexibility in approximating this cost function. For the air mode, it is unlikely that an optimal solution will change under a reasonable approximation.

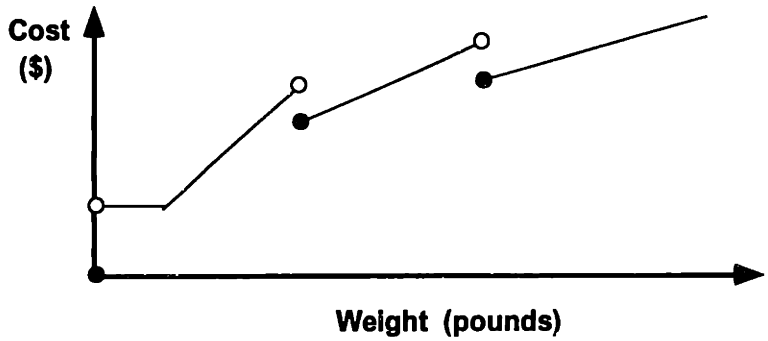

Figure 5-3: A Typical Air Cost Function

5.4.3 LTL and Small Package

Services like UPS and RPS provide Small Package delivery. The cost is a function of weight and distance. The distance of a shipment determines a "shipment zone," with a higher zone corresponding to longer distance. The cost structure for each zone includes price breaks for increasing weight. The cost per weight-class increases as the zone increases. Although the cost varies by distance in this manner, for each lane the distance is fixed and it is therefore only a function of weight.

A firm wishing to ship goods via LTL (less-than-truckload) contacts a carrier with the shipment size, origin and destination. The LTL carrier then collects orders from multiple clients and determines how to consolidate and route all the shipments. It will generally use its own consolidation points so that it can load several orders onto one truck. In terms of cost structure, LTL transportation is analogous to small package transportation, in that the shipper pays a per pound charge with price breaks for increased weight. The difference, however, is that LTL pricing does not use the concept of zones. Instead, the shipper negotiates the cost per weight-class for each lane. From a modeling standpoint, however, the two are analogous.

A minimum charge is imposed on LTL shipments that discourages extremely small shipments. The magnitude of this minimum charge implies that shipments under a specified weight should be sent by the small package mode and those over this weight should be sent by LTL. Therefore, the cost function for LTL can be viewed as the natural extension of the small package cost function. This observation allows us to model small package and LTL transportation as a single mode with a single cost function for each lane. Figure 5-4 shows a typical Small

Package/LTL cost function.

Because LTL carriers are consolidating shipments, LTL shipments have a longer transit time than Truckload shipments. In our model, we assume that an LTL load covers 450 miles/day.

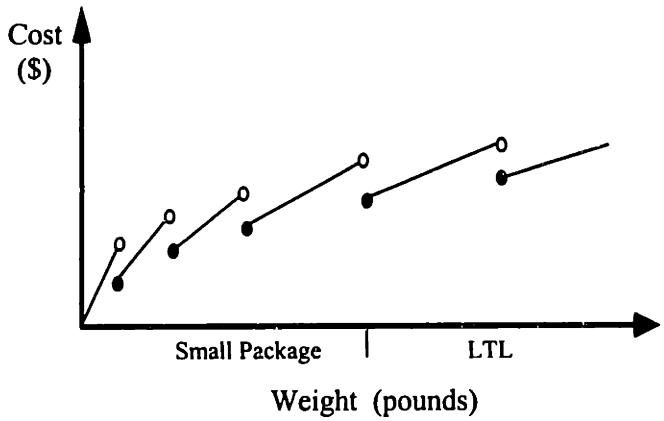

Figure 5-4: A Typical LTL/Small Package Cost Function

5.4.4 Approximations to the Cost Functions

There are several ways to represent these complex cost functions. Generally, we must make a trade-off between modeling and algorithmic simplicity and accuracy. For example, a linear approximation is the easiest to handle both in modeling and in solution development. It misses, however, the advantages of consolidation and, therefore, will not yield quality solutions. In fact, a completely linear approximation will just send everything on the least expensive per-unit path time will allow and will miss all opportunities for consolidation. For the Air and Small Package/LTL functions, we consider three alternative cost functions. Figure 5-5 illustrates each one. The first representation, 5-5a, models the cost with a piecewise linear, discontinuous function. For some applications, this representation is the most accurate, but some solution techniques require continuous cost functions.

Notice that an intelligent shipper would never send a shipment at a weight just to the left of a breakpoint in Figure 5-5a. If this is the weight of a shipment, it is less costly to declare the shipment to be at the next price break. For example, using the function of Figure 5-5a, sending a shipment of 9 pounds costs $90. However, declaring this shipment to be 10

pounds (or filling it with sand so that it is indeed 10 pounds), reduces the cost to $80. This observation indicates that it might be appropriate to alter the cost function to one like that of Figure 5-5b which shows the effective function of Figure 5-5a. Indeed many modelers use this representation because they feel it more accurately reflects the actual costs charged by the carriers. The other advantage to this cost function is that it is continuous.

In a third representation, 5-5c, we estimate the cost function with a piecewise linear and concave function. Appropriate algorithms can then exploit the resulting concavity. Note that depending on the application, a concave function might be an exact representation. If a shipper uses an incremental discount rather than an all-unit discount (e.g., the charge is $10 for the first 10 pounds, $9 for the next 10 pounds, etc.), then the actual cost function will be piecewise linear and concave. In this transportation application, however, the price break applies to the entire shipment so a concave function will only provide an approximation.

The most appropriate representation to use will be dictated by the importance of solution accuracy and the assumptions of the solution technique. For example, if we have a very efficient algorithm for piecewise linear concave functions, then approximating the cost function with such a function might be wise. Alternatively, if a solution technique does not require concavity, but relies on a continuous function, then the representation of Figure 5-5b might be appropriate.

Our initial model of the merge-in-transit problem uses the continuous non-concave functions as shown in Figure 5-5b for the air and small package/LTL modes. We believe that because of the operational nature of this model, it is important to use an accurate representation. Although there is some debate, most practitioners seem to feel that this cost model is the most accurate. One issue with this representation, however, is that the resulting cost functions have nearly twice as many segments as the discontinuous representation. We address this concern by implementing an approximation technique that attempts to merge segments without losing significant accuracy. We examine sequential segments and merge them into a single segment if the resulting segment provides a lower approximation and the gap does not exceed a pre-specified value. We have found through computational experience that we can often halve the number of segments while maintaining sufficient accuracy.

We saw in Chapter 4 that disaggregation was particularly effective on problems with con-

cave costs, suggesting that we should consider implementing the model with a concave approximation. However, for the cost data provided to us, the small package/LTL costs had the basic form shown, but there were occasions where the cost increased at a breakpoint. As a result, we were unable to define a quality concave approximation.

Also, although we use a continuous representation, it is important to note that the model and algorithm we develop is general, and so we are able to solve problems with any form of piecewise linear costs.

5.5 The Basic Model

In general form, our model states the following:

Minimize	Transportation + Inventory cost
Subject to	Flow components and products through merge centers so as to satisfy merge constraints,
	Meet demand with a single shipment,
	Don't exceed merge center capacities.

We think of the model working as a "pull system." The flow of products from merge centers to customers must meet customer demand. The solution must maintain flow balance at each merge center, requiring appropriate flow of components from the sources to the merge centers. Because we assume unlimited supply at each source, the model has no stated flow balance constraints at the sources.

Due to the nature of the system, we need to assure that model handles four peculiarities:

- Components flow in and products flow out, and every component in a given product must be present before that product can be shipped.
- We must choose an appropriate mode and account for its cost and transit time structure.
- Flow is not instantaneous – we need to account for the time delay between components being shipped from the sources and arriving at the merge centers, and products being shipped from merge centers and arriving at customers.
- The merge centers might hold short-term inventory.

To address these issues, we introduce a time and mode expanded network. As shown in Figure 5-6, the expanded network contains three sets of nodes and three sets of arcs. The node sets include the set P, with a node for each plant and day, and the set M, with a node for each merge center and day. In addition, the network includes the set D, with a node for each demand point, defined by the customer location and the day of delivery. Each demand point has associated with it an order composed of a set of products required at that node. We can use the product recipes to explode the product demands into component demands. Therefore, with demand node j we associate d_j^k, the total demand for component k in that order.

We define the arcs on the networks by a physical link (source and destination), a mode, and a time (for our purposes, a day). A shipment on arc e then is known to be moving from a particular source to a particular destination, on a particular mode and day. We can then define all the appropriate movements, accounting for travel time, as well as the inventory holdings at merge centers.

For each plant node we include three arcs, one for each mode, going to each of the physical merge centers with the appropriate time delay representing the transit time for that mode. On each of these arcs we define three variables: x_e^s, the total weight of the shipment if it lies in segment s of arc e; u_e^k, the unit flow of component k on arc e; and y_e^s, a binary variable indicating if the flow weight on arc e lies in segment s. The cost of flow on these arcs is piecewise linear and defined by the notation in Figure 3-5.

The second set of arcs, representing inventory holdings, includes an arc for each merge center at time t, to the same physical merge center at time $t+1$. The variable v_e^k defined on these arcs represent the unit inventory of component k at the merge center and day appropriate to arc e. The cost of the "flow" on these arcs, h_e^k, which varies by component, equals the daily cost of holding component k at this merge center.

The arc set also includes arcs from merge centers to customers. For each demand point the network contains three arcs, again one for each mode, from each physical merge center. The head of each arc is a demand node and its tail is a node representing a merge center at an appropriate time, allowing for transit time, prior to the delivery date of the order. There are several ways to handle the merge and single shipment constraints. We could, for example, explicitly write constraints using component flow variables on all the arcs. Another possibility

is to define *components* flowing from suppliers to merge centers and *products* from merge centers to customers. The flow balance constraints then need to account for the recipes that turn components into products. Instead of either of these two methods, we have chosen to define a single binary variable, w_e, on each arc from a merge center to a customer that indicates whether we ship demand for that customer and day on that arc. This approach is valid since products must arrive in a single shipment and cannot be early or late. By restricting only one of these binary variables to be one for each demand point, we are enforcing both the merge constraints and the single shipment constraints. The cost on these arcs is a constant, C_e, which is the cost of sending all of the demand associated with demand node j, $\sum_k d_j^k$, on this arc.

In summary, we use the following notation:

SETS:

P	nodes in the expanded network representing plants
M	nodes in the expanded network representing merge centers
D	nodes in the expanded network representing demand points
A_P	arcs in the expanded network from plants to merge centers
A_I	arcs in the expanded network representing inventory holdings
A_D	arcs in the expanded network from merge centers to demand points
H_i	arcs arriving at node i
T_i	arcs leaving node i

DATA:

h_e^k	daily cost of holding component k at the merge center associated with arc e
p^k	unit weight of component k
l^k	unit volume of component k
q_i	volumetric capacity at merge center i
d_j^k	units of component k required to satisfy demand node j
C_e	total cost of shipment which satisfies demand associated with arc $e \in A_C$

VARIABLES:

x_e^s total weight of shipment on arc $e \in A_P$ if it lies in segment s

v_e^k unit inventory of component k at a merge center on arc $e \in A_I$

u_e^k unit flow of component k on arc $e \in A_P$

y_e^s binary variable indicating if flow on arc $e \in A_P$ lies in segment s

w_e binary variable indicating if the demand associated with arc e is met with a shipment on arc $e \in A_C$

With our expanded network and this notation, we can formulate the merge-in-transit problem as:

$$\text{Minimize} \quad \sum_{e \in A_P, s} (f_e^s y_e^s + c_e^s x_e^s) \quad + \sum_{e \in A_I, k} h_e^k v_e^k + \sum_{e \in A_D} C_e w_e \tag{5.1}$$

$$\text{subject to:} \quad \sum_{e \in (H_i \cap A_P)} u_e^k + \sum_{e \in (H_i \cap A_I)} v_e^k = \sum_{e \in T_i} d_j^k w_e + v_e^k \quad \forall \ k, \ i \in M \tag{5.2}$$

$$\sum_k \left(\sum_{e \in (H_i \cap A_P)} l^k u_e^k + \sum_{e \in (H_i \cap A_I)} v_e^k \right) \leq q_i \quad \forall \ i \in M \tag{5.3}$$

$$\sum_k p^k u_e^k = \sum_s x_e^s \quad \forall \ e \in A_P \tag{5.4}$$

$$\sum_{e \in H_i} w_e = 1 \quad \forall \ i \in D \tag{5.5}$$

$$\sum_s y_e^s \leq 1 \quad \forall \ e \in A_P \tag{5.6}$$

$$b_e^{s-1} y_e^s \leq x_e^s \leq b_e^s y_e^s \quad \forall \ e \in A_P \tag{5.7}$$

$$x_e^s \geq 0, \ v_e^k, \ u_e^k \in Z^+, \ w_e, \ y_e^s \in \{0,1\}. \tag{5.8}$$

The objective function has three terms: the fixed and variable charge for all flows on arcs from suppliers to merge centers, the inventory cost of items held in inventory at the merge centers, and the total charge of shipments from merge centers to customers. Constraints (5.2) act as the flow balance constraints at the merge centers. They state that the flow for each component into a merge center on a particular day plus the available inventory must equal the flow out of the merge center plus the inventory held on the next day. Notice that the flow out of the merge center is expressed by the single shipment variables, assuring that the appropriate components are merged into each order. Constraint (5.3) enforces the capacity of each merge center. Constraint (5.4) expresses the relationship between the u variables, representing the

unit flow of each component, and the x variables, representing the total weight of the flow on each segment. Constraint (5.5) assures that we meet each demand with a single shipment. Finally, the inequalities (5.6) and (5.7) are the now-familiar constraints on the selection of a segment.

We can tighten this formulation by adding forcing constraints:

$$u_e^k \leq M_e^k \sum_s y_e^s \quad \forall \ k, e \in A_P \tag{5.9}$$

Clearly, the smaller we can make M_e^k, the tighter the formulation. We will let D^{t+} be the set of demand points with a delivery date later than day t. Valid bounds on u_e^k, the unit flow of a component, include $\sum_{j \in D^{t+}} d_j^k$, the total future demand for the component, and $\lfloor \frac{q_i}{l^k} \rfloor$, the capacity of the merge center in units of component k. We will set M_e^k to the smaller of these two numbers, i.e., $M_e^k = min(\sum_{j \in D,\, t+} d_j^k, \lfloor \frac{q_i}{l^k} \rfloor)$

5.6 Approaches to Disaggregating the Model

Although the formulation given by (5.1)-(5.9) is valid, we can apply the disaggregation technique described in Chapter 3 to develop a formulation with a tighter linear programming relaxation. Disaggregating is possible at several levels. We can consider disaggregations of both the u variables and the x variables. We will first consider disaggregating the flow variables by defining u_e^{sk}, the unit flow of component k on segment s of arc e. We then write

the formulation as:

$$\text{Minimize} \sum_{e\in A_P,s} (f_e^s y_e^s + c_e^s x_e^s) + \sum_{e\in A_I,k} h_e^k v_e^k + \sum_{e\in A_D} C_e w_e \tag{5.10}$$

$$\text{subject to:} \sum_{s,e\in(H_i\cap A_P)} u_e^{sk} + \sum_{e\in(H_i\cap A_I)} v_e^k = \sum_{e\in T_i} d_j^k w_e + v_e^k \quad \forall\ k,\ i\in M \tag{5.11}$$

$$\sum_k \left(\sum_{s,e\in(H_i\cap A_P)} l^k u_e^{sk} + \sum_{e\in(H_i\cap A_I)} v_e^k \right) \le q_i \quad \forall\ \ i\in M \tag{5.12}$$

$$\sum_k p^k u_e^{sk} = x_e^s \quad \forall\ s,e\in A_P \tag{5.13}$$

$$\sum_{e\in H_i} w_e = 1 \quad \forall\ i\in D \tag{5.14}$$

$$\sum_s y_e^s \le 1 \quad \forall\ e\in A_P \tag{5.15}$$

$$u_e^{sk} \le M_e^{sk} y_e^s \quad \forall\ k,s,e\in A_P \tag{5.16}$$

$$b_e^{s-1} y_e^s \le x_e^s \le b_e^s y_e^s \quad \forall\ s,e\in A_P \tag{5.17}$$

$$x_e^s \ge 0,\ \ v_e^k,\ u_e^{sk} \in Z^+,\ w_e,\ y_e^s \in \{0,1\}. \tag{5.18}$$

Note that we can use constraint (5.13) to substitute for the x_e^s variables, but we will include them in the formulation for conceptual simplicity. The redefinition of the flow variables leads to two major changes in the constraints. We have disaggregated constraints (5.4) and (5.9) into (5.13) and (5.16), respectively. We can now bound u_e^{sk} by not only the future demand and the capacity at the merge center, which were the two bounds on u_e^k, but also by the maximum flow of component k on segment s. Therefore, $M_e^{sk} = min(\sum_{e\in A_D,\ t+} d_j^k, \left\lfloor \frac{q_i}{l^k} \right\rfloor, \left\lfloor \frac{b_e^s}{p^k} \right\rfloor)$. The forcing constraint, (5.16), is tighter than (5.9) both because $M_e^{sk} \le M_e^k$ and because we now have a constraint for each segment, rather than a single constraint over the sum of segments. Although this disaggregated formulation adds significantly to both the number of variables and constraints, it provides a tighter linear programming bound.

In the previous formulation, we disaggregated the flow variables by segment. We could also consider disaggregating by demand point; that is, we could define the flow variables, u_e^{kd}, as the unit flow of component k destined for demand node d, a particular customer with demand

on a particular day. The forcing constraints for this disaggregation are:

$$u_e^{kd} \leq M_e^{kd} \sum_s y_e^s \qquad k, d, e \in A_P. \tag{5.19}$$

To define M_e^{kd}, instead of considering the total future demand of component k as we did in the aggregated model, we need only to consider d^k, the demand for component k for this demand node. Therefore, $M_e^{kd} = min(d_j^k, \lfloor \frac{q_i}{l^k} \rfloor)$.

This disaggregation, however, can weaken the formulation if we are not careful in its implementation. Recall that $M_e^k = min(\sum_{j \in D,\, t+} d_j^k, \lfloor \frac{q_i}{l^k} \rfloor)$. Notice that if $M_e^k = M_e^{kd} = \lfloor \frac{q_i}{l^k} \rfloor$, then (5.9) is actually stronger than (5.19). If we simply replace (5.9) with (5.19), we might weaken the formulation. We therefore need to include the original forcing constraints in the new formulation to assure that we do not lose anything when we disaggregate. This disaggregation of the u variables can then lead to a stronger formulation.

Another possibility is to combine the two prior disaggregations and define u_e^{skd}, the unit flow on segment s, of component k, destined for d. In this disaggregation we include forcing constraints of the form:

$$u_e^{skd} \leq M_e^{skd} y_e^s \qquad s, k, d, e \in A_P. \tag{5.20}$$

We can define $M_e^{skd} = min(d_j^k, \lfloor \frac{q_i}{l^k} \rfloor, \lfloor \frac{b_e^s}{p^k} \rfloor)$. Again, we do not want to just replace (5.16) with (5.20). If $M_e^{sk} = M_e^{skd}$, (5.16) is stronger than (5.20), so we should include both sets of constraints in the formulation. If we do, this further disaggregation of the u variables can then lead to an even stronger formulation.

We have now examined three possible levels of aggregation for the u variables. We can similarly disaggregate the x variables. We can consider disaggregating by component, defining x_e^{sk}, the weight of the flow of component k if the total flow lies in segment s. We can also further disaggregate by demand point and define x_e^{skd}. For each of these disaggregations we can add forcing constraints of the form $x_e \leq P_e y_e^s$, using an appropriate bound P_e on the flow variable. If $P_e < b_e^s$, these forcing constraints will improve upon (5.7).

With multiple definitions of both the unit flow variables and the weight variables, we can define numerous formulations. The task, therefore, is to find the formulation that is tight,

but where size is not unwieldy. The choice of disaggregation level will depend on the size of the underlying application and the choice of solution algorithm.

5.7 Five Valid Formulations

As we have seen, we can employ several levels of disaggregation to model the merge-in-transit problem. To solve the problem, we consider five formulations. We label each with two letters. The first denotes how we define a commodity. We can either define a commodity by component (A), or we can disaggregate by destination (D). The second letter indicates whether we include no forcing constraints (N), aggregated forcing constraints (A), or disaggregated forcing constraints (D) which are disaggregated by segment. The five formulations we consider then are:

(1) AN: This is the most basic formulation, including constraints (5.2)-(5.8), with no valid inequalities. The flow variables are of the form u^k.

(2) AA: We add aggregated forcing constraints, (5.9), to AN.

(3) AD: We add flow variables, u^{sk}, disaggregated by segment and add disaggregated forcing constraints, (5.16), to AA.

(4) DA: We disaggregate the flow variables by destination, introducing the variables u^{kd}, and add aggregated forcing constraints for these variables, (5.19), to AD.

(5) DD: We add flow variables, u^{skd}, disaggregated by segment and add disaggregated forcing constraints, (5.20) to DA.

Notice that for each subsequent formulation, we are adding to the previous one. In this way, the resulting formulation might include redundant constraints, but we are assured that we do not weaken the formulation by replacing forcing constraints with weaker ones, as discussed in Section 3.10. If we let $Z(XX)$ denote the optimal value of the linear relaxation of formulation XX, this strategy assures us of the following:

$$Z(AN) \leq Z(AA) \leq Z(AD) \leq Z(DA) \leq Z(DD) \tag{5.21}$$

5.8 The Solution Algorithm

The large size of industrial merge-in-transit applications significantly complicates the task of solving the problem. As an example, a real network for a computer company has ten components, each produced at one of eight plants, five products, ten merge centers, and 73 customers. Over a time horizon of seven days, the simplest model, AN, consisting only of constraints (5.2)-(5.8), contains more than 100,000 variables (most are integer) and over 60,000 constraints. This problem size is well beyond the capabilities of commercial integer programming solvers. For the disaggregated formulations, even solving the complete linear relaxation is problematic. For these reasons, we developed a solution technique that employs relaxations, cutting plane approaches using both column and row generation, a branch-and-bound procedure, and heuristics to generate quality lower bounds and near-optimal solutions.

5.8.1 Generating Quality Lower Bounds

With all five formulations we consider, the number of forcing constraints is prohibitively large when we consider a realistically sized network. In addition, most of these constraints will be inactive at the optimal solution. Similarly, each disaggregation introduces a large number of variables, most of which will have an optimal value of zero. To address both these issues, we implement an iterative row and column generation technique to solve linear relaxations of the formulations to obtain lower bounds for the problem. The steps of the algorithm are outlined as follows:

1. We initially solve the linear relaxation of the problem consisting of constraints (5.1)-(5.6). This is formulation AN without the segment bound constraints. We then implement an iterative cutting plane approach that adds violated segment bound constraints until we have an optimal solution to the relaxation of AN.

2. We then examine this solution and determine which of the forcing constraints in AA this solution violates (this separation problem is easy to solve) and iteratively add these constraints and solve the resulting linear relaxations. Once we cannot find additional violated forcing constraints, we return to the initial relaxation from Step 1 and look for more violated segment bound constraints and resolve the problem. We continue in this

manner until we have successfully solved the linear relaxation of AA.

3. We can then consider moving to formulation AD. We can develop necessary conditions for the disaggregated forcing constraints (5.16) to be possibly violated. For example, one necessary condition for a disaggregated forcing constraint to be violated is to have flow on the arc, $\sum_s x^s > 0$. In addition, if the flow on that particular arc is split between segments (that is, $y^s > 0$ for two or more y variables), or there exists a commodity k such that $M^k > \left\lfloor \frac{b^s}{p^k} \right\rfloor$, then the disaggregated forcing constraints might be violated. We examine the solution to AA and use these conditions to determine which forcing constraints the current solution is likely to violate. In order to add these constraints, however, we also need to define and add the disaggregated flow variables that appear in these constraints. We are, therefore, integrating a dynamic variable generation strategy within the cutting-plane approach. In this way, we are not only limiting the number of constraints we add, we are also limiting the number of variables that we need to include in the formulation at each stage. Once again, we employ the cutting plane technique to iteratively solve the relaxation of AD.

4. Once we have a solution to AD, we can repeat the process of Step 3 to solve for a solution to DA, and then repeat it again to solve for a solution to DD. We can also choose to stop the procedure after solving any interim formulation.

At the completion of this process we have an optimal solution to whichever relaxation we chose to solve, but we have solved it with only a subset of the variables and constraints defined in the full formulation. We also periodically clean-up the interim models by removing constraints and variables that we believe will no longer be needed in future iterations. This clean-up procedure maintains a manageable problem size.

5.8.2 Generating Feasible Upper Bounds

In addition to using this row and column generation procedure to efficiently obtain quality lower bounds, we implement two heuristic procedures for finding feasible (integral) solutions. In the first, the w-integral heuristic, we aim to find a solution that is w-integral, meaning that the w variables assume integral values, and we, therefore, satisfy the single shipment

constraints. In this procedure we round up those w variables whose fractional value exceeds a pre-specified threshold. We then solve the linear relaxation with these fixed variables, again using the cutting-plane procedure. We iteratively fix variables and re-solve the linear relaxation in this way until we either obtain a w-integral solution or we violate one of the merge-center capacity constraints, in which case the heuristic does not find a feasible solution. If the w-integral heuristic finds a feasible solution, we then invoke the (u, v)-integral heuristic. The (u, v)-integral heuristic uses a w-integral solution as input and again attempts to apply a rounding procedure to construct an all-integral solution that satisfies all the constraints. This rounding heuristic examines each time period sequentially and first rounds the flow variables to the nearest integer. It then rounds the inventory variables and adjusts the total flow so as to satisfy the flow balance and, hopefully, the capacity constraints.

We can invoke these heuristics at any iteration of the cutting-plane procedure. Each call of the heuristics can yield us an integer feasible solution, which is valuable, but calling the heuristics too often can be too time consuming. In our implementation, we invoke the heuristics whenever the current lower bound has improved significantly (as defined by a user-specified threshold) over the one used for the previous call of the heuristics. The solution procedure maintains the best feasible solution found with these heuristics, and its objective value, in memory.

After applying the cutting-plane procedure, with its embedded rounding heuristics, we use the resulting formulation in a branch-and-bound procedure. We branch only on the binary w variables to obtain w-integral solutions. For each w-integral solution, we again apply the (u, v)-integral heuristic to construct an all-integral solution. At the end of this branch-and-bound process, we obtain a new lower bound which corresponds to the optimal solution of the problem when we relax only the integrality restrictions on the u and v variables. We can use this lower bound to assess the quality of our best feasible solution obtained through the rounding heuristic.

5.9 Computational Results

The algorithm is designed to find both quality lower bounds and feasible solutions in an acceptable running time. Our objectives in testing the algorithm are two fold: (1) to compare

the relative performance of the different formulations, and (2) to study the complexity of solving instances with different demand patterns and time horizons. We performed these computations at the University of Montreal on a Sun Ultra 10/300 workstation.

The large-scale application that we wish to solve has nine sources producing ten components, ten merge centers, and 73 customers. As mentioned, even the fully aggregated formulation, without the forcing constraints, results in a model with over 100,000 variables and 60,000 constraints.

To calibrate the algorithm and gain insight into the effects of various modeling and algorithm characteristics, we have performed computational testing on smaller networks. We have generated 16 test instances as shown in Table 5.1. We use the following notation: $|N_S|$ = number of sources, $|N_M|$ = number of merge centers, $|N_C|$ = number of customers, $|K|$ = number of components, $|P|$ = number of products, $|T|$ = number of time periods, ω = the proportion of all possible demand points (combination of location and day) with some positive demand for at least one product, I_δ = interval on which product demands are randomly and uniformly generated for each destination. As the table shows, although these problems are smaller than the large-scale problem defined above, they are not 'toy' problems. In fact, the second half of the problems vary from the large-scale instance only in the number of customers.

Table 5.2 provides results on the relative strength of the five formulations. We have used the iterative column and row generation technique to solve the linear relaxation of each formulation. The column labeled ΔZ^L reports the average (over the 16 instances) percent difference between the optimal value of the linear programming relaxation of each formulation with that of the tightest formulation, that is DE. The column labeled ΔZ^U reports the average (over the 16 instances) percent difference between the optimal value of the linear programming relaxation of each formulation and the best upper bound obtained through the rounding heuristics (before the branch and bound phase) performed with this formulation. The times provided are for executing both the cutting plane procedure and the heuristics.

These results show that disaggregating the variables is quite effective in generating both better lower bounds and better upper bounds, but at the expense of significant increases in computational time. We also note that disaggregating the forcing constraints yields only small improvements in the bounds, but increases the computation time considerably. From these

Problem	$\|N_S\|$	$\|N_M\|$	$\|N_C\|$	$\|K\|$	$\|P\|$	$\|T\|$	ω	I_δ
P1	5	5	20	5	5	3	0.2	[5,35]
P2	5	5	20	5	5	3	0.2	[5,15]
P3	5	5	20	5	5	3	0.8	[5,35]
P4	5	5	20	5	5	3	0.8	[5,15]
P5	5	5	20	5	5	5	0.2	[5,35]
P6	5	5	20	5	5	5	0.2	[5,15]
P7	5	5	20	5	5	5	0.8	[5,35]
P8	5	5	20	5	5	5	0.8	[5,15]
P9	9	10	40	10	10	3	0.2	[5,35]
P10	9	10	40	10	10	3	0.2	[5,15]
P11	9	10	40	10	10	3	0.8	[5,35]
P12	9	10	40	10	10	3	0.8	[5,15]
P13	9	10	40	10	10	5	0.2	[5,35]
P14	9	10	40	10	10	5	0.2	[5,15]
P15	9	10	40	10	10	5	0.8	[5,35]
P16	9	10	40	10	10	5	0.8	[5,15]

Table 5.1: Set of Small-Scale Instances

Formulation	ΔZ^L	ΔZ^U	**CPU (s)**
AN	6.23	21.91	25
AA	6.17	21.19	32
AD	6.16	21.19	93
DA	0.44	10.39	3707
DD	0.00	10.10	10511

Table 5.2: Relative Performance of Each Formulation

			Cutting Plane			Branch & Bound	
				CPU			
Prob	ω	I_δ	% Gap	Heuristic	Total	% Gap	CPU
U_1	0.2	[5,15]	7.2	14044	24196	7.2	14483
U_2	0.2	[5,35]	7.3	8676	19331	4.6	10344
U_3	0.8	[5,15]	3.8	13453	26981	2.9	8059
U_4	0.8	[5,35]	3.4	12983	29589	2.0	10537
C_1	0.2	[5,15]	5.8	18502	28517	3.7	14420
C_2	0.2	[5,35]	6.4	7943	18382	4.7	12032
C_3	0.8	[5,15]	3.8	13887	27612	2.0	8654
C_4	0.8	[5,35]	4.5	22415	38656	3.1	14420

Table 5.3: Results on the Large-Scale Network

observations, we conclude that solving DA or DD will be the most effective for deriving quality solutions to the merge-in-transit problem, but solving DA is probably more computationally tractable on large instances.

We now consider the large-scale problem with nine sources producing ten components, ten merge centers, and 73 customers. For testing on this network, we generate eight instances, as shown in Table 5.3; four with merge-center capacities and four without. For each subset of instances, we choose $\omega \in \{0.2, 0.8\}$ and $I_\delta = [5, 35]$ or $I_\delta = [5, 15]$. We feel these values reflects situations with both small demands and large demands. We found that when demands were larger than these, the problems were easier to solve. Therefore, these four demand patterns reflect more interesting situations. Table 5.3 provides the results on instances with five time periods. In all these runs, we solve the DA formulation and limit the size of the branch and bound tree to 1 GB. We show the gaps both before and after the branch and bound so that we can see the added value of this phase of the algorithm. We also breakdown the CPU time to show where the solution procedure is spending the time.

The algorithm requires significant time to (approximately) solve these large problems. One issue, however, is that we do not need to run the branch and bound until completion. In fact, if we run it for only one to two hours, we obtain virtually the same results.

The demand pattern of these instances varied in the number and magnitude of the demands. These results suggest that instances with fewer positive demands (i.e., smaller values of ω) produce larger gaps. This conclusion is inconsistent with the results presented in Figure 4-5 which shows the gap for the D/A formulation remaining steady as the number of commodities

increases. In fact, if we perform similar computations with the problem instances studied in Figure 4-5, keeping the expected demand of each commodity constant (making the scenario more equivalent to these merge-in-transit instances), we find that the relaxation gap for the D/A formulation increases as the number of commodities increase. At this time, the only explanation we have for this discrepancy is that the rounding heuristics are more effective for instances with more commodities and therefore we obtain smaller gaps, even though the linear relaxations might be becoming weaker.

5.10 Summary

Merge-in-transit can be an effective distribution system if customer orders are composed of components produced at geographically dispersed locations. A well managed merge-in-transit system can help a company achieve both low transportation costs and reduced inventory levels. An operational model like the one we developed can be an important tool for implementing an efficient merge-in-transit system.

The basic formulation, given by (5.1)-(5.8), is a candidate for tightening through disaggregating the flow variables. As discussed in Chapter 3, this disaggregation allows us to add forcing constraints that improve the lower bounds provided by the linear relaxation. Using these techniques, we have defined a sequence of five valid formulations, each one improving upon the previous one.

A tighter formulation, however, quickly becomes too large to solve by traditional means. We, therefore, develop an algorithm that solves each formulation sequentially, using iterative and dynamic row and column generation. Combined with rounding heuristics and branch and bound, this algorithm attempts to find both good lower bounds as well as quality integer feasible solutions. Computational experiments with this algorithm suggest that we can consistently find solutions within 5% of optimal, and frequently solutions within 3%. The complex formulations contain over 300,000 variables (the vast majority of them integer) and constraints and the CPU requirements are quite high (generally on the order of 8-10 hours). Since the models are likely to be run at most daily, this type of computational performance should permit firms to use this modeling and solution approach to improve operations in the emerging new merge-in-transit approach to supply chain management.

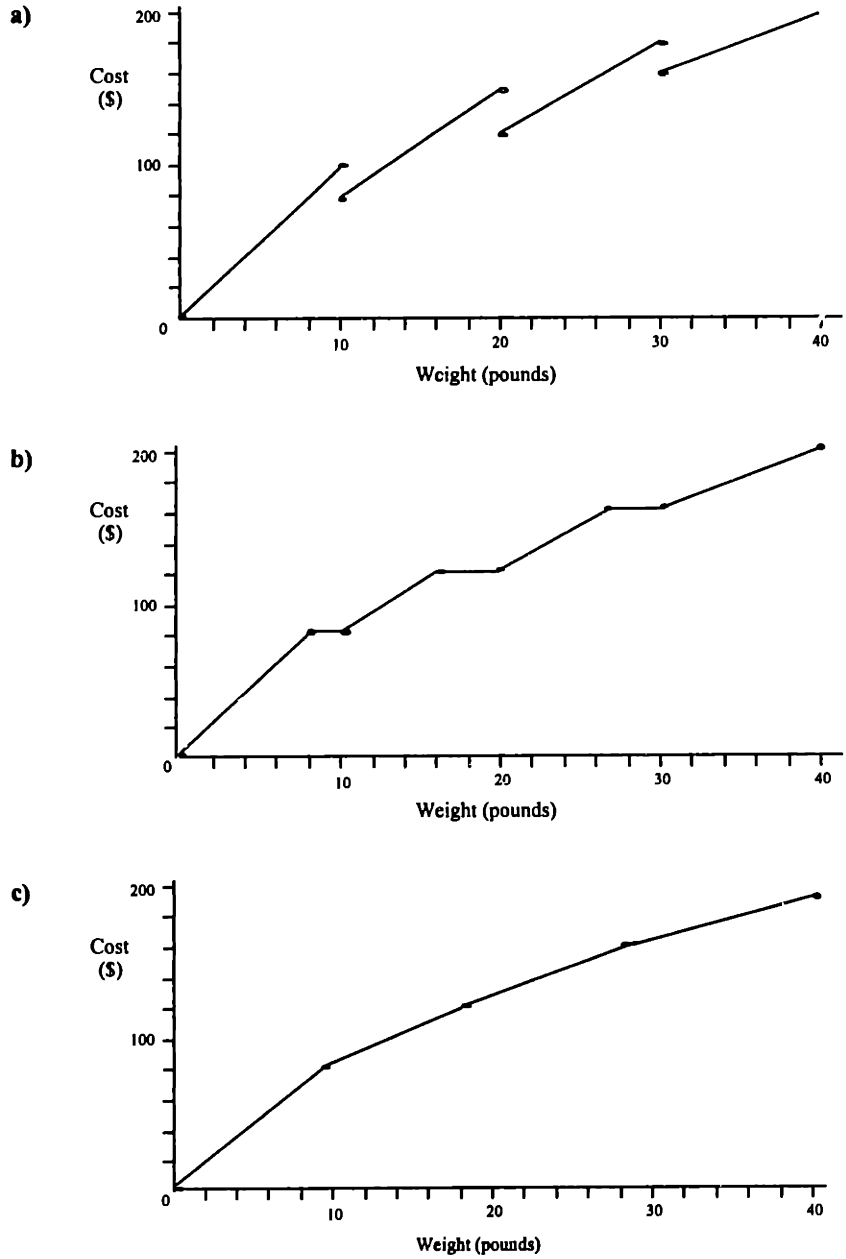

Figure 5-5: Possible Approximations for the MIT Cost Functions

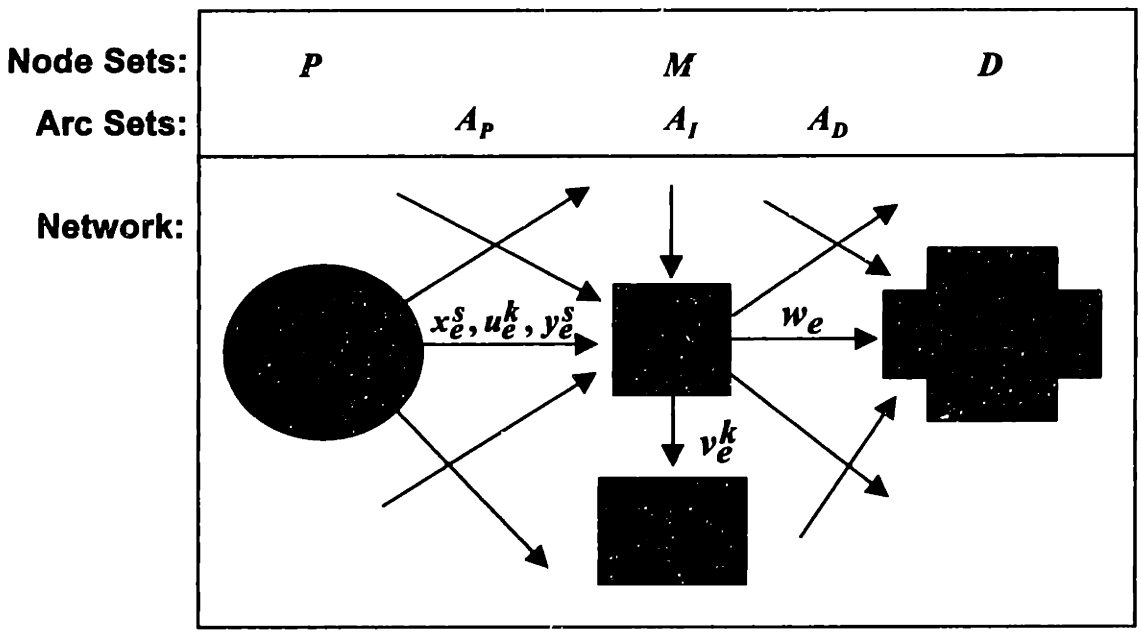

Figure 5-6: The Time and Mode Expanded Network

Chapter 6

Contributions and Ongoing Research

This thesis has examined optimization problems with general separable piecewise linear cost functions. It has focused on network flow models and a topical application, namely the design and operation of a merge-in-transit system. The research has three components: modeling and theoretical insight into a general problem and modeling approach, specific application of this insight into modeling a real-world problem, and algorithm development and testing. We believe this research contributes both through its theoretical components and through its application to the merge-in-transit problem.

6.1 Contributions

We view our contributions thus far to include:

- Modeling problems with piecewise linear cost functions. Although the three models presented are commonly found in the literature, to our knowledge no one had formally shown that the linear programming relaxations of them are equivalent and, in fact, approximate the actual cost function with its lower convex envelope. Although we have focused on network flow models, these modeling approaches can be used for any problem with separable piecewise linear costs.

- Effects of disaggregation. We have shown how several problems can be disaggregated to produce formulations with tighter linear programming relaxations. For models with concave costs, we have examined the structural effects of disaggregation and provided an exact representation of the solution to the linear relaxation of the disaggregated formulation for any flow value. We proved that, in general, if k is the dimensionality of the disaggregation, then the disaggregated model approximates the actual cost function with the lower convex envelope in $k+1$ dimensions. We also developed results for the disaggregated/aggregated model.

- Limitations of disaggregation. We have shown that disaggregation can improve a formulation. It seems intuitive that if it does not help, it at least will not hurt. As we have seen, however, if we disaggregate without maintaining the aggregated structure, we can indeed weaken the formulation. We believe this is a subtle point that is often overlooked.

- Effectiveness of disaggregation. We have shown, through computational results, that disaggregation can be very effective, particularly with concave cost structures. Although the problem with concave costs is not integral in general, in the instances that we examined, the solution to the linear programming relaxation of the disaggregated formulation was always integral. For general cost functions, the disaggregated model does not generally solve with integral solutions, but the solutions are considerably tighter than the solutions to the aggregated model. For example, a typical instance of the aggregated formulation might have a relaxation gap of 80% but the disaggregated formulation will bring this gap to within 1%. As the network grows, or the number of segments increases, however, solving the disaggregated relaxations becomes computationally difficult, often taking over an hour just so solve the linear relaxation.

- Uses of disaggregation. We applied disaggregation to five problems: the network flow problem with a single origin or destination, the multi-commodity network flow problem, the facility location problem with multiple capacity options, the network loading problem, and the merge-in-transit problem. In each of these cases, we were able to develop a disaggregated formulation that is tighter than the traditional aggregated model.

- Modeling and solving the merge-in-transit problem. The business concept of merge-in-

transit is relatively new and many companies are struggling with its complexities. A model like the one we develop could aid these companies in making operational decisions that use the system efficiently while meeting the specified service requirements.

- Developing algorithms to solve large and complex mixed-integer programs. We have combined several common algorithmic ideas into an effective algorithm to solve a very large and combinatorially complex mixed-integer problem. In doing so, we have exploited very little that is problem specific. As a result, success with the merge-in-transit problem suggests potential success with similar approaches to other large and complex problems with piecewise linear objective functions.

6.2 Potential Future Research

The research to date provides insight into the general nature of this work. Ideas for future research include:

- Further investigate alternative approaches to solving the linear relaxations of the disaggregated models for the general network flow problem. For some applications, even solving the linear relaxation of the disaggregated formulation is too computationally difficult. We have tried implementing Lagrangean relaxations and cutting-plane methods, but these methods were not successful. Although the row and column generation technique that we implemented was effective for the merge-in-transit problem, it would be worthwhile to further investigate alternative approaches. It would be particularly interesting to see if we can use geometric insight concerning the disaggregated formulation to solve these relaxations more efficiently.

- Consider methods for finding good integer, feasible solutions to general network flow problems with piecewise linear cost functions. These methods might be other heuristics, or a better implementation of branch and bound.

- Apply disaggregation and the simultaneous row and column generation approach to other problems. We feel this modeling and algorithmic approach might be effective for other

problems, including many network design applications. Specifically, we want to test these approaches on the network loading problem.

Several extensions of the merge-in-transit problem might increase its applicability. Many of these ideas arose from discussions with corporate executives from a variety of companies and industries.

- Make merging optional, i.e., allow shipments to go directly between the manufacturing facility and the customer if that is more cost effective.
- Allow time at the merge-centers for additional processing, e.g. delayed production, customization, etc.
- Include limited supplies at manufacturing facilities, or production time for make-to-order systems, i.e., don't assume supply is unlimited and always ready at the manufacturing facilities.
- Allow early or late arrivals with penalties.

Bibliography

[1] Ahuja, R.K., T.L. Magnanti and J.B. Orlin (1993), *Network Flows: Theory, Algorithms and Application.* Prentice-Hall.

[2] Arntzen, B.C., G.G. Brown, T.P. Harrison and L.L. Tafton (1995), Global Supply Chain Management at Digital Equipment Corporation. *Interfaces* 25, pp. 69-93.

[3] Balakrishnan, A. and S. Graves (1989), A Composite Algorithm for a Concave-Cost Network Flow Problem. *Networks* 19, pp. 175-202.

[4] Balakrishnan, A., T.L. Magnanti, and P. Mirchandani (1997), Network Design, *Annotated Bibliographies in Combinatorial Optimization.* Edited by M. Dell'Amico, F. Maffioli, and S. Martello, John Wiley and Sons.

[5] Bradley, S.P., A.C. Hax, and T.L. Magnanti (1977), *Applied Mathematical Programming.* Addison Wesley.

[6] Bramel J. and D. Simchi-Levi (1997), *The Logic of Logistics*, Springer-Verlag.

[7] Breitman, R.L. and J.M. Lucas (1987), PLANETS: A Modeling System for Business Planning. *Interfaces* 17, pp. 94-106.

[8] Chan, L., A. Muriel and D. Simchi-Levi (1997), Supply Chain Management: Integrating Inventory and Transportation. Northwestern University.

[9] Cole, M.H. and M. Parthasarathy (1998), Design of Merge-in-Transit Logistics Networks. *Proceedings of "Rensselaer's International Conference on Agile, Intelligent, and Computer-Integrated Manufacturing,"* October, 1998 Troy, NY.

[10] Cominetti, R. and F. Ortega (1997), A Branch & Bound Method for Minimum Concave Cost Network Flows Based on Sensitivity Analysis. Universidad de Chile.

[11] Crainic, T.G. and G. Laporte (1997), Planning Models for Freight Transportation. *European Journal of Operational Research* 97, pp. 409-438.

[12] Dawe, R.L. (1997), Move It Fast...Eliminate Steps. *Transportation and Distribution* 38, Number 9, pp. 67-70.

[13] Epstein, R. (1998), Linear Programming and Capacitated Network Loading, Ph.D. Thesis, Operations Research Center, MIT.

[14] Geoffrion, A. (1977), A Priori Error Bounds for Procurement Commodity Aggregation in Logistics Planning Models. *Naval Research Logisics. Quarartly* 24, pp. 201-212.

[15] Geoffrion, A. and G. Graves (1974), Multicommodity Distribution System Design by Benders Decomposition. *Management Science* 29, pp. 822-844.

[16] Hallefjord, A. and S. Storoy (1990), Aggregation and Disaggregation in Integer Programming Problems. *Operations Research* 38, pp. 619-623.

[17] Hastings, P. (1998), Manufacturing on the Move. *Cargovision*, September 1998, http://www.cargovision.com/magazine3/manufacturing.html

[18] Holmberg, K. (1994), Solving the Staircase Cost Facility Location Problem with Decomposition and Piecewise Linearization. *European Journal of Operational Research* 75, pp. 41-61.

[19] Holmberg, K. and J. Ling (1997), A Lagrangean Heuristic for the Facility Location Problem with Staircase Costs. *European Journal of Operational Research* 97, pp. 63-74.

[20] Hunt-Wesson Foods, Inc.: A Case Study. Harvard Business School, 1978.

[21] Lambert, D.M., M.C. Cooper and J.D. Pagh (1998), Supply Chain Management: Implementation Issues and Research Opportunities. *The International Journal of Logistics Management*, pp. 1-19.

[22] Magnanti T.L. and R.T. Wong (1984), Network Design and Transportation Planning: Models and Algorithms. *Transportation Science* 18, pp. 1-55.

[23] Martin, C.H., D.C. Dent and J.C. Eckhart (1993), Integrated Production, Distribution, and Inventory Planning at Libbey-Owens-Ford. *Interfaces* 23, pp. 68-78.

[24] Mendelssohn, R. (1980), Improved Bounds for Aggregated Linear Programs. *Operations Research* 18, pp. 1450-1453.

[25] Nemhauser, G.L. and L.A. Wolsey (1988), *Integer and Combinatorial Optimization.* Wiley.

[26] Popken, D. (1994), An Algorithm for the Multiattribute, Multicommodity Flow Problem with Freight Consolidation and Inventory Costs. *Operations Research* 42, pp. 274-286.

[27] Robinson, E.P., L. Gao and S.D. Muggenborg (1993), Designing an Integrated Distribution System at DowBrands, Inc. *Interfaces* 23, pp. 107-117.

[28] Shapiro J.F. (1993), Chapter 8 in *The Handbook in Operations Research in Management Science, Vol 4: Logistics of Production and Inventory.* Edited by S.C. Graves, A.H.G. Rinnooy Kan, P.H. Zipking, Elsevier Science Publishers.

[29] Zipkin, P. (1980), Bounds on the Effect of Aggregating Variables in Linear Programs. *Operations Research* 28, pp. 403-417.

[30] Zipkin, P. (1980), Bounds for Row-Aggregation in Linear Programming. *Operations Research* 28, pp. 903-916.

Appendix A

A Translation Between Formulations

In Section 3.5, we provided three valid formulations for modeling piecewise linear costs and then showed that the linear relaxation of each approximates the true cost with the lower convex envelope of the cost function. We can also show their equivalency by providing a translation between extreme point solutions to each formulation. Here, we show that for a given extreme point solution to one formulation, we can derive a solution to either of the other formulations with the same cost. We do so by showing how to translate between the multiple choice model and either the incremental or convex combination models. In this discussion, we are considering a single arc with total flow F. The linear relaxations of the formulations, then, can be written as:

Incremental Model:

$$\begin{aligned} &\textit{minimize} \sum_s c^s w^s + \widehat{f}^s v^s \\ &\textit{subject to:} \quad \sum_s w^s = F \\ &(b^s - b^{s-1}) v^{s+1} \leq w^s \leq (b^s - b^{s-1}) v^s \\ &w^s \geq 0, \quad 0 \leq v^s \leq 1. \end{aligned}$$

Multiple Choice Model:

$$\begin{aligned} \textit{minimize} \sum_s c^s x^s + f^s y^s & \\ \textit{subject to}: \sum_s x^s &= F \\ b^{s-1} y^s \leq x^s &\leq b^s y^s \\ \sum_s y^s &\leq 1 \\ x^s \geq 0, \quad 0 \leq y^s &\leq 1. \end{aligned}$$

Convex Combination Model:

$$\begin{aligned} \textit{minimize} \sum_s \mu^s (c^s b^{s-1} + f^s) + \lambda^s (c^s b^s + f^s) & \\ \textit{subject to}: \sum_s (\mu^s b^{s-1} + \lambda^s b^s) &= F \\ \mu^s + \lambda^s &= y^s \\ \sum_s y^s &\leq 1 \\ \mu^s, \lambda^s \geq 0, \quad 0 \leq y^s &\leq 1. \end{aligned}$$

A.1 Multiple Choice and Convex Combination Models

We show that any feasible solution to one of these formulation can be translated into a feasible solution to the other with the same cost.

A.1.1 Multiple Choice $\longrightarrow$ Convex Combination

Consider a feasible solution, (x^s, y^s), to the multiple choice model. For this solution to be feasible, $b^{s-1} y^s \leq x^s \leq b^s y^s$ which implies that for some $0 \leq \alpha^s \leq 1$, $x^s = \alpha^s b^{s-1} y^s + (1 - \alpha^s) b^s y^s$. Let $\mu^s = \alpha^s y^s$ and $\lambda^s = (1 - \alpha^s) y^s$. Then $\mu^s + \lambda^s = y^s$ and $x^s = \mu^s b^{s-1} + \lambda^s b^s$. Moreover, the constraint $\sum_s x^s = F$ implies that $\sum_s (\mu^s b^{s-1} + \lambda^u b^s) = F$. Therefore, (y^s, μ^s, λ^s) is feasible for the convex combination formulation. Moreover, the cost of this solution is $\sum_s \mu^s (c^s b^{s-1} + f^s) + \lambda^s (c^s b^s + f^s) = \sum_s c^s (\mu^s b^{s-1} \lambda^s b^s) + f^s (\mu^s + \lambda^s) = \sum_s c^s x^s + f^s y^s$ which is equal to the cost of the solution to the multiple choice formulation.

A.1.2 Convex Combination $\longrightarrow$ Multiple Choice

Consider a feasible solution (y^s, μ^s, λ^s) to the convex combination model. Define $x^s = \mu^s b^{s-1} + \lambda^s b^s$. The conditions $b^{s-1} \leq b^s$ and $\mu^s + \lambda^s = y^s$ imply that $b^{s-1} y^s \leq x^s \leq b^s y^s$. Therefore, (x^s, y^s) is feasible for the multiple choice formulation. As shown above, the cost of this solution is the same as the cost of (y^s, μ^s, λ^s).

A.2 Multiple Choice and Incremental Models

Here, we start with an extreme point solutions of one formulation and derive a feasible solution to the other with the same cost. We will then argue that if we have a mapping for extreme point solutions, we indeed have a mapping for all feasible solutions. This translation is more complex than the previous one because the flow and binary variables have very different interpretations in each formulation. We let $\Delta^s = b^s - b^{s-1}$ to simplify the notation. Recall that in the proof of Proposition 1 in Chapter 3 we showed the following results:

Corollary 2 *At an extreme point of the linear programming relaxation of the multiple choice formulation, the set of y variables takes either a form where one $y^s > 0$ (when one of the endpoints corresponds to the origin), or two $y^s > 0$ and their sum is one.*

Corollary 3 *At an extreme point of the linear programming relaxation of the incremental formulation, the set of y variables will either be of the form $(\overline{y}, \overline{y}, ..., \overline{y}, 0, 0, ..., 0)$ (when one of the endpoints is the origin) or $(1, 1, ..., 1, \overline{y}, \overline{y}, ..., \overline{y}, 0, 0, ..., 0)$, for some constant $0 \leq \overline{y} \leq 1$.*

A.2.1 Multiple Choice $\longrightarrow$ Incremental

Consider an extreme point solution (x^s, y^s) to the multiple choice model. Corollary 2 provides us with two distinct cases for extreme points of the multiple choice formulation; either one y^s variable is positive or two are positive and their sum is one.

CASE 1: Only one y^s variable is positive, that is, $y^{\widehat{s}} > 0$, and $y^s = 0, \forall\ s \neq \widehat{s}$. In this case $x^{\widehat{s}} = F$, and $x^s = 0, \forall\ s \neq \widehat{s}$. We also know that in an extreme point solution, either $y^{\widehat{s}} = \frac{x^{\widehat{s}}}{b^{\widehat{s}-1}}$ or $y^{\widehat{s}} = \frac{x^{\widehat{s}}}{b^{\widehat{s}}}$.

Let (w^s, v^s) be a solution to the incremental model, and in particular, let $v^s = y^{\widehat{s}}, \forall\ s \leq \widehat{s}$ and $v^s = 0, \forall\ s > \widehat{s}$. If $y^{\widehat{s}} = \frac{x^{\widehat{s}}}{b^{\widehat{s}}}$, let $w^s = \Delta^s y^{\widehat{s}} = \Delta^s \frac{F}{b^{\widehat{s}}}, \forall\ s \leq \widehat{s}$ and $w^s = 0, \forall\ s > \widehat{s}$. If

$y^{\hat{s}} = \frac{x^{\hat{s}}}{b^{\hat{s}-1}}$, let $w^s = \Delta^s y^{\hat{s}} = \Delta^s \frac{F}{b^{\hat{s}-1}}$, $\forall\ s \leq \hat{s}-1$ and $w^s = 0$, $\forall\ s \geq \hat{s}$. Notice that for all s, $v^s \geq v^{s+1}$.

We need to first show that (w^s, v^s) is feasible for the incremental model.

1. If $y^{\hat{s}} = \frac{x^{\hat{s}}}{b^{\hat{s}}}$, then $\sum_s w^s = \sum_{s \leq \hat{s}} \Delta^s \frac{F}{b^{\hat{s}}} = \frac{F}{b^{\hat{s}}} \sum_{s \leq \hat{s}} \Delta^s = F$, as required by the first constraint.

   If $y^{\hat{s}} = \frac{x^{\hat{s}}}{b^{\hat{s}-1}}$, then $\sum_s w^s = \sum_{s \leq \hat{s}-1} \Delta^s \frac{F}{b^{\hat{s}-1}} = \frac{F}{b^{\hat{s}-1}} \sum_{s \leq \hat{s}-1} \Delta^s = F$, as required by the first constraint.

2. If $y^{\hat{s}} = \frac{x^{\hat{s}}}{b^{\hat{s}}}$, then for $s \leq \hat{s}$, $w^s = \Delta^s y^{\hat{s}} = \Delta^s v^s$, and since $v^s \geq v^{s+1}$, $w^s \geq \Delta^{s+1} v^s$. For $s > \hat{s}$, $w^s = 0$, $v^s = 0$, and $v^{s+1} = 0$. Therefore, we satisfy the second constraint.

   If $y^{\hat{s}} = \frac{x^{\hat{s}}}{b^{\hat{s}-1}}$, then for $s \leq \hat{s}-1$, $w^s = \Delta^s y^{\hat{s}} = \Delta^s v^s$, and since $v^s \geq v^{s+1}$, $w^s \geq \Delta^{s+1} v^s$. For $\hat{s}$, $w^{\hat{s}} = 0$, $v^{\hat{s}} = y^{\hat{s}}$ and $v^{\hat{s}+1} = 0$. Therefore, $\Delta^{\hat{s}} v^{\hat{s}+1} \leq w^{\hat{s}} \leq \Delta^{\hat{s}} v^{\hat{s}}$. For $s > \hat{s}$, $w^s = 0$, $v^s = 0$ and $v^{s+1} = 0$. Therefore, we satisfy the second constraint.

3. $0 < y^{\hat{s}} \leq 1$, therefore, $0 \leq v^s \leq 1, .\forall\ s$. Also, $\Delta^s > 0$ and $y^{\hat{s}} > 0$, therefore, $w^s \geq 0$, $\forall\ s$.

We have just shown that this solution is feasible. We now need to show that (x^s, y^s) and (w^s, v^s) have the same cost. Recall that the objective function of the incremental model uses $\hat{f}^s$, which is the gap in the cost function at the beginning of the sth segment. Note that for any $\bar{s}$, $f^{\bar{s}} + c^{\bar{s}} b^{\bar{s}} = \sum_{s \leq \bar{s}} \hat{f}^s + c^s \Delta^s$ and $f^{\bar{s}} + c^{\bar{s}} b^{\bar{s}-1} = \hat{f}^{\bar{s}} + \sum_{s \leq \bar{s}-1} \hat{f}^s + c^s \Delta^s$. Therefore:

a) If $y^{\hat{s}} = \frac{x^{\hat{s}}}{b^{\hat{s}}}$, then Cost $(w^s, v^s) = \sum_s c^s w^s + \hat{f}^s v^s = \sum_{s \leq \hat{s}} c^s \Delta^s \frac{F}{b^{\hat{s}}} + \hat{f}^s \frac{F}{b^{\hat{s}}} = \frac{F}{b^{\hat{s}}} \sum_{s \leq \hat{s}} c^s \Delta^s + \hat{f}^s = \frac{F}{b^{\hat{s}}} \left(f^{\hat{s}} + c^{\hat{s}} b^{\hat{s}}\right) = c^{\hat{s}} F + f^{\hat{s}} \frac{F}{b^{\hat{s}}} = c^{\hat{s}} x^{\hat{s}} + f^{\hat{s}} y^{\hat{s}} =$ Cost (x^s, y^s).

or

b) If $y^{\hat{s}} = \frac{x^{\hat{s}}}{b^{\hat{s}-1}}$, then Cost $(w^s, v^s) = \sum_s c^s w^s + \hat{f}^s v^s = \sum_{s \leq \hat{s}-1} (c^s \Delta^s \frac{F}{b^{\hat{s}-1}} + \hat{f}^s \frac{F}{b^{\hat{s}-1}}) + \hat{f}^{\hat{s}} \frac{F}{b^{\hat{s}-1}} = \frac{F}{b^{\hat{s}-1}} (\hat{f}^{\hat{s}} + \sum_{s \leq \hat{s}-1} c^s \Delta^s + \hat{f}^s) = \frac{F}{b^{\hat{s}-1}} \left(f^{\hat{s}} + c^{\hat{s}} b^{\hat{s}-1}\right) = c^{\hat{s}} F + f^{\hat{s}} \frac{F}{b^{\hat{s}-1}} = c^{\hat{s}} x^{\hat{s}} + f^{\hat{s}} y^{\hat{s}} =$ Cost (x^s, y^s).

Therefore, we can take any extreme point of the multiple choice formulation that doesn't split flow between segments and derive a feasible solution to the incremental formulation that has the same cost.

CASE 2: Two y^s variables, $y^{\hat{s}}$ and $y^{\hat{t}}$, are positive and $y^{\hat{s}} + y^{\hat{t}} = 1$. Assume that $\hat{s} < \hat{t}$.

As in Case 1, this implies that $x^{\hat{s}} > 0$ and $x^{\hat{t}} > 0$, and $y^{\hat{s}} = \frac{x^{\hat{s}}}{b^{\hat{s}-1}}$ or $y^{\hat{s}} = \frac{x^{\hat{s}}}{b^{\hat{s}}}$, and $y^{\hat{t}} = \frac{x^{\hat{t}}}{b^{\hat{t}-1}}$ or $y^{\hat{t}} = \frac{x^{\hat{t}}}{b^{\hat{t}}}$.

We now define (w^s, v^s), a solution to the incremental model. Let $v^s = 1, \forall\ s \leq \hat{s}$, $v^s = y^{\hat{t}}$, $\forall\ \hat{s} < s \leq \hat{t}$, and $v^s = 0, \forall\ s > \hat{t}$. Note that again, $v^s \geq v^{s+1}, \forall\ s$. For $s \leq \hat{s} - 1$, let $w^s = \Delta^s$. If $y^{\hat{s}} = \frac{x^{\hat{s}}}{b^{\hat{s}}}$, let $w^{\hat{s}} = \Delta^{\hat{s}}$, but if $y^{\hat{s}} = \frac{x^{\hat{s}}}{b^{\hat{s}-1}}$, let $w^{\hat{s}} = \Delta^{\hat{s}} y^{\hat{t}}$. For $\hat{s} < s < \hat{t}$, let $w^s = \Delta^s y^{\hat{t}}$. If $y^{\hat{t}} = \frac{x^{\hat{t}}}{b^{\hat{t}}}$, let $w^{\hat{t}} = \Delta^{\hat{t}} y^{\hat{t}}$, but if $y^{\hat{t}} = \frac{x^{\hat{t}}}{b^{\hat{t}-1}}$, let $w^{\hat{t}} = 0$. Finally, for $s > \hat{t}$, let $w^s = 0$.

We need to first show that (w^s, v^s) is feasible for the incremental model.

1. If $y^{\hat{s}} = \frac{x^{\hat{s}}}{b^{\hat{s}}}$ and $y^{\hat{t}} = \frac{x^{\hat{t}}}{b^{\hat{t}}}$, then $\sum_s w^s = \sum_{s \leq \hat{s}} \Delta^s + \sum_{\hat{s} < s \leq \hat{t}} \Delta^{\hat{t}} y^{\hat{t}} = b^{\hat{s}} + y^{\hat{t}}(b^{\hat{t}} - b^{\hat{s}}) = (1 - y^{\hat{t}}) b^{\hat{s}} + y^{\hat{t}} b^{\hat{t}} = y^{\hat{s}} b^{\hat{s}} + y^{\hat{t}} b^{\hat{t}} = x^{\hat{s}} + x^{\hat{t}} = F.$

   If $y^{\hat{s}} = \frac{x^{\hat{s}}}{b^{\hat{s}}}$ and $y^{\hat{t}} = \frac{x^{\hat{t}}}{b^{\hat{t}-1}}$, then $\sum_s w^s = \sum_{s \leq \hat{s}} \Delta^s + \sum_{\hat{s} < s < \hat{t}} \Delta^{\hat{t}} y^{\hat{t}} = b^{\hat{s}} + y^{\hat{t}}(b^{\hat{t}-1} - b^{\hat{s}}) = (1 - y^{\hat{t}}) b^{\hat{s}} + y^{\hat{t}} b^{\hat{t}-1} = y^{\hat{s}} b^{\hat{s}} + y^{\hat{t}} b^{\hat{t}-1} = x^{\hat{s}} + x^{\hat{t}} = F.$

   If $y^{\hat{s}} = \frac{x^{\hat{s}}}{b^{\hat{s}-1}}$ and $y^{\hat{t}} = \frac{x^{\hat{t}}}{b^{\hat{t}}}$, $\sum_s w^s = \sum_{s < \hat{s}} \Delta^s + \sum_{\hat{s} \leq s \leq \hat{t}} \Delta^{\hat{t}} y^{\hat{t}} = b^{\hat{s}-1} + y^{\hat{t}}(b^{\hat{t}} - b^{\hat{s}-1}) = (1 - y^{\hat{t}}) b^{\hat{s}-1} + y^{\hat{t}} b^{\hat{t}} = y^{\hat{s}} b^{\hat{s}-1} + y^{\hat{t}} b^{\hat{t}} = x^{\hat{s}} + x^{\hat{t}} = F.$

   If $y^{\hat{s}} = \frac{x^{\hat{s}}}{b^{\hat{s}-1}}$ and $y^{\hat{t}} = \frac{x^{\hat{t}}}{b^{\hat{t}-1}}$, then $\sum_s w^s = \sum_{s < \hat{s}} \Delta^s + \sum_{\hat{s} \leq s < \hat{t}} \Delta^{\hat{t}} y^{\hat{t}} = b^{\hat{s}-1} + y^{\hat{t}}(b^{\hat{t}-1} - b^{\hat{s}-1}) = (1 - y^{\hat{t}-1}) b^{\hat{s}-1} + y^{\hat{t}} b^{\hat{t}-1} = y^{\hat{s}} b^{\hat{s}-1} + y^{\hat{t}} b^{\hat{t}-1} = x^{\hat{s}} + x^{\hat{t}} = F.$

   Therefore, this solution satisfies the first constraint.

2. For $s \leq \hat{s} - 1, w^s = \Delta^s = \Delta^s v^s \geq \Delta^s v^{s+1}$ since $v^s \geq v^{s+1}$.

   For $\hat{s}$, either $w^{\hat{s}} = \Delta^{\hat{s}} = \Delta^{\hat{s}} v^{\hat{s}} \geq \Delta^{\hat{s}} v^{\hat{s}+1}$, or $w^{\hat{s}} = \Delta^{\hat{s}} y^{\hat{t}} = \Delta^{\hat{s}} v^{\hat{s}+1} \leq \Delta^{\hat{s}} v^{\hat{s}}$.

   For $\hat{s} < s < \hat{t}$, $w^s = \Delta^s y^{\hat{t}} = \Delta^s v^s \geq \Delta^s v^{s+1}$

   For $\hat{t}$, either $w^{\hat{t}} = \Delta^{\hat{t}} y^{\hat{t}} = \Delta^s v^s \geq \Delta^s v^{s+1}$, or $w^{\hat{t}} = 0 = \Delta^{\hat{t}} v^{\hat{t}+1} \leq \Delta^{\hat{s}} v^{\hat{s}}$.

   For $s > \hat{t}$, $w^s = 0$, $v^s = 0$, and $v^{s+1} = 0$.

   Therefore, this solution satisfies the second constraint.

3. $0 < y^{\hat{t}} \leq 1$, therefore, $0 \leq v^s \leq 1, \forall\ s$. Also, $\Delta^s > 0$ and $y^{\hat{t}} > 0$, therefore, $w^s \geq 0, \forall\ s$.

Now that we have shown (w^s, v^s) to be feasible, we must show it has the same objective cost as (x^s, y^s). We will show this only for the case $y^{\hat{s}} = \frac{x^{\hat{s}}}{b^{\hat{s}}}$ and $y^{\hat{t}} = \frac{x^{\hat{t}}}{b^{\hat{t}}}$. The other cases are similar.

Cost $(w^s, v^s) = \sum_s c^s w^s + \hat{f}^s v^s = \sum_{s \le \hat{s}} (c^s \Delta^s + \hat{f}^s) + \sum_{\hat{s} < s \le \hat{t}} (c^s \Delta^s y^{\hat{t}} + \hat{f}^s y^{\hat{t}}) = \sum_{s \le \hat{s}} (c^s \Delta^s + \hat{f}^s) + y^{\hat{t}} \sum_{\hat{s} < s \le \hat{t}} (c^s \Delta^s + \hat{f}^s) = (f^{\hat{s}} + c^{\hat{s}} b^{\hat{s}}) + y^{\hat{t}} [(f^{\hat{t}} + c^{\hat{t}} b^{\hat{t}}) - (f^{\hat{s}} + c^{\hat{s}} b^{\hat{s}})] = (1 - y^{\hat{t}})(f^{\hat{s}} + c^{\hat{s}} b^{\hat{s}}) + y^{\hat{t}} (f^{\hat{t}} + c^{\hat{t}} b^{\hat{t}}) = y^{\hat{s}} (f^{\hat{s}} + c^{\hat{s}} b^{\hat{s}}) + y^{\hat{t}} (f^{\hat{t}} + c^{\hat{t}} b^{\hat{t}}) = c^{\hat{s}} b^{\hat{s}} y^{\hat{s}} + c^{\hat{t}} b^{\hat{t}} y^{\hat{t}} + f^{\hat{s}} y^{\hat{s}} + f^{\hat{t}} y^{\hat{t}} = c^{\hat{s}} x^{\hat{s}} + c^{\hat{s}} x^{\hat{s}} + f^{\hat{s}} y^{\hat{s}} + f^{\hat{t}} y^{\hat{t}} =$ Cost (x^s, y^s).

Therefore, we can take any extreme point of the multiple choice formulation that splits flow between two segments and derive a feasible solution to the incremental formulation that has the same cost.

A.2.2 Incremental $\longrightarrow$ Multiple Choice

We will now consider an extreme point (w^s, v^s) of the incremental formulation and derive, (x^s, y^s), a feasible solution for the multiple choice formulation with the same cost. Again, our characterization of the extreme points of the incremental formulation provides us with two cases to consider.

CASE 1: v^s variables have the form $(v, v, ..., v, 0, 0, ..., 0)$. We let $\hat{s}$ denote the largest s such that $v^s = v$. We know that either $v = \frac{w^{\hat{s}}}{\Delta^{\hat{s}}}$ or $v = \frac{w^{\hat{s}-1}}{\Delta^{\hat{s}-1}}$, and for all other s, $v = \frac{w^s}{\Delta^s} = \frac{w^{s-1}}{\Delta^{s-1}}$. To define the solution to the multiple choice model, let $y^{\hat{s}} = v$, $x^{\hat{s}} = F$, and $y^s, x^s = 0$, $\forall$ $s \neq \hat{s}$. We first show that this solution is feasible for the multiple choice model.

1. $\sum_s x^s = x^{\hat{s}} = F$, so this first constraint is satisfied.

2. If $v^{\hat{s}} = \frac{w^{\hat{s}}}{\Delta^{\hat{s}}}$, then $x^{\hat{s}} = F = \sum_s w^s = \sum_s \Delta^s v^s = b^{\hat{s}} y^{\hat{s}} \geq b^{\hat{s}-1} y^{\hat{s}}$, and $\forall$ $s \neq \hat{s}$, $x^s = 0$ and $y^s = 0$, therefore $b^{s-1} y^s \leq x^s \leq b^s y^s$.

   If $v^{\hat{s}} = \frac{w^{\hat{s}-1}}{\Delta^{\hat{s}-1}}$, then $x^{\hat{s}} = F = \sum_s w^s = \sum_s \Delta^s v^{s+1} = b^{\hat{s}-1} v = b^{\hat{s}-1} y^{\hat{s}} \leq b^{\hat{s}} y^{\hat{s}}$, and $\forall$ $s \neq \hat{s}$, $x^s = 0$ and $y^s = 0$, therefore $b^{s-1} y^s \leq x^s \leq b^s y^s$.

   Therefore, the second constraint is satisfied.

3. $\sum_s y^s = y^{\hat{s}} = v$, but $0 \leq v^s \leq 1$, therefore $\sum_s y^s \leq 1$.

4. $F > 0$, therefore, $x^s \geq 0$, and since $0 \leq v^s \leq 1$, $0 \leq y^s \leq 1$.

We now need to show that these two solutions have the same cost. We can save ourselves some work here by noticing that (w^s, v^s) and (x^s, y^s) have the same form as they did in Case 1 above. Therefore, their costs must be the same here as well.

CASE 2: v^s variables have the form $(1,1,...,1,v,v,...,v,0,0,...,0)$. We let $\widehat{s}$ denote the largest s such that $v^s = 1$ and $\widehat{t}$ denote the largest s such that $v^s = v$. We know that either $v = \frac{w^{\widehat{t}}}{\Delta^{\widehat{t}}}$ or $v = \frac{w^{\widehat{t}-1}}{\Delta^{\widehat{t}-1}}$, and either $v^{\widehat{s}} = 1 = \frac{w^{\widehat{s}}}{\Delta^{\widehat{s}}}$ or $v^{\widehat{s}} = 1 = \frac{w^{\widehat{s}-1}}{\Delta^{\widehat{s}-1}}$. To define the solution to the multiple choice model, let $y^{\widehat{t}} = v$, $y^{\widehat{s}} = 1 - v$, and $y^s = 0$, $\forall\ s \neq \widehat{s}, \widehat{t}$. If $v^{\widehat{s}} = \frac{w^{\widehat{s}}}{\Delta^{\widehat{s}}}$, then let $x^{\widehat{s}} = b^{\widehat{s}}(1-v)$. If $v^{\widehat{s}} = \frac{w^{\widehat{s}-1}}{\Delta^{\widehat{s}-1}}$, then let $x^{\widehat{s}} = b^{\widehat{s}-1}(1-v)$. If $v^{\widehat{t}} = \frac{w^{\widehat{t}}}{\Delta^{\widehat{t}}}$, then let $x^{\widehat{t}} = b^{\widehat{t}}v$. If $v^{\widehat{t}} = \frac{w^{\widehat{t}-1}}{\Delta^{\widehat{t}-1}}$, then let $x^{\widehat{t}} = b^{\widehat{t}-1}v$. We will first check for feasibility of this solution.

1. If $v^{\widehat{s}} = \frac{w^{\widehat{s}}}{\Delta^{\widehat{s}}}$ and $v^{\widehat{t}} = \frac{w^{\widehat{t}}}{\Delta^{\widehat{t}}}$, then $\sum_s x^s = x^{\widehat{s}} + x^{\widehat{t}} = b^{\widehat{s}}(1-v) + b^{\widehat{t}}v = b^{\widehat{s}} + v(b^{\widehat{t}} - b^{\widehat{s}}) = \sum_{s \leq \widehat{s}} \Delta^s + \sum_{\widehat{s} < s \leq \widehat{t}} v\Delta^s = \sum_s w^s = F.$

   If $v^{\widehat{s}} = \frac{w^{\widehat{s}}}{\Delta^{\widehat{s}}}$ and $v^{\widehat{t}} = \frac{w^{\widehat{t}-1}}{\Delta^{\widehat{t}-1}}$, then $\sum_s x^s = x^{\widehat{s}} + x^{\widehat{t}} = b^{\widehat{s}}(1-v) + b^{\widehat{t}-1}v = b^{\widehat{s}} + v(b^{\widehat{t}-1} - b^{\widehat{s}}) = \sum_{s \leq \widehat{s}} \Delta^s + \sum_{\widehat{s} < s < \widehat{t}} v\Delta^s = \sum_s w^s = F.$

   If $v^{\widehat{s}} = \frac{w^{\widehat{s}-1}}{\Delta^{\widehat{s}-1}}$ and $v^{\widehat{t}} = \frac{w^{\widehat{t}}}{\Delta^{\widehat{t}}}$, then $\sum_s x^s = x^{\widehat{s}} + x^{\widehat{t}} = b^{\widehat{s}-1}(1-v) + b^{\widehat{t}}v = b^{\widehat{s}-1} + v(b^{\widehat{t}} - b^{\widehat{s}-1}) = \sum_{s < \widehat{s}} \Delta^s + \sum_{\widehat{s} \leq s \leq \widehat{t}} v\Delta^s = \sum_s w^s = F.$

   If $v^{\widehat{s}} = \frac{w^{\widehat{s}-1}}{\Delta^{\widehat{s}-1}}$ and $v^{\widehat{t}} = \frac{w^{\widehat{t}-1}}{\Delta^{\widehat{t}-1}}$, then $\sum_s x^s = x^{\widehat{s}} + x^{\widehat{t}} = b^{\widehat{s}-1}(1-v) + b^{\widehat{t}-1}v = b^{\widehat{s}-1} + v(b^{\widehat{t}-1} - b^{\widehat{s}-1}) = \sum_{s < \widehat{s}} \Delta^s + \sum_{\widehat{s} \leq s < \widehat{t}} v\Delta^s = \sum_s w^s = F.$

   Therefore, this first constraint is satisfied.

2. If $v^{\widehat{s}} = \frac{w^{\widehat{s}}}{\Delta^{\widehat{s}}}$, then $x^{\widehat{s}} = b^{\widehat{s}}(1-v) = b^{\widehat{s}}y^{\widehat{s}} \geq b^{\widehat{s}-1}y^{\widehat{s}}$. If $v^{\widehat{s}} = \frac{w^{\widehat{s}-1}}{\Delta^{\widehat{s}-1}}$, then $x^{\widehat{s}} = b^{\widehat{s}-1}(1-v) = b^{\widehat{s}-1}y^{\widehat{s}} \leq b^{\widehat{s}}y^{\widehat{s}}$.

   If $v^{\widehat{t}} = \frac{w^{\widehat{t}}}{\Delta^{\widehat{t}}}$, then $x^{\widehat{t}} = b^{\widehat{t}}v = b^{\widehat{t}}y^{\widehat{t}} \geq b^{\widehat{t}-1}y^{\widehat{t}}$. If $v^{\widehat{t}} = \frac{w^{\widehat{t}-1}}{\Delta^{\widehat{t}-1}}$, then $x^{\widehat{t}} = b^{\widehat{t}-1}v = b^{\widehat{t}-1}y^{\widehat{t}} \leq b^{\widehat{t}}y^{\widehat{t}}$.

   $\forall\ s \neq \widehat{s}, \widehat{t}$, $x^s = 0$ and $y^s = 0$, therefore $b^{s-1}y^s \leq x^s \leq b^s y^s$.

   Therefore, this second constraint is satisfied.

3. $\sum_s y^s = y^{\widehat{s}} + y^{\widehat{t}} = (1-v) + v = 1$, therefore $\sum_s y^s \leq 1$.

4. $v \geq 0$, $1 - v \geq 0$, and $b^s \geq 0$, therefore, $x^s \geq 0$, and since $0 \leq v \leq 1$, $0 \leq y^s \leq 1$.

We have now shown (x^s, y^s) to be feasible. To show it has the same objective cost as (w^s, v^s), we again notice that these two solutions have the same structure as they did in Section A.2.1 . Therefore, they will again have the same cost.

We now have a valid mapping for extreme points of the two polyhedrons corresponding to the linear relaxations of the incremental and multiple choice formulations. We can represent

a feasible solution to one formulation as a convex combination of its endpoints α^j with weights θ^j summing to one; that is $\alpha = \sum_j \theta^j \alpha^j$. We have shown that we can transform each extreme point α^j into a feasible point β^j of the second formulation with the same cost. Then, $\beta = \sum_j \theta^j \beta^j$ is a feasible solution to the second formulation with the same cost as α. Therefore, we can conclude that we can map any feasible solution to one of the two formulations into a feasible solution to the other with the same objective cost. ■