APPROACHES TO NONLINEAR AND INFINITE-DIMENSIONAL NETWORK DESIGN PROBLEMS IN SUPPLY CHAIN OPTIMIZATION

By

THOMAS C. SHARKEY

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

2008

© 2008 Thomas C. Sharkey

To Melissa.

ACKNOWLEDGMENTS

Dr. H. Edwin Romeijn, my advisor, has been a tremendous mentor to me during my graduate career. I would like to thank him for all the time, effort, and patience he has put into working with me. He is a great role model for a career in academia and I hope to emulate his success in the future. I would also like to thank my committee, Dr. Ravindra Ahuja, Dr. Joseph Geunes, Dr. William Hager, Dr. Panos Pardalos, and Dr. Cole Smith, for their thoughtful and constructive feedback throughout my graduate studies. I especially thank Dr. Geunes and Dr. Smith for their help and guidance in my job search. I would also like to acknowledge that this work has been supported by a National Science Foundation Graduate Research Fellowship.

My family and friends have played an important role in my success up until this point. I would like to especially thank my mother, Andrea, my brother, Stephen, and my sister, Annemarie. Whether we were under the same roof or a thousand miles apart, their unconditional love and support has been a huge motivation for me. For all of my other friends and family, too numerous to list here, that have shown me nothing but support through the years, please know that I am forever grateful.

Finally, and most importantly, my beautiful wife Melissa has been amazing throughout the entire graduate school process. Despite her forcing our poor dog Tucker to listen to me practice my talks and presentations, she was always there to support and encourage me in every way possible. I hope to one day find a way to pay her back.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

APPROACHES TO NONLINEAR AND INFINITE-DIMENSIONAL NETWORK DESIGN PROBLEMS IN SUPPLY CHAIN OPTIMIZATION

By

Thomas C. Sharkey

August 2008

Chair: H. Edwin Romeijn
Major: Industrial and Systems Engineering

Network design and network flow problems constitute an extremely important class of optimization problems in supply chain management. The design of the infrastructure of a company, the location of the facilities or warehouses of a company, and the management of the inventory of a facility are all examples of network design and network flow problems in supply chain management. Due to the complexity of the real world, many of these network design and network flow problems make assumptions on the problem in order for the resulting optimization model to be effectively studied. However, these assumptions limit the applicability of the model due to the assumptions placed on the real world problem. In this dissertation, we study several new classes of network design problems that more explicitly capture real world supply chain optimization problems. These classes of problems often lead to nonlinear or infinite-dimensional extensions of classic optimization problems. However, unlike most nonlinear or infinite-dimensional optimization problems, the supply chain foundations of these problems allow for the development of effective approximate, heuristic, and exact methods to solve them.

In this dissertation, we examine three new classes of network design and network flow problems that have previously received little attention in the literature. The first problem that we examine is concerned with assigning a set of customers to a set of facilities where the facilities are then responsible for producing the amount of demand assigned to the facility. This leads to a new class of nonlinear generalized assignment problems. In many

practical situations, the resulting nonlinear functions in these problems are ill-structured (e.g., non-differentiable or discontinuous) so that standard optimization techniques are no longer applicable. We overcome this challenge through novel applications of the theory of linear programming. The second problem that we consider is a problem where the customers have dynamic demand over a finite horizon. We must assign each customer to an open facility and manage production and inventory levels at each open facility to meet the demand of the customers assigned to the facility. We develop approximation algorithms and results for this class of problems as well as setting the theoretical foundation for an algorithm to solve these problems to optimality. The third problem that we consider is a class of minimum-cost flow problems in infinite networks. This class of problems encompasses many supply chain planning problems where a sequence of decisions needs to be made over an infinite horizon. We are able to overcome the typical mathematical difficulties associated with linear programming in infinite-dimensional spaces and extend the well-known network simplex method to solve this class of problems in infinite networks. We do so in a nonstandard but intuitively appealing way that, to the greatest extent possible, employs concepts from finite-dimensional linear programming.

CHAPTER 1
INTRODUCTION

Network design problems are an extremely important class of optimization problems that arise in supply chain management. The design of the infrastructure of a company, the location of the facilities and warehouses of a company, and the assignment of customers to facilities or warehouses are all examples of network design problems in supply chain management. Network flow problems also play an important role in supply chain management. Many production planning problems can be formulated as minimum cost flow problems in appropriately defined networks. Moreover, sequential decision-making problems, in which a sequence of decisions has to be made over time, can be formulated as network flow problems. Unfortunately, due to the complexity of the real world, many of these network design problems and network flow problems make assumptions upon the situation that result in the optimization model losing some of its accuracy in capturing the real-world problem. In this dissertation, we study several new classes of network design problems that more explicitly capture a real-world supply chain optimization problem. These classes of network design problems lead to extensions of several classic optimization problems including the facility location problem, the generalized assignment problem, the knapsack (or resource allocation) problem, and the economic lot-sizing problem. These extensions often lead to global optimization problems. Unlike most global optimization problems, the special structure that arise from the supply chain foundations of these problems allow for the development of effective exact, heuristic, and approximate methods to solve them.

One of the main focuses of this dissertation is on supply chain optimization problems where we must assign a set of customers to a set of facilities (such as plants, warehouses, or distribution centers). The facility is then responsible for serving the set of customers assigned to it. The objective is to minimize the sum of assignment costs of customers to facilities plus the facility costs associated with serving the set of customers assigned to

the facility. Formally, we will index the set of customers by $j = 1, \ldots, n$ and the set of facilities by $i = 1, \ldots, m$. We define the binary variable x_{ij} to represent the decision of assigning customer j to facility i, i.e., $x_{ij} = 1$ implies that we assign customer j to facility i. It will be convenient to define the vector $x_{i\cdot} \in \{0,1\}^n$ to represent the set of customers assigned to facility i. Further, we define c_{ij} as the cost of assigning customer j to facility i and the function $H_i(x_{i\cdot})$ as the cost of serving the set of customers represented by $x_{i\cdot}$ at facility i. The supply chain network design (SCND) problem can be formulated as

$$\text{minimize} \sum_{i=1}^{m} \left(\sum_{j=1}^{n} c_{ij} x_{ij} + H_i(x_{i\cdot}) \right)$$

subject to

$$\begin{aligned} \sum_{i=1}^{m} x_{ij} &= 1 \qquad \text{for } j = 1, \ldots, n \\ x_{ij} &\in \{0,1\} \quad \text{for } i = 1, \ldots, m, \;\; j = 1, \ldots, n. \end{aligned} \tag{1–1}$$

Constraints (1–1) ensure that we assign each customer to a facility. We note that the facility cost function, $H_i(x_{i\cdot})$, in the SCND problem may require determining the optimal cost to an optimization problem based on the set of customers assigned to the facility. The facility cost function may also be infinite for certain $x_{i\cdot}$ to represent that facility i cannot feasibly serve this set of customers.

Given an optimization problem that belongs to the SCND framework, there are two main research approaches that can be used to study the problem. First, one can examine heuristic and/or approximate methods to return good solutions to the problem quickly. Typically, these methods exploit the structure of the SCND problem (in particular, the structure of $H_i(x_{i\cdot})$). Alternatively, we can develop exact approaches to the SCND problem which, although they may require more effort than the heuristic and approximate methods, determine the optimal solution to the problem. Branch and price algorithms have been extremely effective to solve a large number of problems that belong to the SCND framework. Interestingly enough, the pricing problem that arises in these branch

and price algorithms is an interesting supply chain planning problem in its own right. In particular, the pricing problem can be formulated as

$$\max_{x_{i\cdot} \in \{0,1\}^n} \sum_{j=1}^{n} r_{ij} x_{ij} - H_i(x_{i\cdot}),$$

where r_{ij} can be thought of as the revenue received if customer j is selected by facility i (i.e., $x_{ij} = 1$). Since $H_i(x_{i\cdot})$ is the cost of serving the set of selected customers, the pricing problem is interested in maximizing the profit of the facility, i.e., revenues received minus the cost of serving the selected customers. This problem belongs to the class of supply chain planning problems with customer (or market) selection. This class of problems allows the supplier to shape (to a certain extent) the demand that it will serve through the customer selection decisions. This is important since in some situations the supplier does have control over the set of its customers.

In the SCND problem, the function $H_i(x_{i\cdot})$ represents the cost of serving the set of customers represented by $x_{i\cdot}$ at facility i. In many situations, the facility will need to make decisions over time in order to serve the set of customers assigned to it. The facility may need to determine production/inventory levels in order to serve the set of customers at minimum cost, determine when equipment and machines need to be replaced in order to continue to serve the customers, or when, if at all, to expand the current capacity of the facility. These types of sequential decision making and/or planning problems can naturally be formulated as minimum cost network flow problems in appropriately defined networks. Typically in these formulations, there will be a subgraph of nodes and arcs in the network flow problem corresponding to each time period that a decision is made. There then will be arcs between the subgraphs of different time periods, which model the effects of how a decision in one time period affects the other time periods. If the horizon of the planning problem is finite, the resulting network flow formulation of the problem contains a finite number of nodes and arcs. However, there are many situations when either the horizon of the problem is unknown or it is not appropriate for the horizon to be finite. In both

of these cases, the resulting network flow formulation of the planning problem contains an infinite number of nodes and arcs. Therefore, many supply chain planning problems over an infinite horizon can be formulated as an important class of infinite-dimensional network-flow problems.

In this dissertation, we motivate, formulate, and study two new SCND problems; in particular, a nonlinear generalized assignment problem and an integrated facility location and production planning problem. The nonlinear generalized assignment problem belongs to the SCND framework where the facility cost function $H_{i\cdot}(x_{i\cdot})$ is equal to a nonlinear function whose argument is a linear function of the assignments made to the facility, as long as the assignments satisfy a resource capacity constraint. The pricing problem for this class of problems is a nonlinear knapsack problem whose objective function is that of a linear function minus a nonlinear function whose argument is a linear function of the variables in the problem. In the supply chain context, the linear function of the assignments (or variables) can be interpreted as the amount of demand assigned to the facility (or selected in the knapsack problem) and the nonlinear function is the cost of producing a certain amount of demand at the facility. In supply chain management, there are a vast number of different types of production cost functions, many of which are ill-structured. Therefore, in studying these classes of problems, we wish to develop methods that require as few structural assumptions as possible on the nonlinear cost functions. For an important class of nonlinear continuous optimization problems (that includes the relaxation of the above nonlinear generalized assignment problem and the above nonlinear knapsack problem), we will develop an important structural property of some optimal solution to the problem using theory from linear programming. This property requires no assumptions on the structure of the nonlinear function other than the existence of an optimal solution to the optimization problem. This result will lead to the development of a novel set of dual multipliers associated with the optimal solution to the problem. These dual multipliers will play an important role in developing new methods

and algorithms to solve the nonlinear generalized assignment problem and the nonlinear knapsack problem that require very few (if any, in some cases) structural assumptions on the nonlinearity of the problem.

The integrated facility location and production planning problem studied in this dissertation belongs to the SCND framework where the facility cost function, $H_{i\cdot}(x_{i\cdot})$, for some non-empty subset of customers is equal to a facility opening cost plus the optimal cost of meeting the cumulative demand in each time period of the customers assigned to the facility. In other words, we must determine the optimal production/inventory levels at the facility in each time period to ensure timely delivery of all the demand assigned to the facility over the horizon. A traditional area of research for variations of facility location problems is to examine approximation algorithms for them. We will show that, in general, it is unlikely that a constant factor approximation algorithm exists for our integrated facility location and production planning problems. However, we will develop constant factor approximation algorithms for certain special classes of these problems. In fact, some of these special classes of integrated facility location and production planning problems lead to new classes of facility location problems, for which we develop constant factor approximation algorithms. The pricing problem that arises from this class of problems is an integrated production planning and customer selection problem where we must select a set of customers and meet the demand of all the selected customers in each time period in order to maximize the profits of the supplier. Although this problem is NP-hard in general, we will show that several practically relevant subclasses of the problem can be solved efficiently.

In this dissertation, we also study certain classes of planning problems over an infinite horizon. In particular, we study minimum-cost network-flow problems in networks with a countably infinite number of nodes and arcs and integral flow data. This problem class contains many nonstationary planning problems over time where no natural finite planning horizon exists. We will extend the well-known finite-dimensional network simplex method

to the infinite-dimensional case. We do so in a nonstandard, but intuitively appealing, way that, to the greatest extent possible, employs concepts from finite-dimensional linear programming. We use an intuitive natural dual problem and show that weak and strong duality hold. Using recent results regarding the structure of basic solutions to infinite-dimensional network-flow problems we are able to extend the well-known finite-dimensional network simplex method to the infinite-dimensional case. In addition, we study a class of infinite network-flow problems whose flow balance constraints are inequalities (that include production planning problems) and show that the simplex method can be implemented in such a way that each pivot takes only a finite amount of time.

The remainder of this dissertation is organized as follows. We review the relevant literature associated with this dissertation in Chapter 2. In Chapters 3-6, we motivate, formulate, and study a new class of nonlinear generalized assignment problems. In Chapter 3, we introduce the class of problems, the pricing problem associated with them, and discuss the applications of these two classes of problems. In Chapter 4, we discuss a greedy procedure for the class of nonlinear generalized assignment problems based on the optimal solution to its continuous relaxation. In Chapter 5, we discuss solution methods to solve a class of convex programming problems, which includes the continuous relaxation of the nonlinear generalized assignment problem with convex functions. In Chapter 6, we discuss the relaxation of the pricing problem for this class of nonlinear generalized assignment problems, i.e., a continuous nonlinear knapsack problem. We develop algorithms to solve the nonlinear knapsack problem and variants of it that require virtually no assumptions on the structural properties of the nonlinear function. In Chapters 7-9, we motivate, formulate, and study the integrated facility location and production planning problem. In Chapter 7, we introduce the class of problems and the integrated production planning and customer selection problem that arises as the pricing problem. In Chapter 8, we examine constant factor approximation algorithms for the class

of problems. In Chapter 9, we discuss issues around developing exact algorithms for the integrated facility location and production planning problems. The integrated production planning and customer selection problem is discussed in this chapter as well. In Chapter 10, we discuss infinite-dimensional network-flow problems. We extend the network simplex method to infinite-dimensional spaces and discuss an important class of problems where the infinite-dimensional network simplex method can be implemented in such a way that each pivot requires a finite amount of time and information. We conclude the dissertation in Chapter 11 with a summary and concluding remarks.

CHAPTER 2
LITERATURE REVIEW

In this chapter, we will review the relevant literature for the problems and solution methods studied in this dissertation. We will begin by discussing two fundamental Operations Research problems in supply chain management that can be formulated within the SCND framework. In particular, the generalized assignment problem (GAP) and the facility location problem are special cases of the SCND problem. In this dissertation, we will study nonlinear extensions of both of these problems. Therefore, we discuss the streams of literature for the GAP related to approaches in this dissertation in Section 2.1 and relevant streams of literature for the facility location problem in Section 2.2. We discuss extensions of these problems and other customer assignment problems in Section 2.3. As discussed in Chapter 1, supply chain planning problems with customer selection are important in their own right but also arise as subproblems in solving problems in the SCND framework. In Sections 2.4 and 2.5, we discuss models and algorithms that are related to these planning problems with customer selection. We conclude this chapter in Section 2.6, which discusses network-flow formulations of planning problems and approaches to solve infinite-dimensional optimization problems.

2.1 The Generalized Assignment Problem

The generalized assignment problem (GAP) seeks an assignment of customers to facilities (or, as it is sometimes defined, jobs to machines) that minimizes the sum of the assignment costs while respecting the capacity of each facility. If we define a_{ij} as the capacity requirement of assigning customer j to facility i and b_i as the capacity of facility i, then the GAP can be formulated as a SCND problem where by defining the facility cost function as

$$H_i(x_{i\cdot}) = \begin{cases} 0 & \text{if } \sum_{j=1}^{n} a_{ij}x_{ij} \leq b_i \\ \infty & \text{if } \sum_{j=1}^{n} a_{ij}x_{ij} > b_i. \end{cases}$$

Alternatively, it is much more suitable to formulate the GAP with the capacity requirements in the constraint system of the problem. In other words, the GAP is typically formulated as

$$\text{minimize} \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}$$

subject to

$$\begin{aligned} \sum_{j=1}^{n} a_{ij} x_{ij} &\leq b_i && \text{for } i = 1, \ldots, m \\ \sum_{i=1}^{m} x_{ij} &= 1 && \text{for } j = 1, \ldots, n \\ x_{ij} &\in \{0, 1\} && \text{for } i = 1, \ldots, m, \quad j = 1, \ldots, n. \end{aligned}$$

The GAP was introduced by Ross and Soland [132]. It has many applications including: assigning jobs to computer networks (Balachandran [11]), fixed-charge plant location (Ross and Soland [133]), routing problems (Fisher and Jaikumar [52]), and the single-sourcing problem (De Maio and Roveda [40]). There are many existing solution methods (both exact and heuristic) for the GAP (see Cattrysse and van Wassenhove [31] and Osman [109]).

The GAP is NP-hard (for a definition of NP-hardness, see Garey and Johnson [62]); in fact, it is NP-hard to even determine if the GAP has a feasible assignment of customers to facilities (see Martello and Toth [101]). Therefore, it is appropriate to develop *heuristics* for the GAP, i.e., algorithms that return a 'good' solution to the GAP quickly. The class of greedy heuristics introduced by Romeijn and Romero Morales [124] for the GAP are related to a solution procedure for a nonlinear version of the GAP studied in Chapter 4 of this dissertation. These heuristics define a weight, or pseudo-cost, function, $f(i, j)$ (see Martello and Toth [100]), for each facility/customer pair. This function represents a measure of the cost of the assignment of customer j to facility i. We then define the

desirability of assigning customer j to its best facility as

$$\rho_j = \max_i \min_{k \neq i} (f(k,j) - f(i,j)).$$

Essentially, ρ_j measures the additional cost incurred by not assigning customer j to its most desirable facility. The greedy heuristic then assigns customers in decreasing order of their desirability. The weight function proposed by Romeijn and Romero Morales [124] is $f_\lambda(i,j) = c_{ij} + \lambda_i a_{ij}$. This heuristic possesses several desirable properties when λ_i is set equal to the optimal dual multiplier of the capacity constraint of facility i in the linear programming relaxation of the GAP. In particular, the heuristic can be shown to be both asymptotically feasible and asymptotically optimal under a very general stochastic model of the GAP.

Probabilistic analysis of combinatorial optimization problems and heuristics for them have quite a presence in the literature. The median location problem (Rhee and Talagrand [119], the multi-knapsack problem (Van de Greer and Stougie [63] and Meanti et al. [102]), the set-covering problem (Piersma [114]), and the parallel machine scheduling problem (Piersma and Romeijn [115]) have all been studied under probabilistic models. The main idea of all of these works is to link the problem to an empirical process (see Alexander [5] and Talagrand [150]) to show that the cost of the optimal solution grows linearly as a function of the size of the problem. However, since the GAP may not have a feasible solution, complications arise in any probabilistic analysis for it. Dyer and Frieze [46] examine a stochastic model of the GAP under the assumption that all instances are feasible with probability 1. Romeijn and Piersma [123] provide a stochastic model of the GAP and derive a tight condition under which the resulting GAP is asymptotically feasible with probability 1 and then examine the optimal solution value of the GAP under the stochastic model. Further, Romeijn and Romero Morales [125] generalize these results to the so-called multi-period single-sourcing problem, which can be formulated as a nonlinear version of the GAP. The probabilistic analysis of heuristics for combinatorial

optimization problems are also important since this provides theoretical guarantees for the heuristics. Rinnooy Kan et al. [89] analyze a heuristic for the multi-knapsack problem under a stochastic model. Romeijn and Romero Morales [124] analyze a heuristic for the GAP under a stochastic model. Romeijn and Romero Morales [126, 127] analyze heuristics for the multi-period single-sourcing problem with cyclic and acyclic demands, respectively, under stochastic models. Ahuja et al. [3] analyze a heuristic for the multi-period single-sourcing problem with perishable inventories. Rainwater et al. [118] examine a heuristic for the GAP with flexible jobs under a stochastic model.

It is also of interest to be able to solve the GAP to optimality with an exact algorithm. Since the GAP is an integer program, standard integer programming techniques (see Nemhauser and Wolsey [107]) can be applied to solve it. However, many exact algorithms that employ the structure of the problem have been developed to solve the GAP. The branch and price algorithm of Savelsbergh [136] is an important approach that will be utilized in this dissertation. This algorithm solves the set-partitioning formulation of the GAP using column generation and a branch and bound scheme. In particular, any assignment of customers to facilities in the GAP can be thought of as a partition of the set of customers into m sets and then the assignment of each of these sets to a facility. We will next describe the set-partitioning formulation of the GAP in detail.

Denote the number of distinct subsets that can be feasibly assigned to facility i by L_i. Let the binary vector $\alpha_{i\cdot}^{\ell}$ represent the ℓ^{th} subset associated with facility i, where $\alpha_{ij}^{\ell} = 1$ if customer j belongs to the subset and 0 otherwise. (We also refer to $\alpha_{i\cdot}^{\ell}$ as the ℓ^{th} *column* associated with facility i.) Furthermore, let $h_i(\alpha_{i\cdot}^{\ell}) = \sum_{j=1}^{n} c_{ij}\alpha_{ij}^{\ell}$ denote the cost associated with serving the set of customers given by $\alpha_{i\cdot}^{\ell}$ at facility i. If we then define the decision variable y_i^{ℓ} to be equal to one if facility i serves the ℓ^{th} associated subset and 0 otherwise, the set-partitioning formulation (SPF) of the GAP is

$$\text{minimize} \sum_{i=1}^{m} \sum_{\ell=1}^{L_i} h_i(\alpha_{i\cdot}^{\ell}) y_i^{\ell}$$

subject to

$$\sum_{i=1}^{m} \sum_{\ell=1}^{L_i} \alpha_{ij}^{\ell} y_i^{\ell} = 1 \qquad \text{for } j = 1, \ldots, n \tag{2–1}$$

$$\sum_{\ell=1}^{L_i} y_i^{\ell} = 1 \qquad \text{for } i = 1, \ldots, m \tag{2–2}$$

$$y_i^{\ell} \in \{0, 1\} \qquad \text{for } i = 1, \ldots, m,\ \ell = 1, \ldots, L_i.$$

Constraints (2–1) ensure that customer j is assigned to a facility and constraints (2–2) ensure that we assign exactly one subset of the customers to each facility. It is not difficult to extend this formulation to that of a general SCND problem; we simply define $h_i(\alpha_{i\cdot}^{\ell}) = \sum_{j=1}^{n} c_{ij}\alpha_{ij}^{\ell} + H_i(\alpha_{i\cdot}^{\ell})$.

Unfortunately, the SPF has an exponential number of variables. To address this problem, we use the column generation approach of Gilmore and Gomory [71] to solve the relaxation of SPF (SPFR). This procedure will be used to calculate the LP bounds of SPFR at each node of a branch-and-bound tree, leading to a branch and price algorithm to solve SPF. One of the most vital parts to the success of a branch-and-price algorithm is the ability to quickly solve the pricing problem. Suppose that N is the current set of columns in the reduced LP relaxation of SPFR. Let $(\mu^*(N), \delta^*(N))$ be the optimal dual multipliers associated with the reduced LP relaxation of SPFR. The columns will be searched for each facility $i = 1, \ldots, m$. In particular, we need to solve the following pricing problem for facility i,

$$\min_{x_{i\cdot} \in \{0,1\}^n} \sum_{j=1}^{n} c_{ij}x_{ij} + H_i(x_{i\cdot}) - \sum_{j=1}^{n} \mu_j^*(N)x_{ij} + \delta_i^*(N).$$

By setting $r_{ij} = \mu_j^*(N) - c_{ij}$ and disregarding $\delta_i^*(N)$ (it is a constant), we can formulate this pricing problem as

$$\max_{x_{i\cdot} \in \{0,1\}^n} \sum_{j=1}^{n} r_{ij} x_{ij} - H_i(x_{i\cdot}).$$

If we interpret r_{ij} as the revenue received by facility i for selecting customer j, then this pricing problem can be viewed as maximizing the revenues received from selected customers minus the cost of serving the selected customers, i.e., maximize the profits of facility i. Therefore, we view the pricing problem as a supply chain planning problem with customer (or market) selection. This class of problems allows the supplier to help shape the demand that it must supply in order to maximize profits. This class of problems belongs to a larger class of demand management models in supply chain optimization, which we discuss in Section 2.4. First, we will revisit the pricing problem associated with the GAP. Recall that $H_i(x_{i\cdot}) = 0$ if the selected customers in $x_{i\cdot}$ respect the capacity constraint of facility i and it is infinite otherwise. Therefore, the pricing problem that arises from the SPF of the GAP can be formulated as

$$\max_{a_{i\cdot}^\top x_{i\cdot} \leq b_i,\, x_{i\cdot} \in \{0,1\}^n} \sum_{j=1}^{n} r_{ij} x_{ij},$$

which is simply the traditional knapsack problem. We will consider a nonlinear extension of the GAP in Chapter 3. The resulting pricing problem is thus a nonlinear version of the knapsack problem, so we will review the literature on nonlinear knapsack problems in Section 2.5.

2.2 The Facility Location Problem

Traditional facility location problems focus on determining a set of open facilities and assigning each customer to an open facility, where the goal is to minimize the sum of facility opening costs and connection costs of the customers to the facilities. The uncapacitated facility location problem (UFLP) is one of the fundamental problems in location theory. It can be stated as follows: given a set of facilities and a set of customers, determine an assignment of customers to open facilities that minimize the facility opening

costs plus the assignment costs. If we let f_i be the opening cost of facility i and c_{ij} as the cost of assigning customer j to facility i, then the UFLP fits into the SCND framework by setting $H_i(x_{i\cdot}) = f_i$ if there exists some j such that $x_{ij} = 1$ and 0 otherwise. Although the UFLP can be formulated with an assignment cost c_{ij}, it traditionally is formulated with a demand level, d_j, of customer j and a *connection* cost $\bar{c}_{ij}$, where the assignment cost is then equal to $c_{ij} = d_j \bar{c}_{ij}$. Much like the GAP, the UFLP is typically formulated as an integer programming problem,

$$\text{minimize} \sum_{i=1}^{m} \left(f_i y_i + \sum_{j=1}^{n} d_j \bar{c}_{ij} x_{ij} \right)$$

subject to

$$\begin{aligned}
x_{ij} &\leq y_i && \text{for } i = 1, \ldots, m, \quad j = 1, \ldots, n \\
\sum_{i=1}^{m} x_{ij} &= 1 && \text{for } j = 1, \ldots, n \\
x_{ij} &\in \{0, 1\} && \text{for } i = 1, \ldots, m, \quad j = 1, \ldots, n \\
y_i &\in \{0, 1\} && \text{for } i = 1, \ldots, m,
\end{aligned}$$

where y_i represents the decision of opening facility i. Since even the UFLP is NP-hard, an important stream of work on facility location problems is the design of approximation algorithms (see Hochbaum [80]) for them. We say that an algorithm is a α-approximation algorithm for a minimization problem if the algorithm runs in polynomial time and returns a solution to the problem whose cost is no more than α times the cost of the optimal solution to the problem.

The field of designing constant factor approximation algorithms for facility location problems and variants has been extremely active since Shmoys et al. [143] gave the first constant factor guarantee for the metric UFLP. Their algorithm used the idea of rounding the optimal solution to a linear programming relaxation of the facility location problem. Sviridenko [148] used an LP-rounding technique to develop an approximation algorithm

with a factor of 1.58. Recently, Byrka [29] used LP-rounding, the algorithm of Chudak and Shmoys [37], and the algorithm of Mahdian et al. [98] to achieve an algorithm with a factor of 1.50. These algorithms have high running times since they must solve a linear program. Primal-dual algorithms for the UFLP are computationally attractive and can achieve similar approximation guarantees. Jain et al. [87] offer primal-dual algorithms with approximation guarantees of 1.861 and 1.61. The current best-known approximation factor for the metric UFLP using a primal-dual algorithm achieves a guarantee of 1.52 and is due to Mahdian et al. [98]. This algorithm uses the 1.61-approximation algorithm of Jain et al. [87] and the idea of scaling (see Charikar and Guha [33]).

In Chapter 8, we will discuss a generalization of the UFLP where the facility costs are general concave functions of the amount of demand assigned to the facility. We will refer to this problem as the *concave cost facility location problem* (CCFLP). Hajiaghayi et al. [75] generalize the 1.861-approximation algorithm of Jain et al. [87] for the UFLP to the special case of the CCFLP in which each customer has unit demand; this algorithm runs in $O(n^3 \log n)$ time. In addition, they show that the CCFLP with unit demands can be converted to a UFLP with n customers and nm facilities and then use the algorithm of Mahdian et al. [98] to obtain a 1.52-approximation algorithm that runs in $O(n^6)$ time. For the CCFLP with general integral demand, this approach has a pseudo-polynomial running time. In particular, the UFLP would have $O(nD)$ customers and $O(nmD)$ facilities where D is equal to the sum of the demands of the customers. Further, for the CCFLP with general integral demand, we can develop a $(1.52 + \epsilon)$-approximation algorithm by approximating the concave functions with piecewise linear functions and using a similar reduction as Hajiaghayi et al. [75] to the UFLP. However, this approach has a running time of $O(n^3(\frac{\ln D}{\ln(1+\epsilon)})^3)$, which is dependent on both D and on ϵ. In Chapter 8, we provide a generalization of the 1.61-approximation algorithm of Jain et al. [87] for the UFLP to the CCFLP with general integral (or, rational) demands that runs in $O(n^4 \log n)$ time. We then use this algorithm and a scaling idea to generalize the 1.52-approximation algorithm

of Mahdian et al. [98] to the CCFLP that also runs in $O(n^4 \log n)$ time. Independent of the work in this dissertation, Magnanti and Stratila [97] provided a 1.61-approximation algorithm for the CCFLP with general integral demands with a running time of $O(n^4)$. A summary of past approximation algorithms for facility location problems that are relevant to the work in Chapter 8 appears in Table 2-1.

Table 2-1. Relevant approximation algorithms for facility location problems.

Reference	Problem	Factor	Running Time
Jain et al. [87]	UFLP	1.861	$O(n^2 \log n)$
Jain et al. [87]	UFLP	1.61	$O(n^3)$
Mahdian et al. [98]	UFLP	1.52	Quasi-linear
Hajiaghayi et al. [75]	CCFLP with unit demands	1.861	$O(n^3 \log n)$
Hajiaghayi et al. [75]	CCFLP with unit demands	1.52	$O(n^6)$
Consequence of [75]	CCFLP with general demands	1.52	Pseudo-polynomial
Consequence of [75]	CCFLP with general demands	$1.52 + \epsilon$	$O(n^3(\frac{\ln D}{\ln(1+\epsilon)})^3)$
Chapter 8	CCFLP with general demands	1.52	$O(n^4 \log n)$

2.3 Customer Assignment Problems

There have been many other customer assignment problems belonging to the SCND framework that have appeared in the literature. Some of these customer assignment problems are extensions of the GAP, the facility location problem, or both. We note that the if the capacity requirements of the customers are facility-independent, i.e., $a_{ij} = a_j$ for $i = 1, \ldots, m$ and $j = 1, \ldots, n$, the GAP is often referred to as the *single-sourcing problem*.

The single-sourcing problem has been used to model the assignment of customers to facilities in problems where the demand of each customer is static, i.e., the demand is time-invariant. Examples of such models and problems can be found in Geoffrion and Graves [64], Benders et al. [17], and Fleischmann [54]. However, these models have limited applicability due to the assumption of static demand and the fact that they do not explicitly model inventory problems. Huang et al. [85] consider extensions of the single-sourcing problem where each customer has a constant demand rate and each facility pays costs in an associated Economic Order Quantity (EOQ) model with a demand rate equal to the cumulative demand rates of customers assigned to that facility. Duran [43],

Romein and Romero Morales [125], and Freling et al. [57] all consider single-sourcing problems where the demand of the customers is time-varying. Romeijn and Romero Morales [125] consider the multi-period single-sourcing problem (MPSSP) in which we must meet the demands of the set of customers over a discrete, finite horizon. In the MPSSP, the demand of each customer in each time period must be met by a single facility and production at each facility in each time period is capacitated. Freling et al. [57] consider a variant of the MPSSP with the additional restriction that each customer must be assigned to the same facility and the demand pattern of each customer share the same seasonality pattern, i.e., the demand in time period t of customer j can be decomposed into the product of a seasonal factor σ_t of time period t and the base demand level d_j of customer j.

Daskin et al. [39] and Shen et al. [141] consider a joint-inventory location model. In this problem, which is an extension of the UFLP, each customer has an uncertain demand. We are then interested in determining an assignment of the customers to open facilities, how often to produce (or reorder) at open facilities (or distribution centers), and what level of safety stock to maintain at each open facility in order to minimize all relevant costs while maintaining a specified service level throughout the system. Shen et al. [141] examine a branch and price algorithm to solve the set-partitioning formulation of this problem for two special cases. In particular, they discuss how to efficiently (both theoretically and computationally) solve the pricing problem when (i) the ratio of the variance and the mean of the demand of each customer is constant and (ii) the variance of the demand of each customer is zero. Shu et al. [144] show how to solve the general pricing problem effectively for the general pricing problem that arises in Shen et al. [141] and discuss its implications in the branch and price algorithm for the general joint-inventory location problem. Sourirajan et al. [147] consider an extension of this problem that accounts for lead time in producing the product at the facilities (or delivering the product to distribution centers).

The common theme in these customer assignment models in which the the demands are dynamic is that each facility must manage their production and inventory levels in order to meet the demand of the set of customers assigned to the facility. In Chapter 7, we discuss a customer assignment problem where each facility must manage the production and inventory decisions at the facility in order to ensure we meet the cumulative demand of the set of customers assigned to the facility in each time period. The production planning problem that is faced by a facility is a generalization of the classical economic lot-sizing problem (see Wagner and Whitin [157]) where concave production cost functions replace fixed-charge plus linear production costs and concave holding cost functions replace linear holding costs. This production planning problem can be solved in $O(T^2)$ time (see Wagner [156] and Veinott [154]). The economic lot-sizing problem can be solved in $O(T \log T)$ and can be solved in $O(T)$ time in the case of non-speculative motives (see Aggarwal and Park [1], Federgruen and Tzur [47], and Wagelmans et al. [155]). Krarup and Bilde [92] used a primal-dual algorithm to show that the facility location formulation of the economic lot-sizing problem yielded an integral solution. Levi et al. [95] developed primal-dual algorithms for three important classes of deterministic inventory problems, including the economic lot-sizing problem. Their primal-dual algorithm solved the economic lot-sizing problem to optimality. They also developed a 2-approximation algorithm for the joint-replenishment problem (see Zangwill [159], Veinott [154], and Arkin et al. [9]) and a 2-approximation algorithm for the multistage assembly problem (see Roundy [134]).

2.4 Demand Management Models

In most supply chain (or production) planning problems, the focus is on minimizing the operating costs of the supplier in order to meet the demand. However, in many practical contexts, the supplier has some control over the demand for its product. For example, the supplier may choose the cities and/or stores in which it will make its product available. Therefore, it is important to develop planning models that explicitly allow

the supplier to shape the demand (to a certain degree) for a product. Two important mechanisms that a supplier may use to shape its demand are pricing and customer (or market) selection.

Models and algorithms that integrate pricing decisions to shape demand and planning decisions to meet the resulting demand have been the focus of a recent stream of literature. This research builds upon the earlier works of Thomas [152, 153], Florian and Klein [55], and Kunreuther and Schrage [93, 94]. There have been several papers that consider pricing decisions in the context of inventory planning problems that have a structure related to the economic lot-sizing problem. These papers include Geunes et al. [66], van den Heuvel and Wagelmans [78], Deng and Yano [41], Ahn et al. [2], Geunes et al. [65] and Merzifonluoglu et al. [103]. These papers assume that demand is known for a given price level. Examples of inventory management and pricing decisions where demand is a random variable based upon the price can be found in Gallego and van Ryzin [60], Petruzzi and Dada [113], Chen and Simchi-Levi [35, 36], and Bakal et al. [10].

Supply chain planning problems with customer selection are not only important for their role in shaping the demand of the supplier but can also, as discussed in Section 2.1, arise as subproblems in algorithms (such as branch and price) to solve SCND problems. Several classic inventory control problems have been studied with customer selection decisions. Geunes et al. [67] consider customer selection in various economic order quantity settings. Taaffe et al. [149] consider integrated newsvendor and customer selection problems. Van den Heuvel et al. [77] discuss customer selection decisions in the context of the economic lot-sizing problem.

2.5 Nonlinear Knapsack Problems

In Section 2.1, we discussed that the pricing problem that arises in a branch and price algorithm for the GAP is the knapsack problem. In Chapter 3 of this dissertation, we consider a nonlinear version of the generalized assignment problem which leads to a pricing problem which is a nonlinear knapsack problem. There is extensive literature

regarding both linear and nonlinear knapsack problems. Martello and Toth [101] is an excellent resource for linear knapsack problems and Ibaraki and Katoh [86] offer a comprehensive treatment of the problem where we maximize a convex and separable function subject to a linear constraint. Bretthauer and Shetty [26] offer a thorough review of the literature on nonlinear knapsack problems.

Much of the previous work on the nonlinear extensions of the continuous knapsack problem have focused on separable versions of the problem. Moré and Vavasis [104] showed that determining the global solution to a concave, separable minimization problem with a knapsack constraint is NP-hard and then focused on obtaining locally optimal solutions. Brucker [28] developed a linear-time algorithm for the case of a separable, convex, quadratic objective function through a breakpoint search of the Lagrangian function. This idea has been used several other times, for example by Pardalos and Kovoor [111], Hochbaum and Hong [81], and Kiwiel [90]. Bitran and Hax [22] propose a so-called pegging algorithm for a separable, convex objective function subject to a linear knapsack constraint and bounds on the variables. Robinson et al. [120] develop a pegging algorithm for a separable, quadratic objective function subject to a linear knapsack constraint which relies on projecting the origin onto a relaxed problem. Bretthauer and Shetty [27] develop a pegging algorithm for a class of problems with a separable, convex objective function subject to a generalized knapsack constraint that bounds a separable, convex function of the decision variables.

The main focus of the literature when the objective function is nonseparable has been on the quadratic knapsack problem. Pang [110] and Pardalos et al. [112] consider a quadratic objective function subject to a linear constraint and bounds on the variables. Dussault et al. [44] and Klastorin [91] solve the problem through a series of separable subproblems. Caprara et al. [30] explore a Lagrangian relaxation in order to solve the quadratic knapsack problem exactly. Ceselli and Righini [32] study a class of nonseparable, nonlinear knapsack problems that they call penalized knapsack problems and that have

an objective function of the form $p^\top x - \max_{i=1,\ldots,n} s_i x_i$. Romeijn et al. [122] consider the problem of maximizing the difference of a linear function and a concave function whose argument is a linear function of the variables subject to a knapsack constraint.

Efficient algorithms for continuous knapsack problems along with an observation that an optimal solution exists to the problem with a small number of fractional components form an important step towards discrete knapsack problems using a branch-and-bound algorithm. Some approaches for discrete knapsack problems are algorithms developed by Horowitz and Sahni [83], Martello and Toth [99], and Nauss [106] for the traditional linear knapsack problem; the algorithms developed by Bretthauer and Shetty [25, 27] for a nonlinear integer knapsack problem in which both the objective and constraint function are convex and separable in the decision variables; and the algorithm developed by Gallo et al. [61] for solving a binary quadratic, nonseparable knapsack problem.

2.6 Infinite-Dimensional Network-Flow Problems

Standard static minimum-cost network-flow problems on finite networks have been very widely studied; Ahuja, Magnanti, and Orlin [4] offer an in-depth overview of this field. Moreover, dynamic minimum-cost network-flow problems, in which a time-delay is associated with flow on each arc in the network, have been studied as well. For example, Hoppe and Tardos [82] study such problems with a discrete time parameter, while Anderson et al. [7], Anderson and Philpott [8], and Fleischer and Tardos [53] deal with continuous-time dynamic network-flow problems. In contrast, very little work has been devoted to the solution of network problems on countably infinite networks.

In the area of infinite-horizon optimization, much attention has been devoted to the solution of planning problems over an infinite horizon via an infinite sequence of finite-dimensional truncations (see, e.g., Grinold [73], Bean and Smith [15], Bean et al. [14], Bès and Sethi [21], Schochetman and Smith [137, 138], Federgruen and Tzur [48–50]). One of the important goals is then to establish the existence of *planning* and *forecast horizons*, i.e., finite-dimensional truncations that have the desirable property that

the "first" decision (in the case of sequential decision making) is (close to) optimal. In Chapter 10, we will instead employ a so-called *strategy horizon* approach in which we attempt to directly solve the infinite horizon problem (see also Ghate and Smith [70]). This implicitly means that we approximate different decision sequences with different approximating horizons.

The simplex method for linear programming was generalized to classes of so-called separated continuous infinite-dimensional linear programming problems, which generalize the continuous-time network-flow problems, in Pullan [116] and Weiss [158]. Anderson and Nash [6] studied linear optimization problems in general infinite-dimensional spaces with the goal of generalizing the simplex method. Their results were limited due to the difficulty in characterizing extreme point solutions and basic solutions through a decomposition of the decision variables into sets of basic and nonbasic variables. However, for infinite network-flow problems, Romeijn et al. [128] recently developed extreme point characterizations which led to a characterization of basic solutions through basic and nonbasic variables for network-flow problems with integral data. Ghate and Smith [68] extended these results to a class of doubly infinite linear programming problems.

Another complication is the establishment of weak and strong duality, which is obviously of great importance to the development of a simplex method and the economic interpretation of optimal dual solutions. By posing the optimization problem in appropriate infinite-dimensional spaces, weak duality can usually be obtained in a relatively straightforward manner. However, these spaces then often prove too restrictive to also obtain general strong duality results (see, e.g., Luenberger [96] and Anderson and Nash [6]). Duality issues in semi-infinite linear programs were studied by Charnes et al. [34], Duffin et al. [42], and Borwein [24]. Grinold [72] established conditions for the existence of optimal dual solutions to a special class of doubly-infinite linear programs. Jones et al. [88] applied the theory developed by Grinold and Hopkins [74] to an infinite-horizon equipment replacement problem. Pullan [117] derives duality results

for a class of so-called separated continuous linear programs. Clark [38] studies a general class of infinite-dimensional linear programming problems and is able to obtain strong duality results under a set of assumptions on the problem related to degeneracy; these assumptions, however, are quite restrictive in an infinite-dimensional setting, and in particular in minimum-cost network-flow problems in infinite networks. Finally, Ghate and Smith [69] develop a duality theory for countably infinite linear programs.

CHAPTER 3
A CLASS OF NONLINEAR GENERALIZED ASSIGNMENT PROBLEMS

In this chapter and Chapters 4-6, we will examine problems related to solving the following class of nonlinear generalized assignment problems,

$$\text{minimize} \sum_{i=1}^{m} \left(\sum_{j=1}^{n} c_{ij} x_{ij} + g_i \left(\sum_{j=1}^{n} s_{ij} x_{ij} \right) \right)$$

subject to (NL-GAP)

$$\sum_{j=1}^{n} a_{ij} x_{ij} \leq b_i \qquad \text{for } i = 1, \ldots, m \tag{3–1}$$

$$\sum_{i=1}^{m} x_{ij} = 1 \qquad \text{for } j = 1, \ldots, n \tag{3–2}$$

$$x_{ij} \in \{0, 1\} \qquad \text{for } i = 1, \ldots, m, \;\; j = 1, \ldots, n. \tag{3–3}$$

In the NL-GAP, we are interested in assigning the set of customers to the set of facilities while respecting the capacity constraints of each facility. Constraints (3–1) represent the capacity of each facility. Note that the NL-GAP fits into the SCND framework by defining the function $H_i(x_{i\cdot})$ as follows:

$$H_i(x_{i\cdot}) = \begin{cases} g_i \left(\sum_{j=1}^{n} s_{ij} x_{ij} \right) & \text{if } \sum_{j=1}^{n} a_{ij} x_{ij} \leq b_i \\ \infty & \text{if } \sum_{j=1}^{n} a_{ij} x_{ij} > b_i. \end{cases}$$

The motivation for examining this particular nonlinear extension of the GAP is due to its many applications in supply chain management and parallel machine scheduling. In the context of supply chain management, consider the following general framework to motivate the NL-GAP. Each customer will have a certain demand level (or rate) for a product. If customer j is assigned to facility i, the demand level of customer j for the product is equal to s_{ij}. Further, in meeting the demand of customer j at facility i, it is necessary to use some level, a_{ij}, of a scarce resource (such as storage space, labor hours, or materials). Facility i only has a certain level, b_i, of the resource. The function, $g_i(S)$, represents the cost of producing S units of the product at facility i. In this situation, we

seek an assignment of customers to facilities that minimizes the total of the assignment costs and production costs at the facilities in order to meet the demand of the set of customers. Many times, one can expect that the demand level of customer j will be independent of the facility to which it is assigned, i.e., $s_{ij} = s_j$ for $i = 1, \ldots, m$ and $j = 1, \ldots, n$. Further, for many resources, the value s_{ij} is proportional to a_{ij} for all customer $j = 1, \ldots, n$. In other words, for facility i, we have that $a_{ij} = \alpha_i s_{ij}$ for $j = 1, \ldots, n$. In this situation, constraints (3–1) can be replaced by the constraints

$$\sum_{j=1}^{n} s_{ij} x_{ij} \leq \frac{b_i}{\alpha_i} \text{ for } i = 1, \ldots, m.$$

This means that there is a constraint on the amount of demand that can be assigned to a facility.

In this motivation, the cost function, $g_i(S)$, of meeting S units of demand may take the form of a wide variety of functions due to the fact that there are many types of production cost functions in supply chain management. For example, Huang et al. [85] consider continuous time single sourcing problems where the values $s_{ij} = s_j$ represent the demand rate of customer j per unit time and where the production cost function $g_i(S) = \sqrt{2K_i h_i S}$ represents the cycle stock cost associated with the classic economic order quantity. The joint-inventory model studied by Shen et al. [141] is also an NL-GAP where $s_{ij} = \mu_j$ is the mean demand of customer j and the cost function $g_i(S)$ represents the operating, inventory, and safety stock costs associated with assigning a demand level with mean S and variance γS to facility i. Note that both these problems lead to concave facility production cost functions. In many situations, the production cost functions exhibit economies of scale in production (i.e., $g_i(S)$ is concave). If there are economies of scale in production but the facility i has an internal production capacity and a subcontracting option, then the function $g_i(S)$ will be concave up until the internal capacity (representing the internal production costs) and will typically be convex after the internal capacity (representing the subcontracting costs). At the internal capacity, the

function $g_i(S)$ will often be ill-structured, such as non-differentiable or even discontinuous (for example, if there is a fixed subcontracting cost), for many practical situations.

If the customers have nonstationary demands that share a common seasonality pattern over a finite horizon and the total cost of meeting demand is measured via the solution of an economic lot-sizing problem, we can represent the cost of serving a certain base demand level, S, at facility i by the function $g_i(S)$. Freling et al. [57] assume that production in each period is capacitated and production and inventory holding costs are linear, which leads to a cost function $g_i(S)$ that is piecewise linear and convex. If, alternatively, there are fixed setup costs associated with production but production is uncapacitated, a piecewise linear concave cost function $g_i(S)$ is obtained. Under other assumptions on production and inventory cost functions and capacities the function $g_i(S)$ may be nonconvex and/or nonconcave. In these settings, note that the functions $g_i(S)$ are non-differentiable.

The NL-GAP also has applications in parallel machine scheduling. In context of this motivation for the NL-GAP, we are interested in assigning the set of jobs (indexed by $j = 1, \ldots, n$) to the machines (indexed by $i = 1, \ldots, m$). The value s_{ij} represents the span required to produce job j on machine i. In this problem, the function $g_i(S)$ is the cost of operating machine i for S units of span. When machine i deteriorates over time, i.e., the machine processes jobs at a slower rate the longer the machine is run, then the function $g_i(S)$ is convex. When the machine i requires a warmup period, i.e., the machine does not process jobs at its full capabilities until it warms up, then the function $g_i(S)$ is concave. When machine i needs an operator, whom we must pay overtime wages if he/she works over a certain amount of time, then the function $g_i(S)$ is a piecewise linear convex function.

The relaxation of the NL-GAP and related problems that will be studied in this dissertation belong to a much larger class of global optimization problems. In particular,

they belong to the class of problems

$$\text{minimize } c^\top x + g(Sx)$$

subject to (GP)

$$\begin{aligned} Ax &= b \\ x &\geq 0 \end{aligned}$$

where $x \in \mathbb{R}^n$ is the vector of decision variables, $c \in \mathbb{R}^n$, $S \in \mathbb{R}^{K \times n}$, $A \in \mathbb{R}^{m \times n}$, and $b \in \mathbb{R}^m$ are the problem data and $g : \mathbb{R}^K \to \mathbb{R}$ is a nonlinear function. We have discussed that for many practical applications of the NL-GAP, the nonlinear functions are ill-structured, i.e., non-differentiable or even discontinuous. Therefore, standard optimality conditions (such as the Karush-Kuhn-Tucker conditions) may not be applicable to the NL-GAP or GP. However, we are able to develop an important result dealing with the structure of some optimal solution to GP. This result comes from analyzing a family of linear programs that are closely related to GP. In particular, we examine the family of linear programs

$$\text{minimize } c^\top x$$

subject to (LGP)

$$\begin{aligned} Sx &= \zeta \\ Ax &= b \\ x &\geq 0 \end{aligned}$$

The analysis of LGP will play an important role in solving problems of the form GP. The following result requires no assumptions on the nonlinear function in GP and provides a structural property of some optimal solution to it.

Theorem 3.0.1. *If there exists an optimal solution to GP, there exists an optimal solution, x^*, that is an extreme point of LGP for some $\zeta \in \mathbb{R}^K$.*

Proof. Suppose that $\bar{x}$ is optimal to GP. Define the linear program LGP such that $\zeta = S\bar{x}$. Clearly, $\bar{x}$ is feasible to LGP. Moreover, any solution to LGP, say x, is feasible to GP with objective function value satisfying $p^\top x + g(Sx) = p^\top x + g(S\bar{x}) \geq p^\top \bar{x} + g(S\bar{x})$ since $\bar{x}$ is optimal to GP. Therefore, for any feasible x to LGP, $p^\top x \geq p^\top \bar{x}$, so that $\bar{x}$ is optimal to LGP and any optimal solution to LGP is also optimal to GP. Since there exists an extreme point optimal solution, x^*, to LGP, our desired result follows. □

Furthermore, duality theory from linear programming will play an important role in developing methods to solve problems of the form GP. We note that the feasible region of the dual of LGP,

$$D = \{\omega \in \mathbb{R}^m, \upsilon \in \mathbb{R}^K : A^\top \omega + S^\top \upsilon \leq c\},$$

is independent of the value of ζ. We have the following corollary to Theorem 3.0.1.

Corollary 3.0.2. *If there exists an optimal solution to GP, there exists an optimal solution, x^*, to GP and dual multipliers $(\omega^*, \upsilon^*) \in D$, that satisfy the complementary slackness conditions.*

Proof. This result follows immediately by defining x^* be the extreme point optimal solution defined by Theorem 3.0.1 and (ω^*, υ^*) be its (optimal) complementary dual solution in LGP. □

A recurring theme in this dissertation will be to apply Theorem 3.0.1 and Corollary 3.0.2 to a problem of the form GP and use the results to develop specialized algorithms to solve the problem at hand. For certain problems, these results will lead to extremely powerful specialized algorithms to solve the problem.

In Chapter 4, we will develop a procedure to solve the NL-GAP that is based on solving the continuous relaxation of it. We will use Theorem 3.0.1 to show that there exists only a small number (independent of n) of customers that are assigned fractionally to any facility in the optimal solution to the relaxation of NL-GAP. Our procedure will then convert the optimal solution of the relaxation to a feasible solution to the NL-GAP

without increasing the cost of the solution too much. We then analyze the performance of our procedure under a very general stochastic model as the number of customers tends to infinity.

In Chapter 5, we develop a method to solve the class of GP where g is a convex function. This method is inspired by the simplex method for linear programming and relies on a geometric interpretation of Theorem 3.0.1. It generates a sequence of solutions that is non-decreasing with respect to the objective function where each solution lies on a face of the constraint polyhedron of at most K dimensions. We show that an implementation of the method where we solve 1-dimensional convex optimization problems converges in cost as the number of iterations tends to infinity and an implementation of the method where we solve K-dimensional convex optimization problems terminates in a finite number of iterations. Further, a computational study demonstrates the efficiency of the algorithm and suggests that the average-case performance of these algorithms is a polynomial of low order in the number of decision variables.

Recall that if we were to solve a SCND problem through a branch and price algorithm, that the pricing problem that would arise is of the form

$$\max_{x \in \{0,1\}^n} \quad r^\top x - H(x).$$

Therefore, the pricing problem that would arise in solving the NL-GAP is of the form

$$\max \; r^\top x - g(s^\top x)$$

subject to (DKP)

$$\begin{aligned} a^\top x &\leq b \\ x &\in \{0,1\}^n. \end{aligned}$$

In Chapter 6, we will develop methods to solve the continuous relaxation of DKP and related extensions of the knapsack problem (such as the multiple knapsack problem and

the multiple-choice knapsack problem). We will develop important properties of optimal solutions for the relaxation of DKP, based on applications of Theorem 3.0.1 and Corollary 3.0.2. Using these properties, we provide a solution method that runs in polynomial time in the number of decision variables, while also depending on the time required to solve a particular one-dimensional optimization problem that is based on the function g. Thus, for the many applications in which this one-dimensional function is reasonably well behaved (e.g., unimodal), the resulting algorithm runs in polynomial time. This method to solve the continuous relaxation of DKP will play an important role in solving DKP since (i) the algorithm is theoretically and computationally efficient and (ii) the number of fractional variables in the optimal solution to the relaxation is small. We extend this solution method to solve problems where the knapsack constraints are replaced by multiple knapsack constraints or multiple-choice knapsack constraints. Computational testing demonstrates the power of the proposed algorithms over a commercial global optimization software package.

CHAPTER 4
A GREEDY PROCEDURE FOR THE NL-GAP

Many exact solution methods and heuristics for integer programming problems rely on examining the solution to the continuous relaxation of the problem. In this chapter, we will propose a greedy procedure for the NL-GAP, that is based on the optimal solution of its continuous relaxation. In particular, we develop a set of optimal dual multipliers associated with this optimal solution (using Corollary 3.0.2) and use these dual multipliers in a greedy heuristic (which results in the greedy procedure). This heuristic is similar in spirit to the class of greedy heuristics that Romeijn and Romero Morales [124] examine for the GAP. The greedy procedure itself can be interpreted as attempting to convert the optimal solution of the relaxation to a feasible integer solution without increasing the cost too much.

The remainder of this chapter is organized as follows. In Section 4.1, we examine properties of an optimal solution to the continuous relaxation of the NL-GAP. We will develop a certain set of dual multipliers associated with this optimal solution based on Corollary 3.0.2. In Section 4.1.1, we explore the relationship between these dual multipliers and the generalized KKT conditions applied to a certain formulation of the relaxation of the NL-GAP. Further, in Section 4.1.2, we show that for the case when the functions g_i, $i = 1, \ldots, m$, are convex, these dual multipliers can be determined by solving a certain Lagrangian dual problem. We propose the greedy heuristic in Section 4.2 and discuss the properties of the heuristic when it is applied with the dual multipliers developed in Section 4.1. We propose a stochastic model of the NL-GAP in Section 4.3 and discuss the average case performance of the greedy procedure. We show that the greedy procedure is asymptotically optimal and feasible when the requirements of the NL-GAP are facility-independent, i.e., $a_{ij} = a_j$, $c_{ij} = c_j$, and $s_{ij} = s_j$ for $i = 1, \ldots, m$ and $j = 1, \ldots, n$ for any set of continuous functions, g_i, $i = 1, \ldots, m$. For the case of facility-dependent requirements, we show that there exists a set of dual multipliers that

lead to an asymptotically optimal and feasible greedy procedure when the functions g_i, $i = 1, \ldots, m$, are convex. We conclude the chapter in Section 4.4 with some future research directions.

4.1 Properties of the Relaxation

The optimal solution to the continuous relaxation of the NL-GAP will play a fundamental role in developing the greedy procedure for the NL-GAP. In particular, we examine the problem

$$\text{minimize} \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} + \sum_{i=1}^{m} g_i\left(\sum_{j=1}^{n} s_{ij}x_{ij}\right)$$

subject to (NL-GAPR)

$$\begin{aligned}
\sum_{j=1}^{n} a_{ij}x_{ij} &\leq b_i && \text{for } i = 1, \ldots m && (4\text{–}1)\\
\sum_{i=1}^{m} x_{ij} &= 1 && \text{for } j = 1, \ldots n && (4\text{–}2)\\
x_{ij} &\geq 0 && \text{for } i = 1, \ldots, m,\ j = 1, \ldots, n, &
\end{aligned}$$

where the binary restrictions in the NL-GAP are replaced by the non-negativity restrictions on the variables. The following family of linear programs is closely related to the NL-GAPR and is fundamental in our analysis of it:

$$\text{minimize} \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij}$$

subject to (LNL-GAPR)

$$\begin{aligned}
\sum_{j=1}^{n} s_{ij}x_{ij} &= S_i && \text{for } i = 1, \ldots m && (4\text{–}3)\\
\sum_{j=1}^{n} a_{ij}x_{ij} &\leq b_i && \text{for } i = 1, \ldots m &\\
\sum_{i=1}^{m} x_{ij} &= 1 && \text{for } j = 1, \ldots n &\\
x_{ij} &\geq 0 && \text{for } i = 1, \ldots, m,\ j = 1, \ldots, n, &
\end{aligned}$$

Applying Theorem 3.0.1 to the NL-GAPR yields the following result.

Theorem 4.1.1. *There exists an optimal solution, x^*, to the NL-GAPR that is an optimal basic feasible solution to the LNL-GAPR for some $S \in \mathbb{R}^m$.*

For the remainder of this chapter, we will let x^* denote the optimal solution to NL-GAPR that satisfies the conditions of Theorem 4.1.1. In analyzing the properties of x^*, it will be useful to define several sets associated with it. Given a solution x that is a basic feasible solution to the LNL-GAPR for some $S \in \mathbb{R}^m$, we define

$$F_x = \{(i,j) : 0 < x_{ij} < 1\}$$

as the set of fractional variables in x and

$$S_x = \{j : \text{ there exists } i, i' \text{ such that } (i,j), (i',j) \in F_x\}$$

as the set of split customers in x. For a given basic representation of x, we also define

$$B_x = \{(i,j) : x_{ij} \text{ is basic in } x\}$$

as the set of basic variables in the representation of x,

$$M_x = \{i : \text{ the slack variable associated with the constraint } \sum_{j=1}^{n} a_{ij}x_{ij} \leq b_i \text{ is basic}\}$$

as the set of basic facilities in the representation of x, and

$$SB_x = \{j : \text{ there exists } i, i' \text{ such that } (i,j), (i',j) \in B_x\}$$

as the set of split basic customers in the representation of x. For the remainder of this chapter, when we refer to x^*, we will also be implicitly referring to the optimal basic representation of it. Theorem 4.1.1 is useful in bounding the size of these sets for x^*.

Lemma 4.1.2. *We have that $|B_{x^*}| \leq 2m + n$, $|SB_{x^*}| \leq 2m$, $|F_{x^*}| \leq 4m$, and $|S_{x^*}| \leq 2m$.*

Proof. By the definition of a basic feasible solution to LNL-GAPR, there is exactly $2m + n$ basic variables, i.e, $|B_{x^*}| + |M_{x^*}| = 2m + n$. By constraints (4–2), at least one x_{ij} must

be positive, and therefore basic, for each $j = 1, \ldots, n$. This means that we have at most $2m$ 'free' basic variables. Therefore, at most $2m$ customers can have x_{ij} and $x_{i'j}$ as basic which implies that $|SB_{x^*}| \leq 2m$. In order for $(i,j) \in F_{x^*}$, it is necessary that there exists at least one i' such that $(i,j),(i',j) \in B_{x^*}$. This implies that $|F_{x^*}| \leq 4m$ and $|S_{x^*}| \leq 2m$. □

Therefore, there exists an optimal solution to the NL-GAPR such that the number of split customers is at most $2m$, independent on the number of customers n. The idea behind the greedy procedure for the NL-GAP is to reassign only the split (or split basic) customers from x^*. In order to accomplish this, we will next develop a set of dual multipliers associated with x^* that help in guaranteeing that all non-split jobs are properly assigned in the greedy procedure. As was the case with the feasible region of the dual of LGP, the feasible region of the dual of LNL-GAPR is independent of S. In particular, it is given by

$$D = \left\{ \begin{array}{l} v_j \leq c_{ij} + a_{ij}\lambda_i - s_{ij}\gamma_i \text{ for } i = 1, \ldots, m,\ j = 1, \ldots, n \\ \lambda \in \mathbb{R}^m_+,\ \gamma \in \mathbb{R}^m,\ v \in \mathbb{R}^n \end{array} \right\}.$$

The following result helps characterize the relationship between x^* and its (optimal) complementary dual solution from Corollary 3.0.2.

Lemma 4.1.3. *If D is non-degenerate, then there exists a feasible dual solution $(\lambda^*, \gamma^*, v^*)$ with the property that:*

(i) For each $j \notin SB_{x^*}$, $x^*_{ij} = 1$ implies that

$$v^*_j = \min_i c_{ij} + a_{ij}\lambda^*_i - s_{ij}\gamma^*_i$$

and for all $k \neq i$

$$v^*_j < c_{kj} + a_{kj}\lambda^*_k - s_{kj}\gamma^*_k.$$

(ii) For each $j \in SB_{x^*}$, there exists some $i = 1, \ldots m$ such that:

$$c_{ij} + a_{ij}\lambda^*_i - s_{ij}\gamma^*_i = \min_{k \neq i} c_{kj} + a_{kj}\lambda^*_k - s_{kj}\gamma^*_k.$$

Proof. By Corollary 3.0.2, there exists a feasible dual solution, $(\lambda^*, \gamma^*, v^*)$ that satisfies the complementary slackness conditions with x^*. If $j \in SB_{x^*}$, there must exist two indices i, k such that are basic. By complementary slackness, $v_j^* = c_{ij} + a_{ij}\lambda_k^* - \gamma_i^* s_{ij} = c_{kj} + a_{kj}\lambda_k^* - s_{kj}\gamma_k^*$. Since $(\lambda^*, \gamma^*, v^*)$ is dual feasible, we have that $v_j^* \leq c_{i'j} + a_{i'j}\lambda_{i'}^* - s_{i'j}\gamma_{i'}^*$ for all i'. This proves (ii).

We will now focus on proving (i). By complementary slackness and the feasibility of x^*, for every j, there must exist i such that $v_j^* = c_{ij} + a_{ij}\lambda_i^* - s_{ij}\gamma_i^*$ and by dual feasibility we have $v_j^* \leq c_{ij} + a_{ij}\lambda_i^* - s_{ij}\gamma_i^*$ for all i. Therefore, $v_j^* = \min_i c_{ij} + a_{ij}\lambda_i^* - s_{ij}\gamma_i^*$. It remains to be shown that for $j \notin SB_{x^*}$, there is a single facility i that achieves this minimum or, equivalently, has $v_j^* = c_{ij} + a_{ij}\lambda_i^* - s_{ij}\gamma_i^*$. Since $j \notin SB_{x^*}$, there is a single i such that $(i, j) \in B_{x^*}$. Note that a basis of D can be characterized by a selection of $2m + n$ active constraints. The choices $(i, j) \in B_{x^*}$ and $i \in M_{x^*}$ form such a selection due to the complementary slackness principle. Since D is non-degenerate, we have that only the hyperplanes $v_j = c_{ij} + a_{ij}\lambda_i^* - s_{ij}\gamma_i^*$ for $(i, j) \in B_{x^*}$ and $\lambda_i = 0$ for $i \in M_{x^*}$ may be active at $(\lambda^*, \gamma^*, v^*)$. Therefore, if $j \notin SB_{x^*}$, there exists a single i such that $v_j^* = c_{ij} + a_{ij}\lambda_i^* - s_{ij}\gamma_i^*$ which proves (i). □

The greedy procedure that will be developed in Section 4.2 will use the dual solution (λ^*, γ^*) that satisfies Lemma 4.1.3. In principle, determining the optimal solution to NL-GAPR may be a difficult problem. For example, the NL-GAPR with concave functions g_i, $i = 1, \ldots, m$, and no capacity constraints is NP-hard. However, for a certain class of functions g_i, $i = 1, \ldots, m$, we will show that obtaining (λ^*, γ^*) can be done by solving a certain Lagrangian relaxation of NL-GAPR. In the remainder of this section, we will explore the relationship between some fundamental concepts (such as KKT conditions and Lagrangian relaxation) in nonlinear programming and the problem LNL-GAPR.

4.1.1 Relationship Between LNL-GAPR and the KKT Conditions

In this section, we discuss the relationship between the problem LNL-GAPR and the optimality conditions of a related problem to NL-GAPR. In particular, we examine the

related problem

$$\text{minimize } f(x,S) = \sum_{i=1}^{m}\left(\sum_{j=1}^{n} c_{ij}x_{ij} + g_i(S_i)\right)$$

subject to (NL-GAPR′)

$$\sum_{j=1}^{n} s_{ij}x_{ij} = S_i \qquad \text{for } i = 1,\dots m \tag{4–4}$$

$$\sum_{j=1}^{n} a_{ij}x_{ij} \leq b_i \qquad \text{for } i = 1,\dots m \tag{4–5}$$

$$\sum_{i=1}^{m} x_{ij} = 1 \qquad \text{for } j = 1,\dots n \tag{4–6}$$

$$x_{ij} \geq 0 \qquad \text{for } i = 1,\dots,m,\ j = 1,\dots,n, \tag{4–7}$$

where the main difference between NL-GAPR′ and LNL-GAPR is that S is now a vector of decision variables in NL-GAPR′. It is clear that the problems NL-GAPR and NL-GAPR′ are equivalent. Assuming the functions g_i, $i = 1,\dots,m$, are locally Lipschitz continuous, the generalized KKT conditions (see Hiriart-Urruty [79]), which are necessary but typically not sufficient for optimality, for NL-GAPR′ are

$$c_{ij} + a_{ij}\lambda_i - s_{ij}\gamma_i - v_j - \mu_{ij} = 0 \qquad \text{for } i = 1,\dots,m,\ j = 1,\dots,n \tag{4–8}$$

$$\partial g_i(S_i) + \gamma_i \ni 0 \qquad \text{for } i = 1,\dots,m \tag{4–9}$$

$$\mu_{ij} \geq 0 \qquad \text{for } i = 1,\dots,m,\ j = 1,\dots,n \tag{4–10}$$

$$\lambda_i \geq 0 \qquad \text{for } i = 1,\dots,m \tag{4–11}$$

$$\mu_{ij}x_{ij} = 0 \qquad \text{for } i = 1,\dots,m,\ j = 1,\dots,n \tag{4–12}$$

$$\left(\sum_{j=1}^{n} a_{ij}x_{ij} - b_i\right)\lambda_i = 0 \qquad \text{for } i = 1,\dots,m \tag{4–13}$$

$$\left(\sum_{j=1}^{n} s_{ij}x_{ij} - S_i\right)\gamma_i = 0 \qquad \text{for } i = 1,\dots,m \tag{4–14}$$

$$\left(\sum_{i=1}^{m} x_{ij} - 1\right)v_j = 0 \qquad \text{for } j = 1,\dots,n \tag{4–15}$$

$$\sum_{j=1}^{n} s_{ij}x_{ij} = S_i \quad \text{for } i = 1, \dots m \tag{4–16}$$

$$\sum_{j=1}^{n} a_{ij}x_{ij} \leq b_i \quad \text{for } i = 1, \dots m \tag{4–17}$$

$$\sum_{i=1}^{m} x_{ij} = 1 \quad \text{for } j = 1, \dots n \tag{4–18}$$

$$x_{ij} \geq 0 \quad \text{for } i = 1, \dots, m,\ j = 1, \dots, n. \tag{4–19}$$

We will now show an intuitively appealing relationship between any solution to the generalized KKT conditions to NL-GAPR′ and the LNL-GAPR.

Theorem 4.1.4. *If (x, S) and $(\lambda, \gamma, v, \mu)$ satisfy the generalized KKT conditions (4–8)-(4–19), then x is optimal to the LNL-GAPR with values S and (λ, γ, v) is optimal to the dual of LNL-GAPR.*

Proof. By conditions (4–16)-(4–19), x is feasible to the LNL-GAPR with values S. We will first show that $(\lambda,\ \gamma,\ v)$ is feasible to the dual of LNL-GAPR. Recall that the feasible region of the dual of LNL-GAPR is given by

$$v'_j \leq c_{ij} + a_{ij}\lambda'_i - s_{ij}\gamma'_i \text{ for } i = 1, \dots, m,\ j = 1, \dots, n \tag{4–20}$$

$$\lambda' \in \mathbb{R}^m_+,\ \gamma' \in \mathbb{R}^m,\ v' \in \mathbb{R}^n. \tag{4–21}$$

By condition (4–8), we have that

$$c_{ij} + a_{ij}\lambda_i - s_{ij}\gamma_i - v_j = \mu_{ij} \geq 0,$$

where the inequality follows from condition (4–10). This implies that (4–20) is satisfied. Further, (4–21) follows from condition (4–11). Therefore, $(\lambda,\ \gamma,\ v)$ is dual feasible. If we show that x and $(\lambda,\ \gamma,\ v)$ satisfy the complementary slackness conditions applied to LNL-GAPR, then our desired result follows. Note that if $x_{ij} > 0$, it follows from condition (4–12) that $\mu_{ij} = 0$. By condition (4–8), we have that

$$c_{ij} + a_{ij}\lambda_i - s_{ij}\gamma_i - v_j = 0.$$

This implies that for all $i = 1, \ldots, m$ and $j = 1, \ldots, n$ that

$$(c_{ij} + a_{ij}\lambda_i - s_{ij}\gamma_i - v_j)x_{ij} = 0.$$

The remainder of the complementary slackness conditions of LNL-GAPR are exactly conditions (4–13)-(4–15). Therefore, x satisfies the complementary slackness conditions of LNL-GAPR with (λ, γ, v). The desired result follows since (λ, γ, v) is dual feasible. □

This result says that if we have a solution to the generalized KKT conditions, then the primal solution is optimal to the associated LNL-GAPR. Although this is an interesting result, it may only be useful for the case when the functions g_i, $i = 1, \ldots, m$ are convex, since the KKT conditions are sufficient for optimality to NL-GAPR′ in this situation.

4.1.2 Relationship Between LNL-GAPR and a Lagrangian Relaxation

In this section, we consider the relationship between the optimal dual multipliers of NL-GAPR and the optimal solution to a certain Lagrangian relaxation of NL-GAPR′. Let X be the set of continuous assignment constraints and consider dualizing the nonlinear argument constraints (4–4) and the capacity constraints (4–5) from the NL-GAPR′. For a fixed λ, γ, we define the function $L(\lambda, \gamma)$ to be equal to the optimal solution value of the problem

$$\min_{x \in X, S \in \mathbb{R}^m} \sum_{i=1}^{m} \left(\sum_{j=1}^{n} c_{ij}x_{ij} + g_i(S_i) \right) + \sum_{i=1}^{m} \left(\lambda_i \left(\sum_{j=1}^{n} a_{ij}x_{ij} - b_i \right) + \gamma_i \left(S_i - \sum_{j=1}^{n} s_{ij}x_{ij} \right) \right).$$

For any $\lambda \in \mathbb{R}^+_m$ and $\gamma \in \mathbb{R}^m$, $L(\lambda, \gamma)$ is a lower bound on the optimal solution value of NL-GAPR′. Note that this problem decomposes by each customer and each facility, i.e., it can be written as

$$\min_{x \in X, S \in \mathbb{R}^m} \sum_{i=1}^{m} \left(\sum_{j=1}^{n} (c_{ij} + a_{ij}\lambda_i - s_{ij}\gamma_i)x_{ij} + g_i(S_i) + \gamma_i S_i + \lambda_i b_i \right) =$$

$$\sum_{j=1}^{n} \min_{i=1,\ldots,m} (c_{ij} + a_{ij}\lambda_i - s_{ij}\gamma_i) + \sum_{i=1}^{m} \lambda_i b_i + \sum_{i=1}^{m} \min_{S_i \in \mathbb{R}} (g_i(S_i) + \gamma_i S_i). \tag{4–22}$$

Therefore, for a fixed λ, γ, we can determine the value of the function $L(\lambda, \gamma)$ in $O(nm + m\phi) = O(m \max\{n, \phi\})$ time where ϕ is the time required to solve a one-dimensional optimization problem of the form $\min_{S \in \mathbb{R}}(g_i(S) + \gamma S)$. The Lagrangian dual problem is then defined as

$$\max_{\lambda \in \mathbb{R}^+_m, \gamma \in \mathbb{R}^m} L(\lambda, \gamma). \tag{4–23}$$

There are well known methods to solve the Lagrangian dual problem. These methods are especially attractive if the inner problem, i.e., $L(\lambda, \gamma)$ can be solved efficiently. For most functions g_i (for example, convex, unimodal, or concave functions), we will be able to solve $L(\lambda, \gamma)$ efficiently. For certain classes of functions, we also have the desirable property that the optimal solution to (4–23) also satisfies Lemma 4.1.3 with the optimal solution x^* to NL-GAPR.

Theorem 4.1.5. *If there is no duality gap between NL-GAPR′ and the Lagrangian dual problem, (4–23), then the optimal dual solution, (λ^*, γ^*), to (4–23) is part of the complementary dual solution of the optimal solution x^*.*

Proof. Since x^* is optimal to NL-GAPR, the solution (x^*, S^*), where

$$S_i^* = \sum_{j=1}^{n} s_{ij} x_{ij}^* \text{ for } i = 1, \ldots, m,$$

is optimal to NL-GAPR′. Furthermore, if there is no duality gap between NL-GAPR′ and the Lagrangian dual problem, i.e., $f(x^*, S^*) = L(\lambda^*, \gamma^*)$, then by Propositions 5.1.1 and 5.1.4 of Bertsekas [18], we have that (x^*, S^*) and (λ^*, γ^*) satisfy the complementary slackness with respect to the dualized constraints,

$$\left(\sum_{j=1}^{n} a_{ij} x_{ij}^* - b_i \right) \lambda_i^* = 0 \quad \text{for } i = 1, \ldots, m \tag{4–24}$$

$$\left(\sum_{j=1}^{n} s_{ij} x_{ij}^* - S_i^* \right) \gamma_i^* = 0 \quad \text{for } i = 1, \ldots, m, \tag{4–25}$$

and

$$(x^*, S^*) = \arg \min_{x \in X, S \in \mathbb{R}^m} \sum_{i=1}^{m} \left(\sum_{j=1}^{n} (c_{ij} + a_{ij}\lambda_i - s_{ij}\gamma_i) x_{ij} + g_i(S_i) + \gamma_i S_i + \lambda_i b_i \right). \quad (4\text{–}26)$$

We now must define the dual variables v^* to have a complete dual solution $(\lambda^*, \gamma^*, v^*)$. In particular, we define

$$v_j^* = \min_{i=1,\ldots,m} (c_{ij} + a_{ij}\lambda_i^* - s_{ij}\gamma_i^*) \text{ for } j = 1, \ldots, n.$$

By the definition of v^*, the solution $(\lambda^*, \gamma^*, v^*) \in D$, i.e., it is feasible to the dual of LNL-GAPR. Further, if $x_{ij}^* > 0$, equations (4–22) and (4–26) imply that

$$c_{ij} + a_{ij}\lambda_i^* - s_{ij}\gamma_i^* = \min_k c_{kj} + a_{kj}\lambda_k^* - s_{kj}\gamma_k^* = v_j^*.$$

Therefore, for any i and j, we have that

$$\left(c_{ij} + a_{ij}\lambda_i^* - s_{ij}\gamma_i^* - v_j^*\right) x_{ij}^* = 0. \quad (4\text{–}27)$$

Equations (4–24), (4–25), (4–27) and primal feasibility of x^* imply the complementary slackness conditions of the LNL-GAPR defined with S^*. Note the dual feasibility of $(\lambda^*, \gamma^*, v^*)$ implies that this solution is the complementary dual solution to x^*. □

This result implies that determining (λ^*, γ^*) satisfying the conditions of Lemma 4.1.3 is equivalent to solving the Lagrangian dual problem when the functions g_i, $i = 1, \ldots, m$, are convex. This means that for this special class of the NL-GAP, we simply need to focus on the Lagrangian dual problem to obtain (λ^*, γ^*) and do not necessarily need to determine the optimal solution x^* to the NL-GAPR.

4.2 A Greedy Heuristic

We now propose a heuristic for the NL-GAP which is similar to the one proposed in Romeijn and Morales [124] for the GAP. We define a weight function, $f(i, j)$, representing a pseudo-cost of assigning customer j to facility i. We define the *desirability* of a customer

as

$$\rho_j = \max_i \min_{k \neq j} (f(i,k) - f(i,j)).$$

This means that ρ_j represents the cost of not assigning customer j to its most desirable facility. For any vectors $\lambda \in \mathbb{R}^m_+$ and $\gamma \in \mathbb{R}^m$, we define the weight function to be

$$f_{(\lambda,\gamma)}(i,j) = c_{ij} + a_{ij}\lambda_i - s_{ij}\gamma_i.$$

The greedy algorithm chooses the customer j with the highest value ρ_j and assigns it to its most desirable facility, if the assignment does not violate the capacity constraint of the facility. For a given (λ, γ), we then perform the following greedy algorithm (This presentation is the same as the Modified Greedy Algorithm in [124]):

Greedy Algorithm

Step 0. Set $J = \{1, \ldots, n\}$, $b'_i = b_i$ for $i = 1, \ldots, m$, and $\mathcal{F}_j = \{1, \ldots, m\}$ for $j = 1, \ldots n$.

Step 1. If $\mathcal{F}_j = \emptyset$ for some $j \in J$, terminate since the algorithm could not find a feasible solution. Otherwise, set

$$i_j = \arg\min_{i \in \mathcal{F}_j} f(i,j) \text{ for } j \in J,$$

$$\rho_j = \min_{k \in \mathcal{F}_j \setminus \{i_j\}} f(k,j) - f(i_j,j) \text{ for } j \in J.$$

Step 2. Let $\hat{j} = \arg\max_{j \in J} \rho_j$, i.e., $\hat{j}$ is the customer to be assigned next. If $a_{i_{\hat{j}}\hat{j}} > b'_{i_{\hat{j}}}$, then the assignment is not feasible. Set $\mathcal{F}_j = \{i : a_{ij} \leq b'_i\}$ for $j \in J$ and go to Step 1. Otherwise, set $x^G_{i_{\hat{j}}\hat{j}} = 1$, $x^G_{i\hat{j}} = 0$ for $i \neq i_{\hat{j}}$, $b'_{i_{\hat{j}}} = b'_{i_{\hat{j}}} - a_{i_{\hat{j}}\hat{j}}$, and $J = J \setminus \{\hat{j}\}$.

Step 3. If $J = \emptyset$, terminate the algorithm with a feasible solution to the NL-GAP, x^G. Otherwise go to Step 1.

Note that the Greedy Algorithm is presented in such a way that we have flexibility in choosing the values for (λ, γ). We refer to the Greedy Algorithm applied with (λ^*, γ^*) from Lemma 4.1.3 as the *greedy procedure* for the NL-GAP. The reason that we refer to this as a 'procedure' rather than 'heuristic' is that determining (λ^*, γ^*) may be a difficult

problem itself. We are now interested in analyzing the properties of the greedy procedure. Define NS_{x^*} to be the set of non-split basic customers in x^*, i.e., $NS_{x^*} = \{1, \ldots, n\} \backslash SB_{x^*}$.

Lemma 4.2.1. *Suppose that D is non-degenerate and we perform the greedy procedure. For all $j \in NS_{x^*}$, we have that $x^*_{ij} = x^G_{ij}$.*

Proof. Since $j \in NS_{x^*}$, we have that $\rho_j > 0$ by Lemma 4.1.3. Further, for any $j' \in SB_{x^*}$, we have that $\rho_j = 0$. Therefore, we will consider j before any $j' \in SB_{x^*}$ in the greedy algorithm. Since x^* does not violate any of the capacity constraints, the partial solution given by x^*_{ij} for $j \in NS_{x^*}$ will not violate the capacity constraints either. This implies that we will assign all $j \in NS_{x^*}$ to its most desirable facility, i.e $i_j = \arg\min_{i=1,\ldots m} c_{ij} - a_{ij}\lambda_i - s_{ij}\gamma_i$. By Lemma 4.1.3, we have that i_j will be the facility which customer i is assigned to in x^*. □

If we define $\bar{C} = \max_{(i,j)} c_{ij}$ and $\bar{S} = \max_{(i,j)} s_{ij}$, we can provide a performance guarantee on the greedy procedure. The following result will be quite useful in analyzing the greedy procedure under a stochastic model.

Lemma 4.2.2. *If x^G is feasible, we have that*

(i) $\sum_{i=1}^m \sum_{j=1}^n c_{ij} x^G_{ij} \leq 2m\bar{C} + \sum_{i=1}^m \sum_{j=1}^n c_{ij} x^*_{ij}$ *and*

(ii) For any facility i, $\sum_{j=1}^n s_{ij} x^G_{ij} \in \left[\sum_{j=1}^n s_{ij} x^*_{ij} - 2m\bar{S}, \sum_{j=1}^n s_{ij} x^*_{ij} + 2m\bar{S}\right]$.

Proof. Lemma 4.2.1 says that we can reassign up to $2m$ customers in obtaining x^G from x^*. Therefore, at worst, we can reassign each of these customers to a facility such that $c_{ij} = \bar{C}$. This implies (i). For a particular facility, we can fully assign (or fully remove) up to $2m$ new customers in obtaining x^G from x^*. This implies that (ii) holds. □

4.3 Stochastic Models of the NL-GAP

We are interested in analyzing the performance of the greedy procedure for the NL-GAP as the number of jobs, n, goes to infinity while the number of jobs, m, remains fixed for a stochastic model of the problem parameters. Let $A_j = (A_{1j}, \ldots, A_{mj})$ be i.i.d. random vectors in the bounded set $[A_L, A_U]^m$, $C_j = (C_{1j}, \ldots C_{mj})$ be i.i.d random

vectors in the set $[C_L, C_U]^m$ and $S_j = (S_{1j}, \ldots S_{mj})$ be i.i.d random vectors in $[S_L, S_U]^m$ (with $C_L, S_L \geq 0$). These definitions follow along the lines of Romeijn and Piersma [123]. Further, we will let b_i depend linearly on n, i.e. $b_i = \beta_i n$ for some $\beta_i > 0$, which was done in Dyer and Frieze [46] and Romeijn and Piersma [123]. In other words, constraints (4–1) can be written as either

$$\sum_{j=1}^{n} a_{ij} x_{ij} \leq \beta_i n \text{ or } \frac{1}{n} \sum_{j=1}^{n} a_{ij} x_{ij} \leq \beta_i. \tag{4–28}$$

The stochastic model of the NL-GAP that we examine in this section normalizes the *arguments* of the nonlinear functions. In particular, we are interested in examining the problem

$$\text{minimize } f_n(x) = \sum_{i=1}^{m} \left(\frac{1}{n} \sum_{j=1}^{n} c_{ij} x_{ij} + g_i \left(\frac{1}{n} \sum_{j=1}^{n} s_{ij} x_{ij} \right) \right)$$

subject to (NA(n))

$$\begin{aligned} \frac{1}{n} \sum_{j=1}^{n} a_{ij} x_{ij} &\leq \beta_i && \text{for } i = 1, \ldots, m \\ \sum_{i=1}^{m} x_{ij} &= 1 && \text{for } j = 1, \ldots, n \\ x_{ij} &\in \{0, 1\} && \text{for } i = 1, \ldots, m,\ j = 1, \ldots, n \end{aligned}$$

As done in previous sections, NAR(n) is the continuous relaxation of NA(n). The motivation for normalizing the arguments of the nonlinear functions in this stochastic model (as opposed to normalizing the value of the function) is that the capabilities of a facility will often depend on the number of customers in the problem. For example, in equation (4–28), the capacity (such as storage space or labor hours) of the facility grows as the company becomes larger (i.e., it has more and more customers). In some ways, we can view this as the size of the facilities of the company growing as the number of customers becomes larger. Therefore, the processing capabilities (such as the number of production lines and/or the number of workers) of the facilities also grows as the number of jobs becomes larger, so that the facility can produce more demand for similar prices.

We also note that equation (4–28) is standard in asymptotic analysis of the GAP and that normalizing the arguments can be thought of as generalizing equation (4–28) to the nonlinear portion of the NL-GAP.

We are now in a position to examine the asymptotic feasibility/optimality of the greedy procedure. We say that the greedy procedure is *asymptotically feasible* if x^G is feasible to NA(n) with probability 1 as $n \to \infty$. We will say that the procedure is *asymptotically optimal* if

$$\lim_{n\to\infty} f_n(x^G) - f_n(x^*) = 0.$$

Our analysis of the asymptotic feasibility and optimality of the greedy procedure will be organized as follows. We will begin by assuming the asymptotic feasibility of the greedy procedure, i.e., we assume that the following condition holds:

Condition 4.3.1. *As $n \to \infty$, x^G is feasible to NA(n) with probability* 1.

It turns out that proving that Condition 4.3.1 holds is more difficult than proving that the greedy procedure is asymptotically optimal under Condition 4.3.1. Therefore, we will first prove asymptotic optimality of the greedy procedure under Condition 4.3.1 and then discuss situations under which the condition does indeed hold. Our first preliminary result establishes that Lemma 4.2.1 can be applied with probability 1.

Lemma 4.3.2. *Under the proposed stochastic model of the NL-GAP, the feasible region of the dual of LNL-GAPR, D, is non-degenerate with probability* 1.

Proof. This follows directly from the fact that the parameters c_{ij}, a_{ij}, and s_{ij} for $i = 1, \ldots, m$ and $j = 1, \ldots, n$ are continuous random variables. □

This lemma is useful in proving the greedy procedure is asymptotically optimal under Condition 4.3.1.

Theorem 4.3.3. *Under Condition 4.3.1 and continuous functions g_i, $i = 1, \ldots, m$, the greedy procedure is asymptotically optimal.*

Proof. By Lemmas 4.2.2 and 4.3.2, we know that

$$f_n(x^G) \leq \frac{2mC_U}{n} + \frac{1}{n}\sum_{j=1}^{n} c_{ij}x^*_{ij} + \sum_{i=1}^{m} \max_{z\in[-\frac{2mS_U}{n},\frac{2mS_U}{n}]} g_i\left(z + \frac{1}{n}\sum_{j=1}^{n} s_{ij}x^*_{ij}\right).$$

which implies that

$$f_n(x^G) - f_n(x^*) \leq \frac{2mC_U}{n} + \sum_{i=1}^{m}\left(\max_{z\in[-\frac{2mS_U}{n},\frac{2mS_U}{n}]} g_i\left(z + \frac{1}{n}\sum_{j=1}^{n} s_{ij}x^*_{ij}\right) - g_i\left(\frac{1}{n}\sum_{j=1}^{n} s_{ij}x^*_{ij}\right)\right).$$

Note that for any feasible solution to any NAR(n), we have that $\frac{1}{n}\sum_{j=1}^{n} s_{ij}x_{ij} \in [0, S_U]$. Since the function $g_i(S_i)$ is continuous and the interval $[0, S_U]$ is compact, the function $g_i : [0, S_U] \to \mathbb{R}$ is uniformly continuous (see, for example, Munkres [105]). This implies that for any $\epsilon > 0$, there exists some $n(\epsilon, i)$ such that for every $n \geq n(\epsilon, i)$ and $\varsigma \in [0, S_U]$ that

$$\max_{z\in[\frac{-2mS_U}{n},\frac{2mS_U}{n}]} g_i(z+\varsigma) - g_i(\varsigma) < \epsilon.$$

Therefore, for any $\epsilon > 0$, there exists some $n(\epsilon) = \max_{i=1,\ldots,m} n(\epsilon, i)$ such that for every $n \geq n(\epsilon)$, $\varsigma \in [0, S_U]$, and $i = 1, \ldots, m$,

$$\max_{z\in[\frac{-2mS_U}{n},\frac{2mS_U}{n}]} g_i(z+\varsigma) - g_i(\varsigma) < \epsilon.$$

This implies that for any $\epsilon > 0$, there exists n_0 such that for every $n \geq n_0$,

$$f_n(x^G) - f_n(x^*) \leq \frac{2mC_U}{n} + \sum_{i=1}^{m}\left(\max_{z\in[-\frac{2mS_U}{n},\frac{2mS_U}{n}]} g_i\left(z + \frac{1}{n}\sum_{j=1}^{n} s_{ij}x^*_{ij}\right) - g_i\left(\frac{1}{n}\sum_{j=1}^{n} s_{ij}x^*_{ij}\right)\right) < \epsilon.$$

Since $f_n(x^G) - f_n(x^*) \geq 0$ (because x^G is feasible to NAR(n)) and the definition of the limit, we have that

$$\lim_{n\to\infty} f_n(x^G) - f_n(x^*) = 0.$$

□

We will now explore the asymptotic feasibility of the greedy procedure or, equivalently, situations where Condition 4.3.1 holds. Romeijn and Piersma [123] show that the problems NA(n) and NAR(n) are not necessarily feasible for the stochastic model

described above. The following assumption guarantees that both problems are feasible with probability 1 as $n \to \infty$ and infeasible with probability 1 as $n \to \infty$ if the assumption is violated.

Assumption 4.3.4. *The excess capacity,*

$$\Delta = \min_{\lambda \in \Omega} \left(\lambda^\top \beta - E(\min_i \lambda_i A_{i1}) \right),$$

(where Ω is the unit simplex) is strictly positive.

Theorem 4.3.5. *(Romeijn and Piersma [123]) As $n \to \infty$, NA(n) and NAR(n) are feasible with probability 1 if $\Delta > 0$ and is infeasible with probability 1 if $\Delta < 0$.*

Therefore, in our probabilistic analysis of NA(n), it is appropriate to assume that Assumption 4.3.4 holds. In the following sections, we analyze the feasibility of the procedure under two separate models, facility-independent parameters and facility-dependent parameters.

4.3.1 Facility-Independent Parameters

In this section, we will consider problems characterized by facility-independent parameters, i.e., $C_{ij} = C_{1j}$, $A_{ij} = A_{1j}$, and $S_{ij} = S_{1j}$. We will show that selecting the optimal dual solution from the NAR(n) leads to an asymptotically feasible procedure. This in combination with Theorem 4.3.3 proves that the greedy procedure with $(\lambda, \gamma) = (\lambda^*, \gamma^*)$ is asymptotically feasible and optimal. We first begin with a preliminary result that deals with the unused capacity of any feasible solution to NAR(n).

Lemma 4.3.6. *When the parameters are facility-independent, the aggregate unused capacity of any feasible solution to NAR(n) grows linearly with n with probability 1 as $n \to \infty$.*

Proof. For the facility-independent case, Romeijn and Piersma [123] show that Assumption 4.3.4 is equivalent to the condition

$$E(A_{1j}) < \sum_{i=1}^{m} \beta_i. \tag{4–29}$$

For any continuous assignment vector, x, that is feasible to NAR(n), the normalized aggregate unused capacity of any feasible solution is given by

$$\begin{aligned}\sum_{i=1}^{m}\left(\beta_i - \frac{1}{n}\sum_{j=1}^{n} A_{ij}x_{ij}\right) &= \sum_{i=1}^{m}\beta_i - \frac{1}{n}\sum_{i=1}^{m}\sum_{j=1}^{n} A_{1j}x_{ij} \\ &= \sum_{i=1}^{m}\beta_i - \frac{1}{n}\sum_{j=1}^{n} A_{1j}\left(\sum_{i=1}^{m} x_{ij}\right) \\ &= \sum_{i=1}^{m}\beta_i - \frac{1}{n}\sum_{j=1}^{n} A_{1j} \\ &> 0\end{aligned}$$

with probability 1 as $n \to \infty$, by equation (4–29) and the Central Limit Theorem. □

We can now prove our main result about the greedy procedure applied to the stochastic model of the NL-GAP, NA(n), with facility-independent parameters.

Theorem 4.3.7. *When the parameters are facility-independent, if we set* $(\lambda, \gamma) = (\lambda^*, \gamma^*)$, *then the greedy heuristic is asymptotically feasible and optimal with probability* 1 *as* $n \to \infty$.

Proof. In the stochastic model, Lemma 4.3.2 shows that D is non-degenerate. Therefore, Lemma 4.2.1 holds and we only reassign split basic jobs to obtain x^G from x^*. The number of these reassignments, i.e., the number of split basic jobs, is independent of n since Lemma 4.1.2 shows that $|SB_{x^*}| \leq 2m$. By Lemma 4.3.6, the unused aggregate capacity of non-split facilities grows linearly in n with probability 1 as $n \to \infty$ and therefore, grows sufficiently large to accommodate the reassignments that need to be made to obtain x^G (since this number is independent of n). Therefore, the greedy procedure is asymptotically feasible and Condition 4.3.1 holds. Theorem 4.3.3 shows that the greedy procedure is also optimal. □

4.3.2 Facility-Dependent Parameters

We will now consider the more general situation where the parameters are allowed to vary between facilities. This is more difficult than the situation in Sec 4.3.1 since we

cannot (necessarily) guarantee that the unused capacity for any solution to NAR(n) grows linearly in n. Therefore, rather than focus on the greedy procedure as it was defined in Section 4.2, we will explore an alternative procedure that is based on a different set of dual multipliers. To do so, we examine a problem where we slightly perturb the capacity constraints,

$$\text{minimize } f_n(x) = \sum_{i=1}^{m} \left(\frac{1}{n} \sum_{j=1}^{n} c_{ij} x_{ij} + g_i \left(\frac{1}{n} \sum_{j=1}^{n} s_{ij} x_{ij} \right) \right)$$

subject to (NAR(n, δ_n))

$$\begin{aligned} \frac{1}{n} \sum_{j=1}^{n} a_{ij} x_{ij} &\leq \beta_i - \delta_n \qquad \text{for } i = 1, \ldots, m \\ \sum_{i=1}^{m} x_{ij} &= 1 \qquad\qquad \text{for } j = 1, \ldots, n \\ x_{ij} &\geq 0. \end{aligned}$$

We will choose the sequence of δ_n to have the following properties:

(i) $\lim_{n\to\infty} \delta_n = 0$,

(ii) $\lim_{n\to\infty} n\delta_n = \infty$, and

(iii) $0 < \delta_n \leq \delta < \Delta$.

We will show that (i) implies that NAR(n) and NAR(n, δ_n) behave the same asymptotically for a certain class of problems. In other words, let x^* be the optimal solution to NAR(n) and let $x^*(\delta_n)$ be the optimal solution to NAR(n, δ_n), then we have the following result.

Lemma 4.3.8. *If the functions g_i, $i = 1, \ldots, m$, are convex, then*

$$\lim_{n\to\infty} f_n(x^*) = \lim_{n\to\infty} f_n(x^*(\delta_n)).$$

Proof. Since the functions g_i, $i = 1, \ldots, m$ are convex, there is no duality gap when considering NAR(n) (or NAR(n, δ_n)) and its Lagrangian dual problem. Therefore, we have that

$$f_n(x^*(\delta_n)) = \max_{\lambda \geq 0} \Phi_n(\lambda, \delta_n),$$

where

$$\begin{aligned}\Phi_n(\lambda,\delta_n) &= \min_{x\in X}\left(\sum_{i=1}^{m}\left(\sum_{j=1}^{n}\frac{1}{n}(c_{ij}+\lambda_i a_{ij})x_{ij}+g_i\left(\frac{1}{n}\sum_{j=1}^{n}s_{ij}x_{ij}\right)\right)\right)+\sum_{i=1}^{m}\lambda_i(-\beta_i+\delta_n)\\ &= \Phi_n(\lambda,0)+\delta_n\sum_{i=1}^{m}\lambda_i. \qquad (4\text{–}30)\end{aligned}$$

and X is the set of continuous assignment constraints. We will show that we may restrict ourselves to solutions λ that lie in a compact set in maximizing $\Phi_n(\lambda,\delta_n)$. First, note that we have

$$\max_{\lambda\geq 0}\Phi_n(\lambda,\delta)\geq\Phi_n(0,\delta)\geq C_L+\sum_{i=1}^{m}\min_{S_i\in[S_L,S_U]}g_i(S_i). \qquad (4\text{–}31)$$

Further, for any λ, we have that

$$\begin{aligned}\Phi_n(\lambda,\delta) &= \min_{x\in X}\left(\sum_{i=1}^{m}\left(\sum_{j=1}^{n}\frac{1}{n}(c_{ij}+\lambda_i a_{ij})x_{ij}+g_i\left(\frac{1}{n}\sum_{j=1}^{n}s_{ij}x_{ij}\right)\right)\right)-\sum_{i=1}^{m}\lambda_i\beta_i+\sum_{i=1}^{m}\lambda_i\delta\\ &\leq \min_{x\in X}\left(\sum_{i=1}^{m}\sum_{j=1}^{n}\frac{1}{n}(c_{ij}+\lambda_i a_{ij})x_{ij}\right)+\sum_{i=1}^{m}\max_{S_i\in[S_L,S_U]}g_i(S_i)-\sum_{i=1}^{m}\lambda_i\beta_i+\delta\sum_{i=1}^{m}\lambda_i\\ &= \frac{1}{n}\sum_{j=1}^{n}\min_i(c_{ij}+\lambda_i a_{ij})+\sum_{i=1}^{m}\max_{S_i\in[S_L,S_U]}g_i(S_i)-\sum_{i=1}^{m}\lambda_i\beta_i+\delta\sum_{i=1}^{m}\lambda_i\\ &\leq C_U+\sum_{i=1}^{m}\max_{S_i\in[S_L,S_U]}g_i(S_i)-\left(\sum_{i=1}^{m}\lambda_i\beta_i-\frac{1}{n}\sum_{j=1}^{n}\min_i\lambda_i a_{ij}\right)+\delta\sum_{i=1}^{m}\lambda_i\\ &\leq C_U+\sum_{i=1}^{m}\max_{S_i\in[S_L,S_U]}g_i(S_i)-\min_{\lambda'\geq 0,e^\top\lambda'=1}\left(\sum_{i=1}^{m}\lambda'_i\beta_i-\frac{1}{n}\sum_{j=1}^{n}\min_i\lambda'_i a_{ij}\right)\sum_{i=1}^{m}\lambda_i+\delta\sum_{i=1}^{m}\lambda_i\\ &\to C_U+\sum_{i=1}^{m}\max_{S_i\in[S_L,S_U]}g_i(S_i)+(\delta-\Delta)\sum_{i=1}^{m}\lambda_i. \qquad (4\text{–}32)\end{aligned}$$

with probability 1 as $n\to\infty$ by Romeijn and Piersma [123]. Therefore, for any λ, (4–31) and (4–32) imply that

$$C_L+\sum_{i=1}^{m}\min_{S_i\in[S_L,S_U]}g_i(S_i)\leq C_U+\sum_{i=1}^{m}\max_{S_i\in[S_L,S_U]}g_i(S_i)+(\delta-\Delta)\sum_{i=1}^{m}\lambda_i.$$

Therefore, the function $\Phi_n(\lambda, \delta)$ obtains its maximum on the compact set $\Lambda = \{\lambda \geq 0, \sum_{i=1}^{m} \lambda_i \leq \Gamma\}$, where

$$\Gamma = \frac{C_U - C_L + \sum_{i=1}^{m} (\max_{S_i \in [S_L, S_U]} g_i(S_i) - \min_{S_i' \in [S_L, S_U]} g_i(S_i'))}{\Delta - \delta},$$

with probability 1 as $n \to \infty$. Since $\Phi_n(\lambda, \delta_n) \leq \Phi_n(\lambda, \delta)$, this function also achieves its maximum on the same compact set. Therefore, we have that

$$\Phi_n(\lambda, \delta_n) \leq \Phi_n(\lambda, 0) + \Gamma \delta_n \tag{4–33}$$

with probability 1 as $n \to \infty$. This implies that,

$$\begin{aligned} f_n(x^*) \leq f_n(x^*(\delta_n)) = \max_{\lambda \geq 0} \Phi_n(\lambda, \delta_n) &\leq \max_{\lambda \geq 0} \Phi_n(\lambda, 0) + \Gamma \delta_n \\ &= f_n(x^*) + \Gamma \delta_n \to f_n(x^*) \text{ as } n \to \infty. \end{aligned}$$

□

We now can prove both feasibility and optimality for an alternative greedy procedure.

Theorem 4.3.9. *If the functions g_i, $i = 1, \ldots, m$, are convex, then the solution returned by the greedy heuristic, $x^G(\delta_n)$, where we use the optimal set of dual multipliers from NAR(n, δ_n), $(\lambda^*(\delta_n), \gamma^*(\delta_n))$, is asymptotically feasible and optimal to NA(n).*

Proof. Lemma 4.2.1 states that only split jobs in $x^*(\delta_n)$ are reassigned by the greedy procedure to obtain $x^G(\delta_n)$. Since the amount of additional capacity in the problem NA(n) over the amount in NAR(n, δ_n) goes to infinity (i.e., $\lim_{n \to \infty} n\delta_n$) and the number of reassignments is independent of n (see Lemma 4.1.2), $x^G(\delta_n)$ is feasible to NA(n) with probability 1 as n goes to infinity. This implies that $x^G(\delta_n)$ satisfies Condition 4.3.1. By Theorem 4.3.3 and Lemma 4.3.8, we have that, with probability 1 as $n \to \infty$,

$$\lim_{n \to \infty} f_n(x^G(\delta_n)) = \lim_{n \to \infty} f_n(x^*(\delta_n)) = \lim_{n \to \infty} f_n(x^*).$$

Therefore, $x^G(\delta_n)$ is asymptotically optimal to NA(n). □

4.4 Chapter Summary and Future Research Directions

In this chapter, we have discussed results about the optimal solution to the relaxation of the NL-GAP and a greedy procedure for the NL-GAP. We developed a set of dual multipliers associated with some optimal solution to the relaxation and showed that the greedy procedure using these multipliers only reassigns a small number of customers from the facility to which they are assigned in the optimal solution to the relaxation. Further, we discussed the performance of the greedy procedure for an appropriately defined stochastic model of the parameters. We show that the greedy procedure is asymptotically optimal and feasible for any set of continuous functions when the requirements of the NL-GAP are facility-independent. For the important special case of the NL-GAP with convex functions, we show that there exists a set of dual multipliers that produces an asymptotically feasible and optimal greedy procedure. We have shown that for convex functions, these dual multipliers can be obtained by solving a Lagrangian dual problem. Therefore, for the NL-GAP with convex functions we have provided a solution method that requires solving the Lagrangian dual problem and applying the greedy heuristic with the optimal Lagrangian multipliers that is asymptotically feasible and optimal to the NL-GAP under a very general stochastic model.

We are currently investigating whether the results of Section 4.3.2 can be generalized to other classes of nonlinear functions. We are mainly focusing on the case when the functions g_i, $i = 1, \ldots, m$, are non-decreasing since in all the applications that serve as motivation for studying the NL-GAP, the functions are non-decreasing. It also may be interesting to examine the properties of the greedy heuristic if we were to use the optimal solution to the Lagrangian dual problem studied in Section 4.1.2. This is due to the fact that we may effectively determine the optimal solution to the Lagrangian dual problem, since the subproblems for a fixed dual solution can be determined efficiently.

CHAPTER 5
SIMPLEX-INSPIRED ALGORITHMS FOR SOLVING A CLASS OF CONVEX PROGRAMMING PROBLEMS

In this chapter, we will study the following class of convex programming problems:

$$\text{minimize } c^\top x + g(Sx)$$

$$\text{subject to} \qquad \text{(CP)}$$

$$\begin{aligned} Ax &= b \\ x &\geq 0 \end{aligned}$$

where $x \in \mathbb{R}^n$ is a vector of decision variables, $c \in \mathbb{R}^n$, $S \in \mathbb{R}^{K \times n}$, $A \in \mathbb{R}^{m \times n}$, and $b \in \mathbb{R}^m$ are the problem data, and $g : \mathbb{R}^K \to \mathbb{R}$ is a convex function with continuous partial derivatives (however, we will relax this assumption in Section 5.5.1 to allow for non-differentiable convex functions). In addition, we assume that the feasible region of CP is nonempty and bounded (we will relax the bounded assumption in Section 5.5.2), so that an optimal solution to CP exists. For convenience, we denote the feasible region of CP by P. We will develop two solution methods that solve CP through solving a sequence of either 1-dimensional or K-dimensional convex programming problems. The latter is therefore particularly attractive if $K \ll n$. The methods that we develop are partially inspired by the simplex methods for linear and convex programming, where we employ the fact that we can show that CP has an optimal solution that lies on a face of P of dimension K. Note that this generalizes the well-known result that a linear programming problem has an extreme point optimal solution (provided an optimal solution exists). Our proposed algorithms generate a sequence of solutions on faces of P of dimension no more than K.

Zangwill [160] proposed a so-called convex simplex method (CSM) that can, in principle, be applied to CP. However, that algorithm is only guaranteed to asymptotically converge in value to the optimal solution value. Of course, any other algorithm developed

for constrained nonlinear or convex programming methods could be use to solve CP as well. These methods include gradient projection, descent direction or penalty methods (see Bazaraa et al. [13] or Bertsekas [19]) as well as interior point methods for nonlinear programming (see Forsgen et al. [56] for a review). However, since these methods are for general problems, they will not take advantage of the special structure of CP and some, like the CSM, only guarantee asymptotic convergence. It is our goal to develop an algorithm that achieves convergence in a finite number of iterations by fully employing the structure of our problem.

Our motivation for studying CP stems from its many practical applications, in particular in supply chain optimization, as either a problem of independent interest or a subproblem in an algorithm to solve a larger problem. The relaxation of the NL-GAP with convex functions g_i can be formulated as a CP. In particular, we define the convex function g in CP as

$$g(Sx) = \sum_{i=1}^{m} g_i \left(\sum_{j=1}^{n} s_{ij} x_{ij} \right).$$

As another example, consider a market selection problem where a company can choose to serve the demand for a product in each of n markets. The objective function parameter c_i then corresponds to the negative of the revenue and s_i to the demand rate associated with market i, while x_i indicates the fraction of market i that the company chooses to serve. The function $g(s^\top x)$ would represent the cost of acquiring (or producing) $s^\top x$ units of demand. Under certain cost and capacity structures, the function g is convex (see, for example, Freling et al. [57]). This problem then belongs to the class CP with $K = 1$, where the objective function of the problem is $c^\top x + g(s^\top x)$, i.e., we are minimizing total net cost. This model can be extended to a setting where K products are offered and the demand rate in market i for product k is s_{ik}. Sharkey et al. [140] discuss this application and extensions thereof in more detail.

The remainder of this chapter is organized as follows. In Section 5.1, we formally prove that there exists an optimal solution to CP on a face of P of dimension K and

extend the concept of a basic feasible solution to our problem class CP. In Section 5.2.1, we describe an algorithm with simple pivots (i.e, pivots based on a single nonbasic variable) to solve CP. In Section 5.2.2, we prove the correctness of this algorithm. In particular, it is shown that for $K = 1$, this algorithm terminates in a finite number of iterations. For general K, it is shown that the algorithm asymptotically converges in cost to the optimal solution value. In Section 5.3, we use an interesting insight into the algorithm with simple pivots for $K = 1$ to develop an algorithm with generalized pivots (i.e., pivots based on a collection of nonbasic variables). It is shown that the algorithm with generalized pivots solves CP for general K in a finite number of iterations. We perform a computational study of the algorithms in Section 5.4, and discuss some extensions of our algorithms and results in Section 5.5.

5.1 Mathematical Preliminaries

The algorithms that we will develop take advantage of the special structure of the objective function of CP. The following result provides the foundation for our algorithms.

Theorem 5.1.1. *There exists an optimal solution to CP that lies on a face of P of dimension K.*

Proof. Theorem 3.0.1 says that there is an optimal solution, x^* to CP that is also an extreme point of the linear program:

$$\text{minimize } c^\top x$$

$$\text{subject to} \qquad \text{(LCP)}$$

$$\begin{aligned} Sx &= Sx^* \\ Ax &= b \\ x &\geq 0. \end{aligned}$$

Since x^* is an extreme point of LCP, at least $n - m - K$ nonnegativity constraints are binding at x^* or, equivalently, at most $m + K$ elements of x^* are strictly positive. This implies that x^* lies on a face of P of dimension K. □

This result motivates our algorithms, which will restrict attention to solutions to CP on faces of P of dimension K. Clearly, if the nonlinear function g were absent the problems CP and LCP would be identical and we could apply the standard simplex method for linear programming to CP, which would move between basic feasible solutions along edges of the feasible region. In order to develop our algorithm, we need to establish a characterization of all solutions on faces of P of dimension K. To this end, consider the following reformulation of CP:

$$\text{minimize } c^\top x + g(y)$$

$$\text{subject to} \qquad \text{(ACP)}$$

$$\begin{aligned} Sx - y &= 0 \\ Ax &= b \\ x &\geq 0. \end{aligned}$$

Note that a point $x \in P$ is on a face of dimension K if and only if no more than $m + K$ of its elements are strictly positive. We will use this property to characterize such solutions in terms of the feasible region of ACP, which we will denote by Q. Formally, let (x^B, y^B) denote a subvector of (x, y) with exactly $m + K$ elements (where x^B or y^B may be an empty vector)[1] . Similarly, let (x^N, y^N) denote the remaining decision variables.

[1] In principle, we should write, for example, $\begin{pmatrix} x \\ y \end{pmatrix}$ or, equivalently, $\left(x^\top, y^\top\right)^\top$. However, for ease of exposition we will use the simpler notation (x, y), where the correct interpretation should be clear from the context.

Furthermore, let B and N denote the submatrices of

$$\begin{pmatrix} A & 0 \\ S & -I \end{pmatrix} \tag{5–1}$$

consisting of the columns corresponding to the variables in (x^B, y^B) and (x^N, y^N), respectively. We will assume that the matrix B is invertible, and, therefore, the variables (x^B, y^B) can be expressed in terms of the variables (x^N, y^N). Put differently, this means that any choice of values for (x^N, y^N) *uniquely* determines the values of the remaining decision variables (x^B, y^B). For linear programs with nonnegative variables, this observation is used to define basic solutions as ones that correspond to setting $(x^N, y^N) = (0, 0)$. In our setting, we will define any solution (x, y) that can be found by setting $(x^N, y^N) = (0, \bar{y}^N)$ for *some* $\bar{y}^N$ to be a *generalized basic solution* to ACP. If, in addition, the x-components of this generalized basic solution are all nonnegative, we will say that this solution is a *generalized basic feasible solution* to ACP. Following terminology used in the simplex method for linear programming we will, in the remainder, refer to the decision variables (x^B, y^B) as *basic variables* and to the decision variables (x^N, y^N) as *nonbasic variables*.

It is perhaps interesting to note that our notion of generalized basic feasible solutions bears a strong resemblance to the way simple upper bounds on decision variables are commonly dealt with in the simplex method for linear programming. In particular, if g were linear and we would impose bounds of the form $L \leq y \leq U$, the traditional basic solutions would be obtained by choosing all elements of $\bar{y}^N$ from $\{L, U\}$. In contrast, in our problem the nonlinearity of g requires us to allow $\bar{y}^N$ to, in principle, take on any value.

In the remainder of this chapter we will propose two new algorithms that solve ACP and, equivalently, solve CP. These algorithms proceed in an analogous fashion to the simplex method for linear programming as well as Zangwill's CSM. In particular, a

sequence of *adjacent* generalized basic feasible solutions whose corresponding sequence of function values is nonincreasing is generated. Two generalized basic solutions are said to be adjacent if one can be reached from the other by an appropriate *pivot operation*. However, because of the nonlinearity of our problem we more carefully define (and generalize) the concept of a pivot operation. Our first algorithm employs so-called *simple pivots*, i.e., pivots based on a single nonbasic variable. Next, we will propose an algorithm that employs a particular type of *generalized pivots*, i.e., pivots based on multiple nonbasic variables.

5.2 A Simplex Algorithm with Simple Pivots

5.2.1 Intuitive Development and Description of the Algorithm

It is standard in simplex-type algorithms to express the basic variables in terms of the nonbasic variables:

$$\begin{pmatrix} x^B \\ y^B \end{pmatrix} = B^{-1} \begin{pmatrix} b \\ 0 \end{pmatrix} - B^{-1} N \begin{pmatrix} x^N \\ y^N \end{pmatrix}. \tag{5–2}$$

Note that each column of $-B^{-1}N$ can be used to construct a search direction, each of which corresponds to a particular nonbasic variable. In particular, this search direction is obtained by adding zero elements corresponding to all nonbasic variables except for the one associated with the column, which is set to one. Let $d = (d^x, d^y)$ denote a search direction corresponding to a particular nonbasic variable (where d^x and d^y denote the components corresponding to x and y, respectively). Note that, for nonbasic variables in y, we will explicitly consider two distinct search directions, associated with increasing and decreasing that variable. We then consider a one-dimensional convex optimization problem of the following form:

$$\text{minimize}_{\lambda \in \Lambda} \; c^\top (x + \lambda d^x) + g\,(y + \lambda d^y) \tag{5–3}$$

where λ denotes the quantity by which the associated nonbasic variable is changed and the feasible range of values for λ is implied by the nonnegativity constraints: $\Lambda = \{\lambda \in \mathbb{R} :$

$x + \lambda d^x \geq 0\}$. It is easy to see that the lower limit of Λ is 0 and, due to the boundedness of P, the interval Λ is bounded. We say that the variable is *eligible* to enter the basis if

$$c^\top d^x + \nabla g(y)^\top d^y < 0. \tag{5–4}$$

Note that the eligibility condition simply states that the derivative of (5–3) as a function of λ is negative at the point $\lambda = 0$. Now consider the optimization problem (5–3) corresponding to an eligible nonbasic variable (and direction of change). (Note that here and in the remainder of the development of our algorithm we do not limit ourselves to a specific pivot rule.) If the solution corresponding to the nonzero boundary point of Λ is optimal, then further improvement of the objective function in the direction d is prevented by the nonnegativity constraints. We then pivot the selected nonbasic variable into the basis and replace it by (one of) the basic variable(s) in x^B that reached the value zero. Alternatively, we may have an interior optimal solution to problem (5–3). We then distinguish between the case where the selected eligible nonbasic variable is in x^N or in y^N. In the former case, we pivot the selected nonbasic variable into the basis, replacing a variable in y^B (which is guaranteed to exist, since otherwise the interior point would not be optimal). In the latter case, we could leave the basis unchanged and merely change the value of the selected nonbasic variable in y^N. However, we may alternatively replace a variable in y^B by pivoting the selected nonbasic variable in y^N into the basis, provided such a pivot exists. We will now formally state our general algorithm with simple pivots.

Simplex algorithm with simple pivots

Step 0. Determine a generalized basic feasible solution to ACP with basis (x^B, y^B) and write the set of basic variables in terms of the set of nonbasic ones.

Step 1. Determine an eligible nonbasic variable in (x^N, y^N) if one exists; if not, terminate the algorithm with the current solution.

Step 2. Solve the problem (5–3). Perform a pivot on the selected eligible nonbasic variable and write the new set of basic variables in terms of the new set of nonbasic variables. Return to Step 1.

Clearly, to completely specify the algorithm it is necessary to provide a *pivot rule*, in accordance with the restrictions depending on the solution of (5–3), that determines (i) which eligible variable will be pivoted on in Step 1 and (ii) which basic variable will be pivoted out of the basis (if any) in Step 2.

We will now briefly compare our algorithm to the CSM of Zangwill [160]. The CSM minimizes a convex objective function of $\bar{n}$ variables over a polyhedron defined by $\bar{m}$ equality and $\bar{n}$ nonnegativity constraints. Note that, without any additional structure in the objective function besides convexity, all $\bar{n}$ variables could potentially be nonzero in the optimal solution. The CSM method, similarly to our algorithm, decomposes the vector of decision variables into a collection of $\bar{m}$ basic and $\bar{n} - \bar{m}$ nonbasic variables. In each iteration, a one-dimensional optimization problem is defined based on pivoting on an eligible nonbasic variable. However, this eligible nonbasic variable only enters the basis if the nonnegativity constraint on one of the basic variables is binding in the one-dimensional optimization problem; the basic variable that reaches the value zero then becomes a nonbasic variable. It is clear that the CSM could be applied to CP or to a modification of ACP in which the variables y are replaced by nonnegative variables representing their positive and negative parts. Alternatively, a slight modification of the CSM could be applied directly to ACP, where the unconstrained variables y could be either basic or nonbasic but, perhaps interestingly, would never leave the basis if they ever enter it. It is easy to see that, in general, the sequence of iteration points generated by the CSM could be very different from the sequence of iteration points generated by our algorithm, particularly if $K < n$. We will next present an example that illustrates the differences between the CSM and our algorithm.

Consider the following class of problems in $\mathbb{R}^4$ with $K = 1$:

$$\text{minimize } -x_1 + \frac{1}{2\alpha}(x_1 - x_2)^2$$

subject to (E$_1$)

$$\begin{aligned} x_1 + x_3 &= 1 \\ x_2 + x_4 &= 1 \\ x_1, x_2, x_3, x_4 &\geq 0 \end{aligned}$$

where $1/\alpha$ is some positive integer. Now apply the CSM from the starting point $(0, 0, 1, 1)$. In this starting point, variables x_1 and x_2 are nonbasic and x_3 and x_4 are basic. x_1 is an eligible nonbasic variable, the associated direction is $(1, 0, -1, 0)$, and the corresponding one-dimensional optimization problem reads

$$\text{minimize}_{0 \leq \lambda \leq 1} -\lambda + \frac{1}{2\alpha}\lambda^2$$

whose optimal solution is given by $\lambda^* = \alpha$. The next iterate is $(\alpha, 0, 1 - \alpha, 1)$ with the basis unchanged. Then x_2 is an eligible nonbasic variable, the associated direction is $(0, 1, 0, -1)$, and the corresponding one-dimensional optimization problem reads

$$\text{minimize}_{0 \leq \lambda \leq 1} -\alpha + \frac{1}{2\alpha}(\alpha - \lambda)^2$$

whose optimal solution is given by $\lambda^* = \alpha$. The next iterate is then $(\alpha, \alpha, 1 - \alpha, 1 - \alpha)$ with the basis again unchanged. It is easy to see that the CSM continues by alternating x_1 and x_2 as eligible nonbasic variables, so that the algorithm alternates between solutions of the form $(k\alpha, k\alpha, 1 - k\alpha, 1 - k\alpha)$ and $((k + 1)\alpha, k\alpha, 1 - (k + 1)\alpha, 1 - k\alpha)$ until the optimal solution $(1, 1, 0, 0)$ is reached after $2/\alpha$ iterations.

In contrast, our algorithm proceeds by first reformulating the problem as:

$$\text{minimize } -x_1 + \frac{1}{2\alpha}y_1^2$$

subject to

$$\begin{aligned} x_1 + x_3 &= 1 \\ x_2 + x_4 &= 1 \\ x_1 - x_2 - y_1 &= 0 \\ x_1, x_2, x_3, x_4 &\geq 0. \end{aligned}$$

It is easy to show that the CSM applied to this reformulation leads to the same sequence of iterates as when it is applied to E_1 from the starting solution $(0, 0, 1, 1, 0)$ where x_3, x_4, and y_1 are basic. We will examine the behavior of our algorithm from this starting solution. The first iteration is then identical to the first iteration of the CSM and leads to the solution $(\alpha, 0, 1 - \alpha, 1, \alpha)$, but we replace y_1 in the basis by x_1. In the next iteration, x_2 is an eligible nonbasic variable, the associated direction is $(1, 1, -1, -1, 0)$, and the corresponding one-dimensional optimization problem reads

$$\text{minimize}_{0 \leq \lambda \leq 1-\alpha} \; -(\alpha + \lambda) + \frac{1}{2\alpha}\alpha^2$$

whose optimal solution is given by $\lambda^* = 1 - \alpha$. The next iterate is $(1, 1 - \alpha, 0, \alpha, \alpha)$ and the variable x_3 in the basis is replaced by x_2. Then y_1 is eligible to enter the basis with associated direction $(0, 1, 0, -1, -1)$ and it is easy to see that the next iterate is the optimal solution $(1, 1, 0, 0, 0)$, reached after 3 iterations. Figure 5-1 illustrates the paths that the CSM and our algorithm with simple pivots take in solving E_1.

The above example does not only show that the sequence of points visited by our algorithm may be very different from the sequence of points visited by the CSM, but also that the CSM may visit many more points than our algorithm. In particular, for the class of problems above, the ratio between the number of iterations required by the CSM and the number of iterations required by our algorithm can be arbitrarily large, depending on the value of the constant α. In Section 5.2.2.1 we will explain this behavior by showing

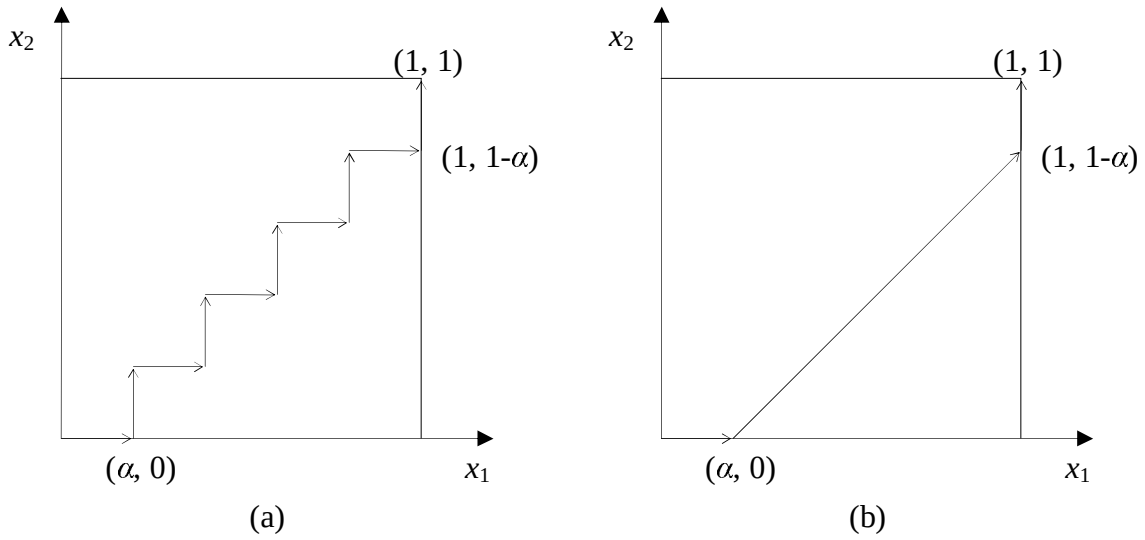

Figure 5-1. An illustration of the paths in (x_1, x_2)-space followed by (a) the CSM and (b) our algorithm when solving E_1.

that, when $K = 1$, the maximum number of iterations required by our algorithm depends only on the structure of P, while the example above clearly demonstrates that the number of iterations required by the CSM may depend on the nonlinear component g of the objective function. Before we return to this issue, however, we will first prove the correctness of our algorithm.

5.2.2 Proof of Correctness

In this section, we will prove that our algorithm with simple pivots is correct in the sense that it produces a sequence of points whose objective function values converge to the optimal solution value of CP. To this end, we will first show that, if no eligible nonbasic variable exists in Step 1 of the algorithm, the current solution is an optimal solution to CP. This result in fact provides optimality conditions for a generalized basic feasible solution that generalize the set of optimality conditions for the simplex method for linear programming and are similar in spirit to the optimality conditions used in the proof of correctness of the CSM in Zangwill [160].

Theorem 5.2.1. *Let $(\bar{x}, \bar{y})$ be a generalized basic feasible solution to ACP with basic variables $(\bar{x}^B, \bar{y}^B)$ and nonbasic variables $(\bar{x}^N, \bar{y}^N)$. If there does not exist an eligible nonbasic variable then $(\bar{x}, \bar{y})$ is an optimal solution to ACP.*

Proof. Let $(\bar{x}, \bar{y})$ be a particular generalized basic feasible solution to ACP along with a basic/nonbasic representation of this solution given by $(\bar{x}^B, \bar{y}^B)$ and $(\bar{x}^N, \bar{y}^N)$. If we denote corresponding vectors of decision variables by $(x(B), y(B))$ and $(x(N), y(N))$, we can rewrite ACP as follows:

$$\text{minimize } c^\top x + g(y)$$

$$\text{subject to} \qquad \text{(ACP}')$$

$$\begin{pmatrix} x(B) \\ y(B) \end{pmatrix} + B^{-1}N \begin{pmatrix} x(N) \\ y(N) \end{pmatrix} = \begin{pmatrix} \bar{x}^B \\ \bar{y}^B \end{pmatrix}$$

$$x(B), x(N) \geq 0.$$

For convenience, let $B_{\bar{x}}$, $B_{\bar{y}}$, $N_{\bar{x}}$, and $N_{\bar{y}}$ denote the index sets of the basic and nonbasic variables in $\bar{x}$ and $\bar{y}$, respectively. Moreover, let j_i denote the index of x_i in $(x(B), y(B))$ if $i \in B_{\bar{x}}$ and in $(x(N), y(N))$ if $i \in N_{\bar{x}}$; similarly, let j_{n+k} denote the index of y_k in $(x(B), y(B))$ if $k \in B_{\bar{y}}$ and in $(x(N), y(N))$ if $k \in N_{\bar{y}}$. Finally, let N_j denote the j^{th} column of the matrix N. If $v \in \mathbb{R}^{m+K}$ is the vector of dual multipliers for the equality constraints in ACP$'$, the necessary and sufficient Karush-Kuhn-Tucker (KKT) optimality conditions for a feasible solution (x, y) to ACP$'$ can be written as

$$c_i + v_{j_i} \geq 0 \qquad \text{for } i \in B_{\bar{x}} \tag{5–5}$$

$$c_i + v^\top B^{-1} N_{j_i} \geq 0 \qquad \text{for } i \in N_{\bar{x}} \tag{5–6}$$

$$\nabla_k g(\bar{y}) + v_{j_{n+k}} = 0 \qquad \text{for } k \in B_{\bar{y}} \tag{5–7}$$

$$\nabla_k g(\bar{y}) + v^\top B^{-1} N_{j_{n+k}} = 0 \qquad \text{for } k \in N_{\bar{y}}. \tag{5–8}$$

Now if we let $v_{j_i} = -c_i$ for $i \in B_{\bar{x}}$ and $v_{j_{n+k}} = -\nabla_k g(\bar{y})$ for $k \in B_{\bar{y}}$ it is easy to see that conditions (5–5) and (5–7) are satisfied. In addition, we let $v_{j_i} = 0$ for $i \in N_x$ and $v_{j_{n+k}} = 0$ for $k \in N_{\bar{y}}$. For $i \in N_{\bar{x}}$, we then have that the search direction corresponding to j_i, $(d^{\bar{x}}, d^{\bar{y}})$, has that

$$c^\top d^{\bar{x}} + \nabla g(\bar{y})^\top d^{\bar{y}} = c_i + v^\top B^{-1} N_{j_i}, \tag{5–9}$$

and, for $k \in N_{\bar{y}}$, we have that the search direction corresponding to j_k, $(d^{\bar{x}}, d^{\bar{y}})$, has that

$$c^\top d^{\bar{x}} + \nabla g(\bar{y})^\top d^{\bar{y}} = \nabla_k g(\bar{y}) + v^\top B^{-1} N_{j_k}. \tag{5–10}$$

It then follows by (5–4), (5–9), (5–10), and the fact that there does not exist an eligible nonbasic variable that conditions (5–6) and (5–8) are satisfied, which yields the desired result. □

5.2.2.1 The Case $K = 1$

In the special case of $K = 1$, it turns out that we can guarantee that the algorithm terminates in a finite number of iterations by employing an anti-cycling pivot rule. Due to the nonlinearity of ACP we have to be careful in defining what makes a pivot rule an anti-cycling one. In our approach, a generalized basic feasible solution is not only characterized by the set of basic variables but also by the value of the variable y when that variable is nonbasic. This will not be a problem if a basis that does not contain y is repeated, as long as that nonbasic variable has a different value. We therefore define an anti-cycling pivot rule to be any rule that prevents the same basis from reappearing *while the variable y is nonbasic and fixed*. Clearly, while the variable y is fixed we are essentially solving a linear programming problem, so that any anti-cycling rule developed for such problems will apply here. The next theorem shows that, under an anti-cycling pivot rule, in a finite number of iterations we find an optimal solution to ACP (and hence CP).

Theorem 5.2.2. *If we use an anti-cycling pivot rule and $K = 1$, then our algorithm with simple pivots terminates in a finite number of iterations with an optimal solution.*

Proof. First, we will show that our algorithm will only perform a finite number of pivot operations. The fact that we are using an anti-cycling rule ensures that the number of consecutive degenerate pivots is finite. Moreover, each nondegenerate pivot operation can be uniquely placed into one of three classes: (i) the eligible nonbasic variable is y, (ii) the eligible variable is in x^N and y is basic, and (iii) the eligible variable is in x^N and y is nonbasic. If we perform a pivot operation where y is basic or a pivot operation where y is the eligible nonbasic variable, we are optimizing the objective function over an edge of P. After optimizing the objective function over an edge of P and performing a nondegenerate pivot, we cannot return to that edge since a nondegenerate pivot strictly decreases the objective function of the current solution. Therefore, the total number of pivot operations from the first two classes is finite (in particular, the number is bounded by the number of edges of P). Furthermore, there can only be a finite number of consecutive nondegenerate pivots from the third class, since when y is fixed we are essentially performing standard linear programming pivots. This implies that the number of nondegenerate, and hence the total number of, pivots performed by our algorithm is finite.

Now since we have established that our algorithm will only generate a finite number of solutions and basic representations of these solutions, one of the solutions and its basic representation must satisfy the conditions of Theorem 5.2.1. This implies that the algorithm terminates with an optimal solution. □

As a byproduct of the proof of Theorem 5.2.2 we obtain that the number of iterations that our algorithm requires for $K = 1$ is bounded by a number that only depends on the structure of the polyhedron P and not on the nonlinear component $g(Sx)$ of the objective function of CP. This result highlights an important advantage of our method over the CSM when $K = 1$.

Corollary 5.2.3. *If we use an anti-cycling pivot rule and $K = 1$, then the maximum number of distinct solutions generated by our algorithm with simple pivots only depends on the structure of P and the particular anti-cycling pivot rule used.*

Proof. If we use any anti-cycling pivot rule, the maximum number of distinct solutions generated where y remains nonbasic and fixed is no more than the number of extreme points of the intersection of P with the hyperplane $Sx = y$, which is bounded by the number of edges of P. Furthermore, since any value of y that we consider during the course of the algorithm is the optimal value over some edge of P, the total number of values of y that we consider is no larger than the number of edges of P. In other words, for each of no more than ρ values of y, we consider no more than ρ distinct solutions, which implies that our algorithm will generate no more than ρ^2 distinct solutions.

Recall that Bland's rule pivots on the eligible non-basic variable with the smallest index. Without loss of generality, we may choose y to be the variable with the smallest index among all decision variables. Now suppose that we use Bland's rule as the pivot rule. Then, whenever we visit an edge of P for the first time, one of two events may occur. If y is an eligible nonbasic variable, Bland's rule will select this variable and we optimize the objective function of ACP over the current edge; if y is ineligible, the current solution is optimal over the current edge. This means that the algorithm performs no more than a single nondegenerate pivot operation with y non-basic before either optimizing over an edge of P or verifying that the current solution is optimal over the edge it lies on. The algorithm therefore generates no more than 2ρ distinct solutions when implemented with Bland's rule. □

Now let us consider the following generalization of the class of problems (E_1):

$$\text{minimize } -x_1 + g(x_1 - x_2)$$

subject to (E_1')

$$\begin{aligned} x_1 + x_3 &= 1 \\ x_2 + x_4 &= 1 \\ x_1, x_2, x_3, x_4 &\geq 0 \end{aligned}$$

for some convex function g. It is not difficult to verify that $\rho = 4$ for this class of problems, so that our algorithm, with Bland's rule, will generate no more than 8 distinct solutions *regardless of the form of* g. We can again contrast this with the $2/\alpha$ solutions that are required by the CSM for the special case of $g(y) = \frac{1}{2\alpha}y^2$.

5.2.2.2 The General Case

We next study the convergence properties of our algorithm with simple pivots for general K. Unfortunately, for $K > 1$ we cannot ensure that our algorithm with simple pivots converges in general, let alone in a finite number of iterations. In this section, we will show that there exists a pivot rule that not only prevents cycling but also yields a convergence guarantee. In particular, we will construct a pivot rule for which we will prove that our algorithm with simple pivots satisfies two attractive convergence properties:

- any convergent subsequence of solutions generated by the algorithm converges to an optimal solution of ACP;
- the sequence of objective function values generated by the algorithm converges to the optimal solution value of ACP.

The pivot rule proceeds by (i) selecting the eligible nonbasic variable that corresponds to the steepest descent rule; and (ii) selecting the basic variable to leave the basis using a modification of the so-called lexicographic rule from the simplex method for linear programming. We will next discuss this pivot rule in more detail. The steepest descent rule selects the nonbasic variable for which the corresponding direction minimizes $c^\top d^x + \nabla g(y)^\top d^y$. We will show later that this rule ensures convergence of our algorithm in the sense as described above *if* the algorithm does not cycle. However, recall that even in the simplex method for linear programming the steepest descent selection rule may cause cycling, i.e., repeat a basis, unless the leaving variable is chosen carefully. Cycling is prevented if, when a set of basic variables is tied in the minimum-ratio test (see Section 3.2 of Bertsimas and Tsitsiklis [20]), i.e., more than one basic variable reach zero in the pivot, the basic variable among these whose row in the simplex tableau (after scaling

by the entry in that row corresponding to the column of the entering nonbasic variable) is lexicographically the smallest is selected as the leaving variable (see Section 3.4 of Bertsimas and Tsitsiklis [20]). Now note, however, that a straightforward implementation of this lexicographic pivot rule will not work for our algorithm, since we do not want to prevent repeating a basis as long as the corresponding vector y^N is distinct. In other words, we are only interested in making sure that we do not repeat a basis along with the same corresponding value of y^N. We therefore modify the lexicographic pivot rule to as follows: whenever we arrive at a basis after performing a nondegenerate pivot, we reindex the variables so that the $m + K$ basic variables appear before the nonbasic variables; we then apply the lexicographic pivot rule as described above. We will refer to this rule as the *modified lexicographic rule.*

Lemma 5.2.4. *If we use the modified lexicographic rule to select the leaving variable, our algorithm with simple pivots does not cycle.*

Proof. Let $(\bar{x}, \bar{y})$ be the generalized basic feasible solution that is obtained just after performing a nondegenerate pivot. Given the current basis, rearrange the tableau so that the first $m + K$ columns correspond to the columns from the basis, and therefore, each row of the tableau is lexicographically positive. Suppose that our algorithm next performs a number of degenerate pivots. We will show that the same sequence of pivots would occur if we would apply some pivot rule from the basic solution $(\bar{x}, \bar{y})$ on an associated linear program. The objective function of the linear program has coefficient c_i for x_i (for $i = 1, \ldots, n$) and coefficient $\nabla_k g(\bar{y})$ for y_k (for $k = 1, \ldots, K$). The variables and constraints of the linear program are virtually identical to ACP, except for the fact that, for each nonbasic variable y_k, we introduce variables y_k^+ and y_k^- and replace the variable y_k by $y_k^+ - y_k^-$ in the constraints and objective function. We will use the same pivot rule on this linear program as we do in the algorithm with simple pivots with one exception: if y_k^+ (or y_k^-) becomes a basic variable, our pivot rule disregards y_k^- (or y_k^+). Note that once a variable y_k^+ becomes basic in our algorithm, it will not leave the basis until we

perform a nondegenerate pivot (since only variables in x^B may be tied according to the minimum ratio rule). Now observe that a non-basic variable x_i has a negative reduced cost in the linear program if and only if that non-basic variable is eligible in ACP. Similarly, a non-basic variable y_k^- (or y_k^+) has a negative reduced cost in the linear program if and only if that variable is eligible in ACP by decreasing (or increasing) its value. This implies that the pivots performed in the algorithm with simple pivots are equivalent to the pivots performed on this linear program until a nondegenerate pivot occurs in both. Since we know that the simplex method with the lexicographic pivot rule from a starting basis where each row in the tableau is lexicographically positive does not cycle, our algorithm does not cycle either. This implies that we will never return to the same basis representing the solution $(\bar{x}, \bar{y})$. □

This lemma is important because it helps show that if the algorithm generates an infinite sequence of solutions, then none of the iteration points of the algorithm can be identical to a cluster point of the sequence of generated solutions. This in fact is made formal and used in proving the next result.

Lemma 5.2.5. *Consider our algorithm with simple pivots using the combination of steepest descent and modified lexicographic rule as the pivot rule. Let* $\{(x^k, y^k)\}_{k=1}^{\infty}$ *denote the sequence of solutions generated. Then, if* $\{k_i\}_{i=1}^{\infty}$ *corresponds to a subsequence of this sequence of solutions such that*

$$\lim_{i \to \infty} (x^{k_i}, y^{k_i}) = (x^*, y^*)$$

the solution (x^*, y^*) *is optimal to ACP.*

Proof. Let $(x(B)^k, y(B)^k)$ denote the *set* of basic variables corresponding to (x^k, y^k), and let $(d^{x,k}, d^{y,k})$ denote the search direction corresponding to the pivot from this solution. We will show the desired result in two steps. First, we will show that the sequence of directions is *gradient related* to the sequence of solutions, i.e., for any subsequence $\{(x^{k_i}, y^{k_i})\}_{i=1}^{\infty}$ that converges to a nonstationary point (i.e., a point that does not satisfy

the KKT conditions), the corresponding subsequence $\{(d^{x,k_i}, d^{y,k_i})\}_{i=1}^{\infty}$ is bounded and satisfies

$$\limsup_{i\to\infty} c^\top d^{x,k_i} + \nabla g(y^{k_i})^\top d^{y,k_i} < 0 \tag{5–11}$$

(see Bertsekas [18]). We then use this result, and the no-cycling result from Lemma 5.2.4, to show that any cluster point of the sequence of solutions generated by the algorithm is an optimal solution to ACP.

It is easy to see that, since there are only a finite number of candidate directions, the sequence of directions is bounded. Now consider a sequence $\{(x^{k_i}, y^{k_i})\}_{i=1}^{\infty}$ that converges to a nonstationary point $(\bar{x}, \bar{y})$. In order to show (5–11), we will break the sequence $\{(x^{k_i}, y^{k_i})\}_{i=1}^{\infty}$ into a finite set of infinite sequences as follows. We decompose the sequence $\{(x^{k_i}, y^{k_i})\}_{i=1}^{\infty}$ into distinct subsequences, each of which corresponds to solutions with an identical basis *and* an identical nonbasic variable and sign selected for entry into the basis in the next iteration (i.e., when an element of y is the nonbasic entering variable, we also specify whether that variable is increased or decreased). Since there are only a finite number of bases and a finite number of variables, there must be at least one choice of basis, nonbasic entering variable, and sign whose corresponding subsequence is infinite; denote the indices of this subsequence by $\{k'_i\}_{i=1}^{\infty}$. It is easy to see that (d^{x,k'_i}, d^{y,k'_i}) is constant for $i = 1, 2, \ldots$, say (d^x, d^y). Since $(\bar{x}, \bar{y})$ is a nonstationary point, there exists an eligible nonbasic variable, i.e., Theorem 5.2.1 says that there exists some $(d^{\bar{x}}, d^{\bar{y}})$ such that

$$c^\top d^{\bar{x}} + \nabla g(\bar{y})^\top d^{\bar{y}} < 0.$$

However, note that

$$c^\top d^x + \nabla g(y^{k'_i}) d^y \leq c^\top d^{\bar{x}} + \nabla g(y^{k'_i})^\top d^{\bar{y}}$$

for all $i = 1, 2, \ldots$ since we use the steepest descent pivot rule. Now recalling that $(d^{x,k'_i}, d^{y,k'_i}) = (d^x, d^y)$ for $i = 1, 2, \ldots$ and using that the partial derivatives of g are

continuous and $\lim_{i\to\infty} y^{k'_i} = \bar{y}$, we obtain

$$\begin{aligned}
\lim_{i\to\infty} c^\top d^{x,k'_i} + \nabla g(y^{k'_i})^\top d^{y,k'_i} &= \lim_{i\to\infty} c^\top d^x + \nabla g(y^{k'_i})^\top d^y \\
&\leq \lim_{i\to\infty} c^\top d^{\bar{x}} + \nabla g(y^{k'_i})^\top d^{\bar{y}} \\
&= c^\top d^{\bar{x}} + \nabla g(\bar{y})^\top d^{\bar{y}} \\
&< 0.
\end{aligned}$$

Clearly, this argument is independent of the particular choice of infinite subsequences into which the sequence $\{(x^{k_i}, y^{k_i})\}_{i=1}^{\infty}$ is decomposed. Since *any* convergent subsequence must contain a subsequence which corresponds to solutions with an identical basis *and* an identical nonbasic variable and sign selected for entry into the basis in the next iteration and there is only a finite number of such subsequences, we conclude that (5–11) holds and thus the sequence of directions generated by our algorithm is gradient related.

Since no iteration of the algorithm can move to a point with a worse objective function value, the sequence of generated solution values $(c^\top x^k + g(y^k))_{k=1}^{\infty}$ is non-increasing. By Lemma 5.2.4, the algorithm does not cycle, so that none of the solutions (x^k, y^k) can itself be a cluster point of the sequence of solutions $(x^k, y^k)_{k=1}^{\infty}$. Let $(x^{k'_i\prime}, y^{k'_i\prime})_{i=1}^{\infty}$ be the infinite sequence of solutions from which a nondegenerate pivot was performed, i.e., solutions in iterations k such that $(x^k, y^k) \neq (x^{k+1}, y^{k+1})$. By the fact that P is bounded, there exists a finite upper bound on the length of any line segment contained in P. Moreover, there are only a finite number of potential search directions, so that the sets Λ in (5–3) are uniformly bounded over *all* nondegenerate pivot operations. This result, together with the fact that the sequence of directions is gradient related, implies by Proposition 2.2.1 in Bertsekas [18] that any cluster point of the subsequence of solutions $(x^{k'_i\prime}, y^{k'_i\prime})_{i=1}^{\infty}$ generated by our algorithm is a stationary point of ACP. Since the objective function of ACP is convex, this implies that any cluster point of $(x^{k'_i\prime}, y^{k'_i\prime})_{i=1}^{\infty}$ is an optimal solution to ACP. The desired result follows by realizing that the set of cluster points of

$(x^k, y^k)_{k=1}^{\infty}$ is equivalent to the set of cluster points of $(x^{k'_i}, y^{k'_i})_{i=1}^{\infty}$ since the subsequence eliminates only duplicate solutions. □

We are now ready to prove our main convergence result.

Theorem 5.2.6. *Consider our algorithm with simple pivots using the combination of steepest descent and modified lexicographic rule as the pivot rule. Then the sequence of objective function values generated by the algorithm converges to the optimal solution value to ACP.*

Proof. If the algorithm terminates in a finite number of iterations, it is because a solution was found in which there exists no eligible nonbasic variable. Theorem 5.2.1 says that this solution is an optimal solution to ACP. Now suppose that the algorithm does not terminate finitely. Since P is bounded, the sequence of solutions generated by the algorithm must have a cluster point. Lemma 5.2.5 says that this cluster point is optimal to ACP. Since our algorithm generates a sequence of solutions that is non-increasing with respect to the objective function, this implies that the costs of the entire sequence of solutions converges to the optimal cost to ACP. □

This immediately leads to the following result:

Corollary 5.2.7. *Consider our algorithm with simple pivots using the combination of steepest descent and modified lexicographic rule as the pivot rule. If ACP has a unique optimal solution, then the sequence of solutions generated by the algorithm converges to the optimal solution ACP.*

The following example demonstrates that the algorithm with simple pivots may generate an infinite number of solutions, even for small values of K. We consider the problem:

$$\text{minimize } (x_1 - 1)^2 + (x_2 - 1)^2 + (x_1 - x_2)^2$$

subject to (E_2)

$$x_1 + x_3 = 1$$

$$\begin{aligned} x_2 + x_4 &= 1 \\ x_1, x_2, x_3, x_4 &\geq 0 \end{aligned}$$

which can be reformulated as:

$$\text{minimize } (y_1 - 1)^2 + (y_2 - 1)^2 + y_3^2$$

subject to (E_2')

$$\begin{aligned} x_1 - y_1 &= 0 \\ x_2 - y_2 &= 0 \\ x_1 - x_2 - y_3 &= 0 \\ x_1 + x_3 &= 1 \\ x_2 + x_4 &= 1 \\ x_1, x_2 &\geq 0. \end{aligned}$$

The optimal solution to E_2' is $(1, 1, 0, 0, 1, 1, 0)$ since its objective function value is 0 and the objective function is non-negative. Suppose that we are at a point in our algorithm where the solution is of the form

$$\left(1 - \frac{1}{2^k}, 1 - \frac{1}{2^{k-1}}, \frac{1}{2^k}, \frac{1}{2^{k-1}}, 1 - \frac{1}{2^k}, 1 - \frac{1}{2^{k-1}}, \frac{1}{2^k}\right) \tag{5–12}$$

where x_1, x_2, x_3, x_4, and y_3 are basic. The direction implied by raising y_2 is $d = (0, 1, 0, -1, 0, 1, -1)$, which is eligible. In order to determine the stepsize, we solve the problem

$$\text{minimize}_{0 \leq \lambda \leq \frac{1}{2^{k-1}}} \left(\lambda - \frac{1}{2^{k-1}}\right)^2 + \left(-\lambda + \frac{1}{2^k}\right)^2$$

whose optimal solution is $\lambda^* = \frac{3}{2^{k+1}}$.We then arrive at the solution

$$\left(1 - \frac{1}{2^k}, 1 - \frac{1}{2^{k+1}}, \frac{1}{2^k}, \frac{1}{2^{k+1}}, 1 - \frac{1}{2^k}, 1 - \frac{1}{2^{k+1}}, -\frac{1}{2^{k+1}}\right)$$

where x_1, x_2, x_3, x_4, and y_3 remain basic. The direction implied by raising y_1 is $d = (1, 0, -1, 0, 1, 0, 1)$, which is eligible. In order to determine our stepsize, we solve the problem

$$\text{minimize}_{0 \leq \lambda \leq \frac{1}{2^k}} \left(\lambda - \frac{1}{2^k}\right)^2 + \left(\lambda - \frac{1}{2^{k+1}}\right)^2.$$

The optimal solution to this problem is $\lambda^* = \frac{3}{2^{k+2}}$. We then arrive at a solution of the form (5–12) with k replaced by $k + 2$. Therefore, although the sequence of solutions generated by the algorithm approaches the optimal solution $(1, 1, 0, 0, 1, 1, 0)$, this solution is not reached in a finite number of iterations.

It is interesting that our algorithm with simple pivots is guaranteed to generate only a finite number of solutions if $K = 1$, but may in general require an infinite number of iterations. A close look at the convergence proof for $K = 1$ reveals that an important property of our algorithm for that case is that we are guaranteed to, after a finite number of iterations, optimize the objective over an edge of P, i.e., over a face of P of dimension $K = 1$. This suggests that if we, instead of optimizing over a 1-dimensional subset of P in each iteration, optimize over a subset of dimension K (or lower), we may be able to construct an algorithm that terminates in a finite number of iterations. In Section 5.3 we show that this is indeed the case.

5.3 A Simplex Algorithm with Generalized Pivots

A major advantage of our algorithm with simple pivots when applied to a problem with $K = 1$ is that it converges finitely. In this section, we will modify our algorithm so that this property can be extended to general K. In other words, we will develop an algorithm that determines an optimal solution to ACP in a finite number of iterations. Where our algorithm for ACP relies on the ability to efficiently solve subproblem (5–3), i.e., a one-dimensional convex optimization problem over a bounded interval, this modified algorithm will rely on the ability to efficiently solve a convex optimization problem over a polytope of dimension at most K. This algorithm can be expected to be particularly

effective when $K \ll n$, i.e., when the subproblems to be solved are of much lower dimension than the original problem ACP.

The simplex algorithm with generalized pivots that we will propose is similar in spirit to the algorithm with the simple pivots. In particular, we will continue to maintain a generalized basic feasible solution in the algorithm. In each iteration, we will select an eligible nonbasic variable but we will give preference to the nonbasic variables in y^N. If this selected nonbasic variable is in the subvector x^N, with associated search direction, we again solve the corresponding one-dimensional convex optimization subproblem (5–3) and perform a pivot operation. However, if the selected nonbasic variable is in y^N, we will actually consider changing *all* elements of y^N. Put differently, rather than viewing each element of y^N individually, we consider the vector y^N in its entirety. This subproblem then, in fact, optimizes the objective function of ACP over the face of the polytope P that is characterized by $x^N = 0$. More formally, this subproblem can be written as

$$\text{minimize}_{\lambda \in \Lambda} \;\; c^\top (x + D^x \lambda) + g\,(y + D^y \lambda)\,. \tag{5–13}$$

Here the elements of the vector λ are the coefficients of the corresponding search directions, i.e., λ represents the change in the nonbasic variables in y^N. Moreover, the rows of D^x and D^y corresponding to the basic variables x^B and y^B, respectively, are the partial rows of the matrix $B^{-1}N$ in (5–2) that characterize how the basic variables change when the vector of nonbasic variables in y^N changes. The remaining rows of D^x are equal to zero (since no element of x^N changes during the pivot) while the remaining rows of D^y are unit vectors. Finally, the set $\Lambda = \{\lambda \in \mathbb{R}^k : x + D^x \lambda \geq 0\}$ (where k is the dimension of y^N or λ) characterizes the set of feasible search direction multipliers. After solving this subproblem, we proceed in a similar fashion as in our algorithm with simple pivots. That is, if the optimal solution of the subproblem lies on the boundary of the set Λ we pivot one of the variables in y^N into the basis, removing one of the variables in x^B that reached the value 0. On the other hand, if the optimal solution of the subproblem lies in the interior of

Λ we have the option of either pivoting one of the variables in y^N into the basis, removing one of the variables in y^B, or simply retain the current basis (which is still valid). Similar to the presentation of the simplex algorithm with simple pivots, we present a general framework for the simplex algorithm with generalized pivots without explicitly limiting ourselves to a particular pivot rule.

Simplex algorithm with generalized pivots

Step 0. Determine a basic feasible solution to ACP with basis (x^B, y^B) and write the set of basic variables in terms of the set of nonbasic ones.

Step 1. Determine an eligible nonbasic variable in (x^N, y^N) if one exists; if not, terminate the algorithm with the current solution. If the selected nonbasic variable is in x^N, go to Step 2. Otherwise, go to Step 3.

Step 2. Solve the problem (5–3). Perform a pivot on the selected eligible nonbasic variable and write the new set of basic variables in terms of the new set of nonbasic variables. Return to Step 1.

Step 3. Solve the problem (5–13). Perform a pivot on the nonbasic subvector y^N and write the new set of basic variables in terms of the new set of nonbasic variables. Return to Step 1.

In order to completely specify the algorithm, it is necessary to provide a pivot rule used in Step 1 to determine the eligible nonbasic variable and in Step 2 to determine the basic variable leaving the basis. As the following theorem shows, this algorithm enjoys a similar convergence result for general K as the algorithm with simple pivots does for the case $K = 1$.

Theorem 5.3.1. *If we use an anti-cycling pivot rule that gives preference to nonbasic variables in y^N, then our algorithm with generalized pivots terminates in a finite number of iterations with an optimal solution.*

Proof. The proof of this result proceeds in a similar fashion to the proof of Theorem 5.2.2. First, we will show that our algorithm will only perform a finite number of pivot

operations. The fact that we are using an anti-cycling rule ensures that the number of consecutive degenerate pivots is finite. Moreover, each nondegenerate pivot operation can be uniquely placed into one of two classes: (i) the eligible nonbasic variable is in y^N and (ii) the eligible nonbasic variable is in x^N. When we perform a pivot operation from the first class, we optimize over some face of P. However, once we have optimized over a face of P and perform a nondegenerate pivot, the algorithm never generates a solution on that face again, since a nondegenerate pivot strictly decreases the objective function value of the current solution. Therefore, the total number of pivot operations from the first class is finite. Since the pivot selection rule always gives preference to variables in y^N, the algorithm can only perform a pivot operation from the second class when none of the the nonbasic variables in y^N are eligible. Therefore, when we perform a pivot from the second class the current solution must be the optimal solution on the face of smallest dimension that contains it. This implies that the maximum number of pivot operations from the second class is bounded by the maximum number of pivot operations from the first class, since we only perform a pivot operation of the second class starting from a solution that is optimal over a face of P. Therefore, the total number of nondegenerate pivot operations is finite. The final conclusion now follows in the same manner as the last paragraph of the proof of Theorem 5.2.2. □

Theorem 5.2.2 proves that our algorithm with simple pivots when applied to problems with $K = 1$ terminates in a finite number of iterations as long as we use an anti-cycling rule. To establish an analogous result for general K and for our algorithm with generalized pivots in Theorem 5.3.1, we need to, in addition, always give preference to y^N in order to guarantee that the algorithm with generalized pivots terminates in a finite number of iterations. The reason for this difference again highlights the special properties of the algorithm for $K = 1$. In the case of $K = 1$, we optimize over an edge of P in any iteration in which either y is selected as the eligible nonbasic variable or y is basic. However, in our algorithm with generalized pivots we only optimize over a face of P in an iteration

in which an element of y^N is selected as the eligible nonbasic variable. Therefore, if we do not give preference to y^N, when $K > 1$, the algorithm could take an infinite number of iterations to find the optimal solution over a particular face of P characterized by the current basis.

5.4 Computational Experiments

The methods that we have developed to solve CP are partially inspired by the simplex method for linear programming. Although the simplex method for linear programming may generate an exponential number of solutions, it is a very effective algorithm for linear programming due to the small number of iterations that it tends to require in practice. In addition, the simplex method has theoretically been shown to enjoy an attractive average case behavior (see Borgwardt [23] and Smale [146]). The goal of our computational tests in this section is to evaluate whether the algorithms developed in this chapter inherit the efficiency in practice of the simplex method for linear programming. In this study, we will focus on examining the average number of solutions that the algorithms generate in solving three different classes of problems. In particular, the problems that we will consider are of the form

$$\text{minimize } c^\top x + g(Sx)$$

subject to (TP)

$$\begin{aligned} Ax + z &= b \\ x &\leq 1 \\ x, z &\geq 0. \end{aligned}$$

We have implemented a slight modification of our algorithms to solve the reformulation ATP, which is obtained from TP in the same way as ACP is obtained from CP. In particular, we have chosen to handle the upper bounds implicitly (as in the simplex method for linear programming). We use Bland's rule as the pivot selection rule, where the variables in y are viewed as having smaller indices than the variables in x and z. We

construct an initial generalized basic feasible solution by choosing the vectors y and z as the initial basis. We generate the objective coefficients c_i from the uniform distribution on $\{-99, \ldots, 100\}$ and the constraint coefficients a_{ij} from the uniform distribution on $\{1, \ldots, 100\}$. The right-hand side of the constraints were generated uniformly from $\{1, \ldots, 50n\}$. (Note that we generated the problem parameters from discrete distributions to allow for the generation of problem instances with degenerate bases.) The distribution from which the objective coefficients s_{ik} were generated depends on the form of the convex function g. We focused much of our attention in the computational study on the case where $K = 1$, because there are existing algorithm(s) in the literature (see Sharkey et al. [140] or Chapter 6) that exploit the structure of CP with $K = 1$ and are thus competitors of the algorithm with simple pivots developed in this chapter. For $K = 1$, we considered 3 different classes of convex functions, each parameterized by a single parameter for which we chose 3 different values, for a total of 9 classes of problems. For each class of problems, we randomly generated 25 instances of the problem for $m = 1, 2, 5, 10, 20, 50, 100$ and $n = 50, 100, 200, 500, 1000$. (This is consistent with standard linear knapsack problems, where the number of items n is typically (much) larger than the number of knapsack constraints m.) We then applied the algorithm with simple pivots to each instance and, for each class of problems and problem size n, we recorded the average number of solutions generated to solve the 25 instances. To test the algorithm with generalized pivots, we also considered values of $K = 2, 5, 10$.

In Chapter 6, we consider a general problem class of the form CP where g is not required to be convex and $K = 1$. We develop an algorithm that has both a worst-case and an average-case running time of $O\left(n^2 \max\{\log n, \phi\}\right)$ for $m = 1$ and $O\left(n^{m+1} \max\{\log n,\ m^3, S(n,m), \phi\}\right)$ for $m \geq 2$, where ϕ is the time required to solve a one-dimensional optimization problem of the form (5–3) and $S(n,m)$ is the time required to solve a linear program with n variables and m constraints. If we let $C(n)$ be the *average* number of solutions that our algorithm with simple pivots generates for $K = 1$, then

the average time complexity of our algorithm can be expressed as $O\left(C(n)\max\{n,\phi\}\right)$ since a pivot operation requires $O(n+\phi)$ time. We will show that, on the classes of functions that we considered, it appears that $C(n) = O(n\log n)$ *independently* of m. This results thus illustrates the power of the algorithm with simple pivots, particularly for larger values of m.

The three classes of functions that we considered are:

- $g(y) = \gamma y^2$, for $\gamma = 1, 5$, and 20. In this case, the objective coefficients s_{ik} are generated from the uniform distribution on $\{-99,\ldots,100\}$.
- $g(y) = -\ln(\epsilon + y/\gamma)$, for $\epsilon = 0.01$ and $\gamma = 100, 500$, and 1000. In this case, the objective coefficients s_{ik} are generated from the uniform distribution on $\{1,\ldots,100\}$.
- $g(y) = e^{y/\gamma}$, for $\gamma = 10, 50$ and 100. In this case, the objective coefficients s_{ik} are generated from the uniform distribution on $\{1,\ldots,100\}$.

We first considered problems with only a single constraint (except for the bound constraints on x) and $K = 1$. For both the first and the second classes of functions, we observed that the average required number of solutions was insensitive to the value of γ. Based on a visual inspection of the data, we then performed a regression of the form $C(n) = \beta\ n\log n$. The scatter plot of the observations $(n, C(n))$ for each γ and the result of the regression appear in Figures 5-2 and 5-3. The R^2 values of the regressions were 0.999 and 0.998, respectively, so these results suggest that $C(n) = O(n\log n)$ for these classes of functions. For the third class of functions, we observed for this class of problems that the average case behavior of our algorithm does depend on γ, despite the fact that Corollary 5.2.3 states that the *worst-case* number of iterations required by the method should be independent of g (and therefore γ). Based on a visual inspection of the data, which suggested that the rate of increase in the average number of iterations as a function of n is slower than linear, we performed regression of the form $C(n) = \beta_0 n^{\beta_1}$ (or, equivalently, $\log C(n) = \log\beta_0 + \beta_1\log n$) for each individual value of γ. The scatter plot of the observations $(n, C(n))$ for each γ and the result of the regression appear in Figure

5-4. The R^2 value of the regressions were all larger than 0.99. Moreover, in all cases the estimate of β_1 was significantly smaller than 1, suggesting that $C(n) = O(n)$ for this class of functions.

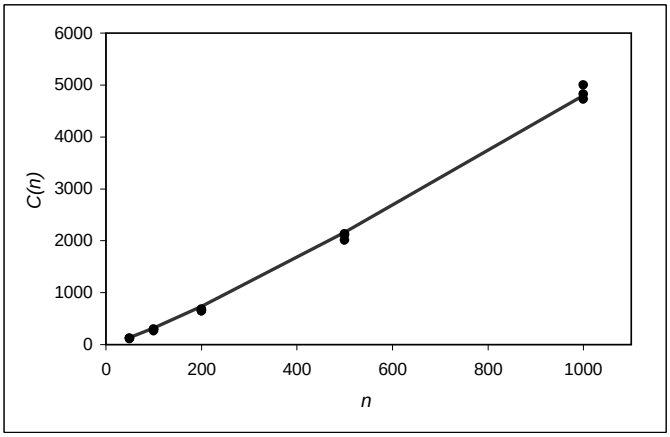

Figure 5-2. Scatter plot and best fit function for $g(y) = \gamma y^2$.

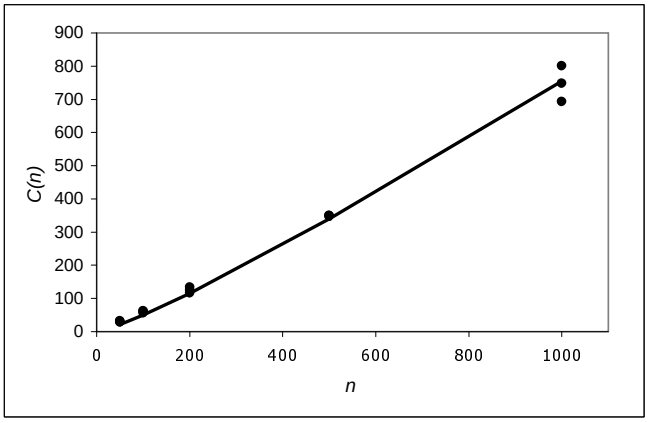

Figure 5-3. Scatter plot and best fit function for $g(y) = -\ln(\epsilon + y/\gamma)$.

We next considered problems with multiple constraints, in particular, $m = 2$, 5, 10, 20, 50, and 100. Very surprisingly, we consistently observed that the number of iterations required *decreased* as m was increased. Figure 5-5 illustrates this behavior for the function $g(y) = 20y^2$ class. For each value of m, we performed a regression of the form $C(n) = \beta\ n \log n$. The R^2 values of each of these regressions were greater than .99. This suggests that $C(n) = O(n \log n)$, i.e., the rate of growth of the number iterations is upper bounded by a function that is independent of m, for this class of functions and, therefore,

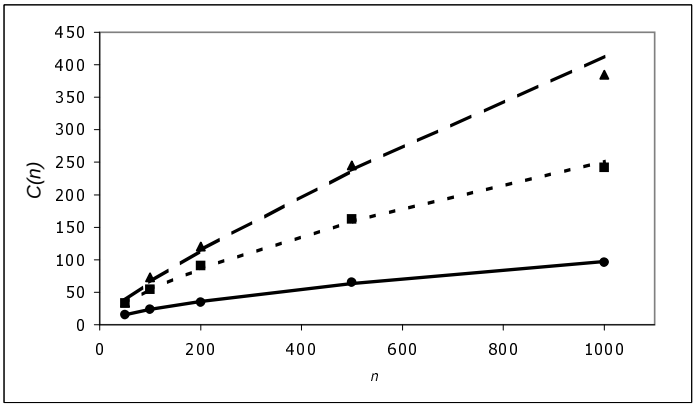

Figure 5-4. Scatter plots and best fit functions for $g(y) = e^{y/\gamma}$ for $\beta = 10$ (solid line), $\gamma = 50$ (dotted line), and $\gamma = 100$ (dashed line).

has a much improved average-time complexity over the algorithm of Sharkey et al. [140] for problems with $m \geq 2$. For each of the other classes of functions tested for $m = 1$, similar results were observed as the class $g(y) = 20y^2$ when we varied m.

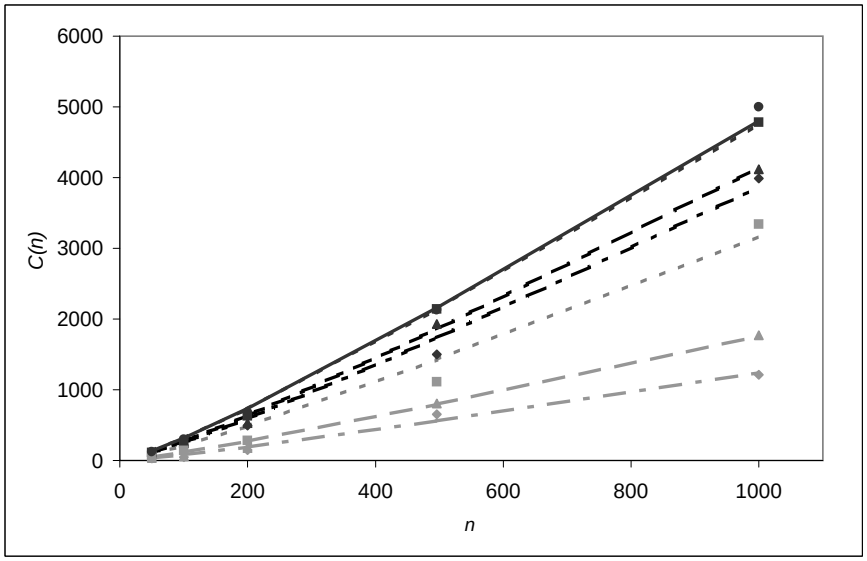

Figure 5-5. Scatter plot and best fit functions for $g(y) = 20y^2$ and $m = 1$ (solid line), $m = 2$ (black dotted line), $m = 5$ (black dashed line), $m = 10$ (black dotted/dashed line), $m = 20$ (grey dotted line), $m = 50$ (grey dashed line), and $m = 100$ (grey dotted/dashed line).

Finally, we studied the average case performance of the algorithm with generalized pivots. We first examined the algorithm with generalized pivots under a single linear constraint, i.e., $m = 1$, along with the upper bounds on the variables. We performed

testing on the algorithm on instances with $K = 2, 5$, and 10. For consistency with the case of $K = 1$ and suitability of the distributions that were chosen for the parameter values for that case, for the nonlinear part of the objective function we used functions of the form $g(Sx) = \sum_{k=1}^{K} g_k(s_k^\top x)$, where each of the functions g_k are generated as in the case of $K = 1$. That is, for each experiment, each of the functions g_k were taken from the same class and with the same value of γ, but each with their own parameters s_{ik}. After performing the tests, it became evident in the results that the average-case performance of this algorithm was dependent on K in an increasing fashion and, therefore, we define the function $C(n, K)$ as the average number of solutions that the algorithm with generalized pivots generates for n variables and K arguments. We are interested in testing the conjecture $C(n, K) = O(n^{\beta_1} K^{\beta_2})$ or, equivalently, we performed a log regression of the form $\log C(n, K) = \beta_0 + \beta_1 \log n + \beta_2 \log K$. For the first class of test problems we observed that $C(n, 1) > C(n, K)$ for $K = 2, 5, 10$ and all n, so we only performed the log regression based on the observations for (n, K) pairs with $K = 2, 5, 10$. This is justified because we are mainly interested in the asymptotic performance of the algorithm. The regression for the squared function had a R^2 value over .99 and indicated that $C(n, K) = O(n^{1.01} K^{.42})$. The scatter plot of $(n, C(n, K))$ and best fit function for the squared function appear in Figure 5-6. For the second and third classes of test problems we included all paris (n, K) in the log regression. For the logarithmic function, the R^2 value was over .97 and indicated that $C(n, K) = O(n^{1.26} K^{.39})$. The scatter plot and best fit function for the logarithmic function appear in Figure 5-7. For the exponential function, the R^2 value of the regression was over .96 and indicated that $C(n, K) = O(n^{1.18} K^{1.45})$. The scatter plot and best fit function for the exponential function appear in Figure 5-8. Based on the standard errors of the estimated parameters we safely conclude that the test results support that $C(n, K) = O(n^2 K^2)$ for the algorithm with generalized pivots on problems with $m = 1$; in fact, for particular classes of problems our results indicate that a much stronger result may in fact hold. Finally, an examination of results for the algorithm with generalized pivots

on instances where $n = 50, 100$ and with larger values of m indicated that, for fixed K, the average number of solutions generated tends to decrease with increasing m, which is consistent with the observed performance of the algorithm with simple pivots.

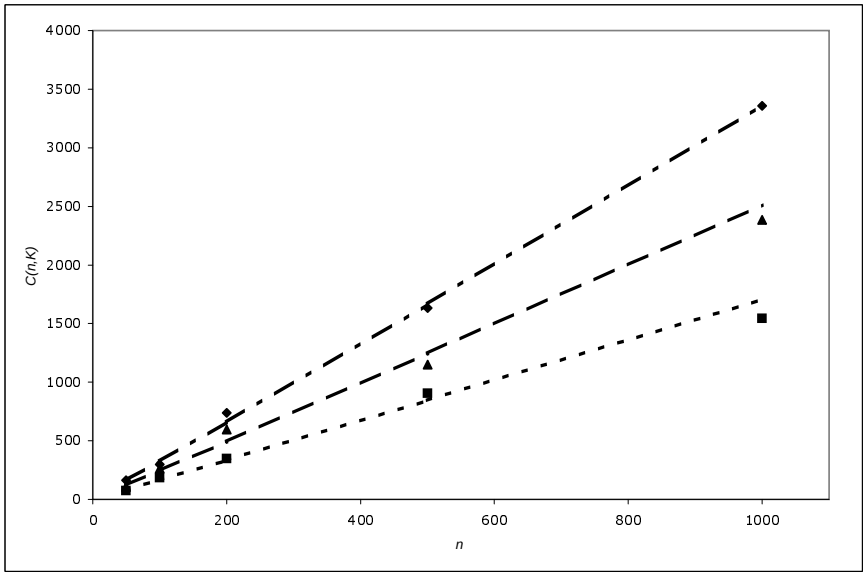

Figure 5-6. Scatter plots and best fit function for $g_k(y) = 20y^2$ and $K = 2$ (dotted line), $K = 5$ (dashed line) and $K = 10$ (dotted/dashed line).

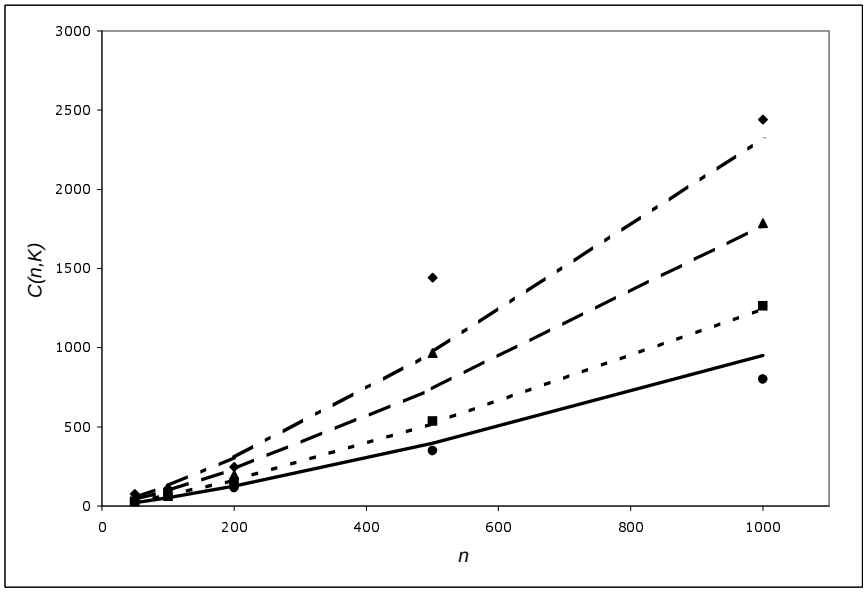

Figure 5-7. Scatter plots and best fit function for $g_k(y) = -\ln(\epsilon + \frac{y}{1000})$ and $K = 1$ (solid line), $K = 2$ (dotted line), $K = 5$ (dashed line) and $K = 10$ (dotted/dashed line).

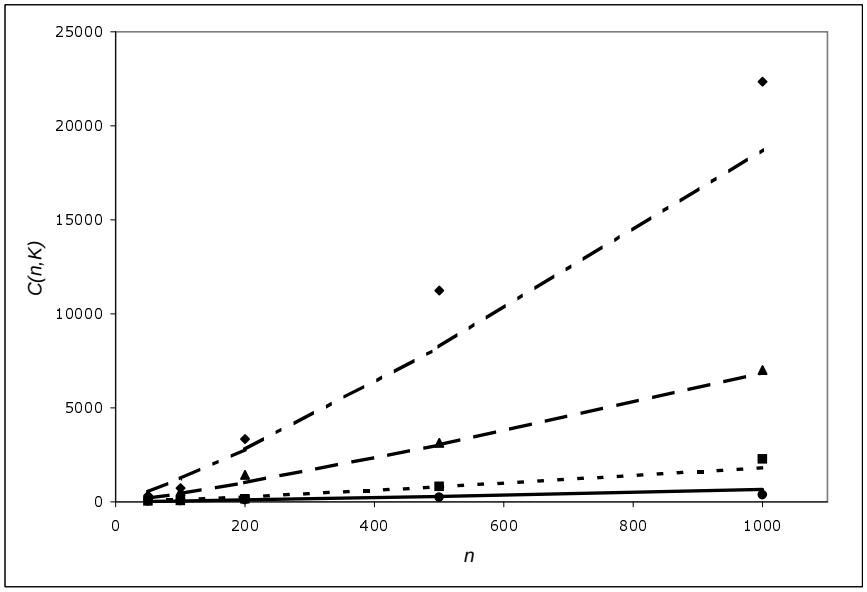

Figure 5-8. Scatter plots and best fit function for $g_k(y) = e^{y/100}$ and $K = 1$ (solid line), $K = 2$ (dotted line), $K = 5$ (dashed line) and $K = 10$ (dotted/dashed line).

5.5 Extensions

5.5.1 Non-Differentiable Objective Function

We have thus far assumed that the function g is convex with continuous partial derivatives. In many cases, nondifferentiability can be dealt with by introducing additional decision variables and constraints. For example, in the case of $K = 1$, if g is piecewise linear CP can be reformulated as a pure linear program. However, in general this may lead to a large increase in the dimensionality of the problem. Therefore, we will show in this section that we can extend our algorithm with generalized pivots to the case where the function g is continuous but not necessarily differentiable.

The main modification that is required is the definition of the term *eligible* nonbasic variable. To this end, we will let $\partial g(y)$ denote the usual set of subgradients of g at y. Furthermore, we define the set of *directional subgradients* of g at a solution y in the direction $d \neq 0$ to be:

$$\partial_d g(y) = \left[\lim_{h \uparrow 0} \frac{g\left(y + h\frac{d}{\|d\|}\right)}{h}, \lim_{h \downarrow 0} \frac{g\left(y + h\frac{d}{\|d\|}\right)}{h} \right].$$

Note that if this interval is a singleton, this value is equal to the directional derivative $\nabla_d g(y)$. Now if, in our algorithm, the current solution is (x, y) with corresponding generalized basis (x^B, y^B), we will say that a nonbasic variable is eligible if the search direction, (d^x, d^y), implied by this variable satisfies

$$0 \notin c^\top d^x + \|d^y\| \cdot \partial_{d^y} g(y).$$

This definition clearly generalizes the earlier definition in equation (5–4). Note that, as before, we should actually consider two opposing search directions for nonbasic variables in y^N. We now extend Theorem 5.2.1 to the non-differentiable case.

Theorem 5.5.1. *Let g be convex and continuous, but not necessarily differentiable. Let (x, y) be a generalized basic feasible solution to ACP with basic variables (x^B, y^B) and nonbasic variables (x^N, y^N). If there does not exist an eligible nonbasic variable then (x, y) is an optimal solution to ACP.*

Proof. This result follows in an analogous fashion to the proof of Theorem 5.2.1 by employing the generalized KKT conditions (see Hiriart-Urruty [79]). In particular, these generalized conditions replace equations (5–7) and (5–8) by

$$\begin{aligned} \nu_k &= v_{j_{n+k}} && \text{for } k \in B_y \\ \nu_k &= v^\top B^{-1} N_{j_{n+k}} && \text{for } k \in N_y \\ 0 &\in \partial g(y) + \nu. \end{aligned}$$

□

We can prove finite convergence of the algorithm with generalized pivots for general continuous and convex functions g in an analogous fashion as Theorem 5.2.2 and Theorem 5.3.1, invoking Theorem 5.5.1 where necessary.

5.5.2 Unbounded Feasible Region

For convenience, we have assumed that the feasible region P of CP is bounded. If this is not the case, it is not hard to show that our algorithm with generalized pivots will, in

a finite number of iterations, determine the optimal solution to CP or determine that the problem is unbounded. For our algorithm with simple pivots the situation is somewhat more complicated. It can be shown that our convergence results will still hold under the somewhat weaker assumption that the level sets of CP are bounded. If the problem CP is unbounded, value convergence of our algorithm with simple pivots still holds in the sense that the sequence of objective function values generated by the algorithms increases without bound as the number of iterations increases. However, while of course unboundedness is detected in finite time if subproblem (5–3) is unbounded, we cannot guarantee that unboundedness of the problem can be detected in finite time.

CHAPTER 6
A CLASS OF NONLINEAR NONSEPARABLE CONTINUOUS KNAPSACK AND MULTIPLE-CHOICE KNAPSACK PROBLEMS

In this chapter, we study global optimization problems of the form:

$$\max_{x \in X} \; r^\top x - g(s^\top x) \tag{P}$$

where $x = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$ is a vector of decision variables, $X \subseteq \mathbb{R}^n$, $r, s \in \mathbb{R}^n$, and g is any real-valued function. The main contribution of this chapter deals with the important case when

$$X = \{x \in [0,1]^n : b^\top x = B\} \tag{6–1}$$

where B is a constant. The problem P is then a continuous nonlinear, nonseparable knapsack problem. Note also that it is the continuous relaxation of the pricing problem for the NL-GAP. We also consider extensions of P where X contains more than one equality constraint (yielding a continuous nonlinear, nonseparable multi-knapsack problem) or contains a set of so-called multiple-choice knapsack constraints (see, e.g., Sinha and Zoltners [145]). The solution methods to solve P and its extensions that we develop in this chapter will rely on the analysis of a family of closely related linear programs. We will use the complementary slackness conditions of linear programming to characterize a set of solutions which will contain an optimal solution. This is in contrast to most approaches to solve nonlinear optimization problems where known optimality conditions (such as the KKT conditions) are fundamental to developing algorithms for these problems. This means that our approach does not require any restrictions on the function g; that is, we do not even require this function to be differentiable or continuous. In fact, the only assumption that we place upon P is the existence of an optimal solution to the problem. For the case of knapsack constraints, we will show that an optimal solution to P can be found by solving a number of one-dimensional optimization problems of the form $\max_{z \in [\ell, u]} \alpha z - g(z)$ that is polynomial in the number of decision variables, n. In the case

of multiple-choice knapsack problems, this number is polynomial in the number of choices per variable (or customer) as well.

Problems falling into the class of P often arise both naturally and as relaxations of discrete nonlinear knapsack problems. When solutions in which some or all of the decision variables are integral are of interest, it can be expected that these problems can be solved effectively using branch-and-bound based on the solution to the continuous relaxation. This expectation is based on the analogous observation for traditional knapsack problems and the fact that our solution approach can be implemented to find a solution with relatively few fractional components. In particular, we will show that an optimal solution to P exists with no more than two fractional components in the case of a single knapsack constraint, no more than two split items in the case of multiple-choice knapsack constraints, and no more than $m + 1$ fractional components in the case of m knapsack constraints.

The problem P with knapsack constraints not only serves as the relaxation of the pricing problem in solving the NL-GAP but also is a problem of independent interest in supply chain management. For example, consider a company which offers a product and wishes to maximize its profits by serving the demand for the product in a subset of n markets. The company receives a total of r_i units of revenue for serving the demand s_i for the product in market i ($i = 1, \ldots, n$), and suppose that there is an upper bound on total production (demand served) of U units. Let $x \in [0, 1]^n$ denote the vector of decision variables representing the fraction of each market's demand to be served. The function $g(z)$ would then represent a measure of the total cost related to producing or acquiring $z = s^\top x$ units of the product. Geunes et al. [67] study the problem where $g(z) = \sqrt{2Khz}$ represents the cycle stock cost associated with the classic economic order quantity function and is therefore concave. Similar problems appear in Shen et al. [141], Freling et al. [57], Teo and Shu [151], and Huang et al. [85].

Taaffe et al. [149] consider a class of selective newsvendor problems whose relaxation belongs to P. In that paper, the model considers a company that faces normally distributed demand in each of n available markets and wishes to select a subset of these markets to serve to maximize total expected profit. In addition, in each selected market, the advertising level may be set to determine the parameters of the demand distribution. Under various assumptions on the response of the demand distribution to advertising levels and the presence or absence of a budget constraint, this model falls within the problem class P as either a nonlinear knapsack problem or a nonlinear multiple-choice knapsack problem.

Recently, Romeijn et al. [122] considered a class of nonseparable knapsack problems that is of the form P with X given by (6–1), but they restrict themselves to the case where g is concave and locally Lipschitz continuous and the coefficients s and b are nonnegative. Since they deal with a convex maximization problem subject to a single linear knapsack constraint, they may restrict themselves to searching the extreme points of the feasible region (see Horst et al. [84]) for the optimal solution. The alternative solution method that we develop in this chapter is not only much more generally applicable than the algorithm developed in Romeijn et al. [122], but it also substantially improves the running time of the their algorithm when applied to the case where g is concave, from $O(n^3)$ time to $O(n^2 \log n)$.

The remainder of this chapter is organized as follows. Section 6.1 begins by focusing on the continuous nonlinear nonseparable knapsack problem. In Section 6.1.1, we examine a family of linear programs related to this knapsack problem to develop the structure of potential optimal solutions to the problem. In Section 6.1.2, we discuss our solution approach, and a special case of the problem, which plays an important role in the algorithm for the general case, is studied in Section 6.1.3. We analyze and improve the running time of the algorithm in Section 6.1.4. We discuss the complexity of determining the optimal solution to the problem with the fewest number of fractional variables in

Section 6.1.5. Section 6.2 deals with the case of multiple knapsack problems and Section 6.3 with multiple-choice knapsack problems. Section 6.4 discusses a computational comparison of the algorithms proposed in this chapter with a state-of-the-art commercial global optimization software package.

6.1 Single-Knapsack Problems

We start by considering the class of problems P with a single knapsack constraint, i.e., X is as in equation (6–1), and refer to this class of problems as KP. We will assume, without loss of generality, that there does not exist an item i such that $b_i = s_i = 0$. Otherwise, we could set $x_i = 1$ if $r_i > 0$ and $x_i = 0$ otherwise and simply eliminate that item from consideration. Furthermore, we assume that the matrix $(b \;\; s)$ has rank two; if not, the problem reduces to either a linear continuous knapsack problem or an instance of KP in which the equality constraint is absent. The former can be solved with the classical greedy method in $O(n \log n)$ time or using a median-finding algorithm (see, for example, Zemel [161] which generalized the work of Balas and Zemel [12]) in $O(n)$ time. The latter can be solved in $O(n \log n + n\phi)$ time, where ϕ is the time required to solve an optimization problem with objective function from the same class as g (see Huang et al. [85]).

Variants of KP in which the equality constraint is replaced by inequality constraints of the form $B' \leq b^\top x \leq B$ with $B' < B$ can be reformulated as instances of KP by introducing an additional variable $x_{n+1} \in [0, 1]$ with $r_{n+1} = s_{n+1} = 0$ and $b_{n+1} = B - B'$. Note that these problems are indeed equivalent: if $x \in [0, 1]^n$ satisfies $B' \leq b^\top x \leq B$ a feasible solution with the same objective function value for the transformed problem is obtained by letting $x_{n+1} = (B - b^\top x)/(B - B') \in [0, 1]$. Similarly, the vector containing the first n components of any feasible solution to the transformed problem is feasible for the original problem. Finally, if we have a one-sided inequality, $b^\top x \leq B$ (or $B' \leq b^\top x$), replacing the equality constraint, we may add in the implicit lower bound $B' = (b^-)^\top e \leq b^\top x$ where $b_i^- = \min\{b_i, 0\}$ and $e \in \mathbb{R}^n$ is a vector consisting of all ones (or

add in the implicit upper bound $b^\top x \leq B = (b^+)^\top e$ where $b_i^+ = \max\{b_i, 0\}$) and apply the reformulation technique as above.

In the remainder of this section, we will develop an algorithm to solve KP based on analysis of a family of closely related linear programs. Through the use of the complementary slackness conditions for linear programming, we will be able to construct partial candidate optimal solutions to KP. It is then shown that it suffices to solve two linear knapsack problems and a one-dimensional global optimization problem to construct full candidate optimal solutions from these partial solutions. The number of candidate optimal solutions that need to be considered is relatively small ($O(n^2)$) and we prove that one of these candidate solutions is optimal to KP.

6.1.1 Structure of Candidate Solutions

The following family of linear programming problems is closely related to KP and plays a critical role in our solution approach for KP:

$$\text{maximize } r^\top x$$

subject to (LKP)

$$s^\top x = S \tag{6–2}$$

$$b^\top x = B \tag{6–3}$$

$$x \in [0,1]^n \tag{6–4}$$

where S is a constant that parameterizes the family of problems. Clearly, if we were to know the value of $s^\top x$ in any optimal solution to KP we could solve the problem by solving the associated LKP. This observation can be used to show that there exists an optimal solution to KP with no more than two fractional components. That is, if we denote the set of indices corresponding to fractional components of a feasible solution x to KP by $F_x = \{i = 1, \ldots, n : 0 < x_i < 1\}$, we have the following result.

Theorem 6.1.1. *There exists an optimal solution x^* to KP such that $|F_{x^*}| \leq 2$.*

Proof. Theorem 3.0.1 shows that there exists an optimal solution x^* to KP that is also an extreme point optimal solution to LKP. Since x^* is a basic solution to LKP, only its two basic variables can be fractional, which proves the desired result. □

We will develop a solution approach by studying the structure of basic feasible and optimal solutions to problems of the form LKP. We will first employ the complementary slackness conditions associated with LKP, which are necessary for optimality, to characterize a partial primal solution that satisfies the complementary slackness conditions with a given feasible dual solution. Denoting the dual multipliers corresponding to constraints (6–3) and (6–2) by λ and γ, respectively, and the ones corresponding to the upper bounds in (6–4) by μ_i $(i = 1, \ldots, n)$, the complementary slackness conditions include

$$\begin{aligned} x_i(\lambda b_i + \gamma s_i + \mu_i - r_i) &= 0 \qquad i = 1, \ldots, n \\ \mu_i(1 - x_i) &= 0 \qquad i = 1, \ldots, n. \end{aligned}$$

These conditions motivate the following *forcing rule* that determines the value of some primal variables for a given dual solution:

Definition 6.1.2 (Forcing Rule)**.** *Given a feasible solution (λ, γ, μ) to the dual of LKP, the primal solution to LKP should satisfy for all $i = 1, \ldots, n$:*

$$\begin{aligned} r_i > \lambda b_i + \gamma s_i &\Rightarrow x_i = 1 \\ r_i = \lambda b_i + \gamma s_i &\Rightarrow x_i \in [0, 1] \\ r_i < \lambda b_i + \gamma s_i &\Rightarrow x_i = 0. \end{aligned}$$

Next, note that any basic feasible solution to LKP can be characterized by exactly two basic variables, indexed by say i and j, with the property that the constraint submatrix

corresponding to these two variables

$$\begin{pmatrix} b_i & b_j \\ s_i & s_j \end{pmatrix}$$

is invertible. For convenience, we will denote the set of pairs of variables that can form a basis by Δ, i.e.,

$$\Delta = \left\{ (i,j) : i < j \text{ and } \begin{pmatrix} b_i & b_j \\ s_i & s_j \end{pmatrix} \text{ is invertible.} \right\}.$$

For a basic solution given by $(i,j) \in \Delta$ to be optimal to LKP it must satisfy the complementary slackness conditions with a dual solution that satisfies

$$\begin{aligned} \lambda b_i + \gamma s_i &= r_i \\ \lambda b_j + \gamma s_j &= r_j. \end{aligned}$$

Since $(i,j) \in \Delta$, this system has a unique solution, say $(\lambda^{(i,j)}, \gamma^{(i,j)})$. We may now conclude that, in order to find an optimal solution to KP, we may restrict ourselves to solutions that satisfy the Forcing Rule (as given in Definition 6.1.2) with $(\lambda^{(i,j)}, \gamma^{(i,j)})$ for some $(i,j) \in \Delta$. Since these partial candidate solutions to LKP are *independent* of the value S, to solve KP we may in fact restrict ourselves to such solutions as well. We have thus obtained the result summarized in the following theorem.

Theorem 6.1.3. *In order to solve KP we may restrict ourselves to solutions $x^{(i,j)}$, $(i,j) \in \Delta$, that are of the following form. For $k = 1, \ldots, n$:*

$$\begin{aligned} x_k^{(i,j)} &= 0 && \text{if } r_k < \lambda^{(i,j)} b_k + \gamma^{(i,j)} s_k \\ x_k^{(i,j)} &\in [0,1] && \text{if } r_k = \lambda^{(i,j)} b_k + \gamma^{(i,j)} s_k \\ x_k^{(i,j)} &= 1 && \text{if } r_k > \lambda^{(i,j)} b_k + \gamma^{(i,j)} s_k. \end{aligned}$$

Note that if the basis given by $(i,j) \in \Delta$ is nondegenerate, the partial solution $x^{(i,j)}$ will contain a value of 0 or 1 for all nonbasic variables x_k (i.e., $k \neq i,j$) but does not specify the values of the basic variables x_i and x_j. However, in general, for some nonbasic variables x_k we may have

$$r_k = \lambda^{(i,j)} b_k + \gamma^{(i,j)} s_k$$

so that the complementary slackness conditions (or the Forcing Rule) cannot determine whether $x_k = 0$ or 1 either. However, in Section 6.1.2 we show how we can, nevertheless, construct a finite collection of full candidate solutions.

6.1.2 Constructing Full Candidate Solutions

We will discuss how to obtain a collection of full candidate solutions from the partial solutions identified by Theorem 6.1.3. Consider some fixed $(i,j) \in \Delta$ and define, for convenience, the sets of indices that decompose the set of variables into three categories according to the Forcing Rule:

$$\begin{aligned} I_0^{(i,j)} &= \{k : r_k < \lambda^{(i,j)} b_k + \gamma^{(i,j)} s_k\}, \\ I^{(i,j)} &= \{k : r_k = \lambda^{(i,j)} b_k + \gamma^{(i,j)} s_k\}, \end{aligned}$$

and

$$I_1^{(i,j)} = \{k : r_k > \lambda^{(i,j)} b_k + \gamma^{(i,j)} s_k\}.$$

As we mentioned before, $i,j \in I^{(i,j)}$ and usually, but not necessarily, we will have $I^{(i,j)} = \{i,j\}$. The Forcing Rule says that $x_k = 0$ for all $k \in I_0^{(i,j)}$ and $x_k = 1$ for all $k \in I_1^{(i,j)}$. We now determine the values of $x_k^{(i,j)}$ for $k \in I^{(i,j)}$ by solving the following optimization subproblem:

$$\text{maximize } \bar{R}^{(i,j)} + \sum_{k \in I^{(i,j)}} r_k x_k - g\left(\bar{S}^{(i,j)} + \sum_{k \in I^{(i,j)}} s_k x_k\right)$$

subject to ($\text{SP}^{(i,j)}$)

$$\begin{aligned} \sum_{k \in I^{(i,j)}} b_k x_k &= B - \bar{B}^{(i,j)} \\ x_k &\in [0,1] \qquad\qquad k \in I^{(i,j)} \end{aligned} \tag{6–5}$$

where

$$\bar{R}^{(i,j)} = \sum_{k \in I_1^{(i,j)}} r_k,\ \bar{S}^{(i,j)} = \sum_{k \in I_1^{(i,j)}} s_k, \text{ and } \bar{B}^{(i,j)} = \sum_{k \in I_1^{(i,j)}} b_k.$$

At first glance, this problem does not appear to have any special structure other than being of the form of a general KP but with $|I^{(i,j)}| \leq n$ variables. However, recalling that for all $k \in I^{(i,j)}$ we have that $r_k = \lambda^{(i,j)} b_k + \gamma^{(i,j)} s_k$, the objective of $\text{SP}^{(i,j)}$ can be written as:

$$\text{maximize } \bar{R}^{(i,j)} + \lambda^{(i,j)} \sum_{k \in I^{(i,j)}} b_k x_k + \gamma^{(i,j)} \sum_{k \in I^{(i,j)}} s_k x_k - g\left(\bar{S}^{(i,j)} + \sum_{k \in I^{(i,j)}} s_k x_k\right).$$

Finally, substituting constraint (6–5) into the objective function, we see that the objective function of the problem $\text{SP}^{(i,j)}$ can be reformulated as:

$$\text{maximize } \bar{R}^{(i,j)} + \lambda^{(i,j)}(B - \bar{B}^{(i,j)}) + \gamma^{(i,j)} \sum_{k \in I^{(i,j)}} s_k x_k - g\left(\bar{S}^{(i,j)} + \sum_{k \in I^{(i,j)}} s_k x_k\right).$$

This problem is therefore an instance of the class of problems KP where $r = \alpha s$ (in the case of $\text{SP}^{(i,j)}$, $\alpha = \gamma^{(i,j)}$). We will develop an efficient method to solve this class of problems in Section 6.1.3. But, first we prove that one of the full candidate solutions is optimal to KP.

Theorem 6.1.4. *The solution $\bar{x} = x^{(\bar{\imath},\bar{\jmath})}$, where $(\bar{\imath},\bar{\jmath}) = \arg\max\{r^\top x^{(i,j)} - g(s^\top x^{(i,j)}) : (i,j) \in \Delta\}$, is an optimal solution to KP.*

Proof. Consider any optimal solution x^* to KP. Suppose that variables x_i and x_j are basic in the basic feasible optimal solution to LKP with $S = s^\top x^*$. By construction, for all $k \notin I^{(i,j)}$ we have that $x_k^{(i,j)} = x_k^*$. This implies that x_k^* for all $k \in I^{(i,j)}$ is

a feasible solution to $\text{SP}^{(i,j)}$ with objective function equal to $r^\top x^* - g(s^\top x^*)$. Since $r^\top x^{(i,j)} - g(s^\top x^{(i,j)}) \geq r^\top x^* - g(s^\top x^*)$ and x^* is optimal to KP, we have that $x^{(i,j)}$ is optimal to KP. □

6.1.3 Solving a Special Case of KP

We will now study the special case of KP where $r = \alpha s$ for some $\alpha \in \mathbb{R}$, which we will call KP^0. In this case, we can solve the problem using a two-step approach. In the first step, we obtain tight bounds on the values of $s^\top x$ that we need to consider by solving the following two continuous knapsack problems:

$$L = \min_{x \in X} s^\top x \tag{6–6}$$

$$U = \max_{x \in X} s^\top x \tag{6–7}$$

where

$$X = \{x \in [0,1]^n : b^\top x = B\}.$$

Denote optimal solutions to these problems by x^L and x^U, respectively. Next, we solve the following one-dimensional optimization problem:

$$\max_{L \leq y \leq U} \alpha y - g(y) \tag{6–8}$$

and denote an optimal solution to this problem by y^*. It is easy to see that $\alpha y^* - g(y^*) \geq \alpha s^\top x - g(s^\top x)$ for all $x \in X$. Therefore, any solution $x^* \in X$ such that $s^\top x^* = y^*$ is an optimal solution to KP^0. Now note that such a solution is given by $x^* = a x^L + (1-a)x^U$, where $a \in [0,1]$ denotes the unique value for which $y^* = aL + (1-a)U$.

The procedure to solve KP^0 depends both on the structure of X and the properties of g. Since X is the feasible region of a continuous knapsack problem, problems (6–6) and (6–7) can be solved in $O(n)$ time using the algorithm of Zemel [161]. It also depends on the time required to solve (6–8), which is defined as follows.

Definition 6.1.5. *We denote the time required to solve problem (6–8) by ϕ.*

If g is concave, then (6–8) can be solved in constant time by simply evaluating and comparing $\alpha L - g(L)$ and $\alpha U - g(U)$. If g is convex then (6–8) can be solved in $O(\log(U - L))$ time by binary search over the interval $[L, U]$. There are many other instances of (6–8) that can be solved efficiently, for example if the objective function of (6–8) is a polynomial of degree three or any unimodal function. Although (6–8) may in general be a hard global optimization problem, it is important to note that the time required to solve it is *independent* of the number of variables n in the knapsack problem KP^0.

Recall that in the objective function of $\text{SP}^{(i,j)}$, the constant $\bar{S}^{(i,j)}$ in the function g corresponds to a slight redefinition of the function g when transitioning from $\text{SP}^{(i,j)}$ to KP^0. However, this redefinition simply translates into a shift of the bounds in the subproblem (6–8). Therefore, we conclude that we can solve the problem $\text{SP}^{(i,j)}$ in $O(|I^{(i,j)}| + \phi)$ time.

6.1.4 Solution Methods and Runtime Analysis

Recall from Theorem 6.1.4 that a basic algorithm would simply determine the solutions $x^{(i,j)}$ for all $(i, j) \in \Delta$. We will first analyze the running time of this approach and then discuss an improvement to the algorithm to ultimately obtain a running time of $O(n^2 \max\{\log n, \phi\})$.

To determine $x^{(i,j)}$ for some $(i, j) \in \Delta$, we need to (i) find $\lambda^{(i,j)}$ and $\gamma^{(i,j)}$, the partial solution as dictated by the Forcing Rule, and the parameters of the problem $\text{SP}^{(i,j)}$ in $O(n)$ time; (ii) solve the first phase of the associated instance of $\text{SP}^{(i,j)}$ in $O(|I^{(i,j)}|)$ time; and (iii) solve the second phase of $\text{SP}^{(i,j)}$ in $O(\phi)$ time. Since $|\Delta| = O(n^2)$ and $|I^{(i,j)}| = O(n)$, this immediately leads to a running time of $O(n^2 \max\{n, \phi\})$ for the algorithm.

We can reduce the running time of the algorithm by employing a relationship between the subproblems that include a particular item i. For a given item i let $\Delta_i = \{j : (i, j) \in \Delta\}$ characterize the items that remain to be considered together with item i. (Note that it

is possible that $\Delta_i = \emptyset$.) We will next show that, by considering the elements of Δ_i in a particular order, we may reduce the worst-case running time derived above.

We will first consider an item for which $s_i \neq 0$. Clearly, we only need to consider pairs (λ, γ) that satisfy $r_i = \lambda b_i + \gamma s_i$, i.e., pairs of the form $(\lambda, (r_i - \lambda b_i)/s_i)$. Moreover, the Forcing Rule says that, to determine the value of x_j for some pair (λ, γ), we should compare r_j to

$$\lambda b_j + \gamma s_j = \lambda(b_j - b_i s_j/s_i) + r_i s_j/s_i.$$

From this, we first conclude that if we consider an item $j \notin \Delta_i$ (i.e., an item for which $b_j s_i - b_i s_j = 0$), the Forcing Rule compares r_j to $r_i s_j/s_i$, independent of λ. In this case, there are two possibilities. If $r_j = r_i s_j/s_i$, the value of x_j cannot be determined by the Forcing Rule for any value of λ. We let

$$D_i = \{j : j > i,\, b_j s_i = b_i s_j,\, r_j s_i = r_i s_j\} \tag{6–9}$$

denote the relevant set of items. On the other hand, if $r_j \neq r_i s_j/s_i$, the value of x_j is determined by the Forcing Rule and independent of λ. Next, if $j \in \Delta_i$ and we consider values of λ in increasing order, the value of x_j cannot be determined by the Forcing Rule precisely if $\lambda = \lambda^{(i,j)}$; moreover, exactly one of the following statements holds:

$$x_j = 0 \text{ if } \lambda < \lambda^{(i,j)} \quad \text{and} \quad x_j = 1 \text{ if } \lambda > \lambda^{(i,j)}$$
$$x_j = 1 \text{ if } \lambda < \lambda^{(i,j)} \quad \text{and} \quad x_j = 0 \text{ if } \lambda > \lambda^{(i,j)}.$$

This implies that, if we consider values of λ in increasing order, each item $j \in \Delta_i$ appears in a subproblem to be solved for at most one value of λ. In addition, the value of the corresponding variable is constant for all smaller values of λ as well as for all larger values of λ. Now recall that the only values of λ that we need to consider are the values $\lambda^{(i,j)}$ for $j \in \Delta_i$. At this point, we may note that if $\lambda^{(i,j)} = \lambda^{(i,k)}$ for some items $j, k \in \Delta_i$, the results of the Forcing Rule as well as the corresponding subproblems $\mathrm{SP}^{(i,j)}$ and $\mathrm{SP}^{(i,k)}$ are the same, so that we only need to consider one of these items. With a slight abuse of

notation, we will therefore assume in the remainder that we have removed appropriate items from Δ_i so that each $j \in \Delta_i$ corresponds to a unique value of $\lambda^{(i,j)}$. With this slight modification we conclude that, for a given item i such that $s_i \neq 0$, we should consider the items $j \in \Delta_i$ in increasing order of $\lambda^{(i,j)}$.

Next, consider an item i for which $s_i = 0$. Since our assumptions on the problem data then imply that $b_i \neq 0$, we only need to consider pairs of the form $((r_i - \gamma s_i)/b_i, \gamma)$. In that case, we achieve a similar result as above by considering items $j \in \Delta_i$ in increasing order of $\gamma^{(i,j)}$.

Since, for each i, the set D_i corresponds to the collection of variables that is included in all subproblems of the form $\mathrm{SP}^{(i,j)}$ they play a large role in reducing the running time of the algorithm. Therefore, it is important to take a closer look at the characterization of these sets. Recall that if $s_i \neq 0$ this set can be characterized as in equation (6–9). Similarly, if $b_i \neq 0$ this set can be characterized as $D_i = \{j : j > i,\, b_j s_i = b_i s_j,\, r_j b_i = r_i b_j\}$. An alternative characterization of the set D_i is that it is the set of items j with the property that $r_i = \lambda b_i + \gamma s_i$ implies that $r_j = \lambda b_j + \gamma s_j$. This characterization immediately leads to the conclusion that, once we have considered all solutions corresponding to item i, we do not need to consider the solutions corresponding to any item $j \in D_i \setminus \{i\}$, leading to a significant reduction in computational effort. Put differently, the sets D_i effectively form a partition of the set of items, which we can index by a set $I \subseteq \{1, \ldots, n\}$ with the property that

$$D_i \cap D_j = \emptyset \text{ for all } i, j \in I \quad \text{and} \quad \bigcup_{i \in I} D_i = \{1, \ldots, n\}.$$

We are now ready to present our algorithm for solving KP.

KP Algorithm

Step 0. Set $\bar{I} = \emptyset$.

Step 1. Select any $i \in I \setminus \bar{I}$. Compute $(\lambda^{(i,j)}, \gamma^{(i,j)})$ for all j such that $(i,j) \in \Delta$, determine the set Δ_i, and denote the appropriately sorted items in Δ_i by $j_1, \ldots, j_{|\Delta_i|}$.

Step 2. Apply the Forcing Rule with $(\lambda^{(i,j_1)}, \gamma^{(i,j_1)})$ to determine $x_k^{(i,j_1)}$ for $k \notin I^{(i,j_1)}$. Set

$$\bar{R}^{(i,j_1)} = \sum_{k:x_k^{(i,j_1)}=1} r_k,\ \bar{S}^{(i,j_1)} = \sum_{k:x_k^{(i,j_1)}=1} s_k, \text{ and } \bar{B}^{(i,j_1)} = \sum_{k:x_k^{(i,j_1)}=1} b_k.$$

Solve $\mathrm{SP}^{(i,j_1)}$. Set $\ell = 2$.

Step 3. Set $x_k^{(i,j_\ell)} = x_k^{(i,j_{\ell-1})}$ for all $k \notin I^{(i,j_{\ell-1})} \cup I^{(i,j_\ell)}$. Apply the Forcing Rule with $(\lambda^{(i,j_\ell)}, \gamma^{(i,j_\ell)})$ to determine $x_k^{(i,j_\ell)}$ for all $k \in I^{(i,j_{\ell-1})}$. Set

$$\begin{aligned}
\bar{B}^{(i,j_\ell)} &= \bar{B}^{(i,j_{\ell-1})} - \sum_{k \in I^{(i,j_\ell)}} b_k x_k^{(i,j_{\ell-1})} + \sum_{k \in I^{(i,j_{\ell-1})}} b_k x_k^{(i,j_\ell)} \\
\bar{S}^{(i,j_\ell)} &= \bar{S}^{(i,j_{\ell-1})} - \sum_{k \in I^{(i,j_\ell)}} s_k x_k^{(i,j_{\ell-1})} + \sum_{k \in I^{(i,j_{\ell-1})}} s_k x_k^{(i,j_\ell)} \\
\bar{R}^{(i,j_\ell)} &= \bar{R}^{(i,j_{\ell-1})} - \sum_{k \in I^{(i,j_\ell)}} r_k x_k^{(i,j_{\ell-1})} + \sum_{k \in I^{(i,j_{\ell-1})}} r_k x_k^{(i,j_\ell)}.
\end{aligned}$$

Solve $\mathrm{SP}^{(i,j_\ell)}$.

Step 4. If $\ell < |\Delta_i|$, set $\ell = \ell + 1$ and return to Step 3. Otherwise, set $\bar{I} = \bar{I} \cup \{i\}$; if $I \setminus \bar{I} \neq \emptyset$, return to Step 1.

The following theorem derives the running time of this algorithm.

Theorem 6.1.6. *We can find an optimal solution to KP in $O(n^2 \max\{\log n, \phi\})$ time.*

Proof. We will first show that we can, for a fixed $i \in I$, find the optimal objective function values to the problems $\mathrm{SP}^{(i,j)}$ for $j \in \Delta_i$ in $O(n \log n + n|D_i| + n\phi)$ time. The bottleneck of Step 1 of the algorithm is the sorting of the items in Δ_i, which requires $O(n \log n)$ time. Step 2 requires $O(n)$ time to compute the parameters of problem $\mathrm{SP}^{(i,j_1)}$. Since each item $k \notin D_i$ is in $I^{(i,j_\ell)}$ for at most one value of ℓ, a total of $O(n)$ time is needed to update the subproblem parameters in all executions of Step 3. Next, solving a subproblem of the form $\mathrm{SP}^{(i,j)}$ requires $O(|I^{(i,j)}|)$ time to find the bounds using problems of the form

(6–6) and (6–7), and $O(\phi)$ time to solve the one-dimensional optimization problem. Since $I^{(i,j_\ell)} \cap I^{(i,j_{\ell-1})} = D_i$ for all $\ell = 2, \ldots, |\Delta_i|$ (i.e., each item $k \notin D_i$ appears in at most one subproblem), this implies that the time required to solve all subproblems encountered in Steps 2 and 3 is $O\left(\sum_{j \in \Delta_i}(|I^{(i,j)}| + \phi)\right) = O(n + n|D_i| + n\phi)$. The total time required for a given $i \in I$ is thus $O(n \log n + n|D_i| + n\phi)$.

Once we have determined the optimal objective function values of $\text{SP}^{(i,j)}$ for $(i,j) \in \Delta$, we can determine the optimal solution x^* to KP in $O(n + \phi)$ time by resolving $\text{SP}^{(i^*,j^*)}$ for the pair (i^*, j^*) achieving the maximum objective function value. This now yields that the total time required to find an optimal solution to KP can be found in

$$O\left(n + \phi + \sum_{i \in I}(n \log n + n|D_i| + n\phi)\right) = O(n^2 \log n + n^2 + n^2\phi) = O(n^2 \max\{\log n, \phi\})$$

time. □

Since the function g is not assumed to possess any particular structure, this result suggests that this algorithm achieves a near-optimum running time for the problem of determining the objective function value for all $x^{(i,j)}$, $(i,j) \in \Delta$, since there are $O(n^2)$ pairs of variables that need to be considered.

Romeijn et al. [122] studied a subclass of KP in which the function g is concave and locally Lipschitz continuous and s and b are nonnegative vectors. (Note that the nonnegativity of one of these vectors can be assumed without loss of generality, but not the nonnegativity of both.) Concavity of g implies that there exists an extreme point optimal solution, i.e., a solution in which at most one variable is fractional. They developed a solution approach based on generalized KKT-conditions for KP that finds an optimal solution with that property in $O(n^3)$ time. It is easy to see that our algorithm, for the case where g is concave, runs in $O(n^2 \log n)$ time since ϕ is a constant. Moreover, since each candidate solution $x^{(i,j)}$ is given by an extreme point of the feasible region of $\text{SP}^{(i,j)}$ together with the binary components as dictated by the Forcing Rule, the algorithm presented in this chapter will find a solution with at most one fractional variable. We can

thus conclude that the new algorithm solves a generalization of the problem studied in Romeijn et al. [122] in reduced worst-case running time.

6.1.5 Solution with Smallest Number of Fractional Components

Consider the integer version of the continuous knapsack problem KP. Its relaxation KP then may appear as a subproblem in, for example, a branch-and-bound solution approach. In such cases, it may be desirable to determine an optimal solution with the smallest number of fractional variables. Recall that Theorem 6.1.1 guarantees that there exists an optimal solution with no more than two fractional components. If the algorithm does not return such a solution, we can of course recover one by solving an instance of LKP with the value of S obtained in the optimal solution found. A solution with fewer fractional values, if one exists, could in principle be found as follows. After determining $x^{(i,j)}$ for some $(i,j) \in \Delta$, we may attempt to find a corresponding candidate solution with no more than one fractional variable by, in turn, fixing each of the variables in $I^{(i,j)}$ and finding optimal solutions to $\text{SP}^{(i,j)}$ for each possible choice of binary values for the remaining variables in $I^{(i,j)}$. In addition, we could consider each possible choice of binary values for all variables in $I^{(i,j)}$. Clearly, this procedure will identify an optimal solution with the smallest number of fractional variables at the expense of considering $O(2^{|I^{(i,j)}|})$ additional solutions for each $(i,j) \in \Delta$. In the, relatively common, situation where $I^{(i,j)} = \{i,j\}$ and thus $|I^{(i,j)}| = 2$, this does not change the order of the running time of the algorithm. However, we will next show that this approach does not lead to a polynomial time algorithm in general unless $P = NP$.

Theorem 6.1.7. *Consider the problem of determining the optimal solution with the minimal number of fractional variables to KP. If $\mathcal{I} \equiv \max_{(i,j)\in\Delta} |I^{(i,j)}|$ grows linearly in n this problem is NP-hard.*

Proof. In particular, we will show that the problem of determining the optimal solution to KP with the fewest number of fractional variables is NP-hard if $\mathcal{I}$ is a linear function of n by a reduction from the Subset Sum problem (see, e.g., Garey and Johnson [62]). The

Subset Sum Problem is defined as follows: Given a set of positive integers $b_1, b_2, \ldots, b_n$ and a target integer B, does there exist $y \in \{0,1\}^n$ such that $b^\top x = B$? We consider the following knapsack problem which belongs to the class KP:

$$\text{maximize} \quad -\cos\left(\pi(2e^\top x + 1)\right)$$

subject to (SS)

$$\begin{aligned} b^\top x &= B \\ x &\in [0,1]^n \end{aligned}$$

where $e \in \mathbb{R}^n$ is a vector all of whose elements are equal to one. Note that SS has a solution with no fractional variables if and only if the Subset Sum problem has a solution. Moreover, it is easy to see that $I^{(i,j)} = \{1, \ldots, n\}$ for all $(i,j) \in \Delta$, so that $\mathcal{I} = n$. □

6.2 The Multi-Knapsack Problem

The algorithm for a class of nonlinear knapsack problems that we developed in Section 6.1 can be extended to problems with multiple linear constraints. We will refer to this class of problems as nonseparable, nonlinear multi-knapsack problems, that may be formulated as follows:

$$\max r^\top x - g(s^\top x)$$

subject to (MKP)

$$\begin{aligned} b_{\cdot j}^\top x &= B_j \qquad j = 1, 2, \ldots, m \\ x &\in [0,1]^n \end{aligned} \tag{6–10}$$

where g, p, s, and x are as in KP, $b_{\cdot j} = (b_{1j}, \ldots, b_{nj})^\top \in \mathbb{R}^n$ $(j =, 1 \ldots, m)$ are vectors of coefficients, and $B_j \in \mathbb{R}$ $(j = 1, \ldots, m)$ are coefficients as well. We assume that the matrix $(b_{\cdot 1} \ \cdots \ b_{\cdot m} \ s)$ has rank $m+1$; if not, the problem reduces to either a linear program or

an instance of MKP with fewer constraints in (6–10). It is straightforward to extend the result of Theorem 6.1.1 to MKP:

Theorem 6.2.1. *There exists an optimal solution x^* to MKP such that $|F_{x^*}| \leq m + 1$*

Proof. This follows in a similar way as the result of Theorem 6.1.1 from a study of the structure of extreme point solutions to the linear program

$$\text{maximize } r^\top x$$

subject to (LMKP)

$$\begin{aligned} s^\top x &= S \\ b_{\cdot j}^\top x &= B_j \qquad j = 1, \ldots, m \\ x &\in [0,1]^n. \end{aligned}$$

□

For convenience, let us define the vectors $b_{i\cdot} = (b_{i1}, \ldots, b_{im})^\top$ $(i = 1, \ldots, n)$. A study of the complementary slackness conditions associated with LMKP then yields the following generalization of the Forcing Rule:

Definition 6.2.2 (Forcing Rule)**.** *Given a dual feasible solution (λ, γ, μ) (where $\lambda \in \mathbb{R}^m$) the primal solution should satisfy for all $i = 1, \ldots, n$:*

$$\begin{aligned} r_i > \lambda^\top b_{i\cdot} + \gamma s_i &\Rightarrow x_i = 1 \\ r_i = \lambda^\top b_{i\cdot} + \gamma s_i &\Rightarrow x_i \in [0,1] \\ r_i < \lambda^\top b_{i\cdot} + \gamma s_i &\Rightarrow x_i = 0. \end{aligned}$$

Next, the set Δ is extended as follows:

$$\Delta = \left\{ (i_1, \ldots, i_{m+1}) : 1 \leq i_1 < \cdots < i_{m+1} \leq n \text{ and } \begin{pmatrix} b_{i_1 1} & \cdots & b_{i_{m+1},m+1} \\ \vdots & \ddots & \vdots \\ b_{i_1 m} & \cdots & b_{i_{m+1},m} \\ s_{i_1} & \cdots & s_{i_{m+1},m} \end{pmatrix} \text{ is invertible} \right\}.$$

For $\delta \in \Delta$, define $(\lambda^\delta, \gamma^\delta)$ to be the unique solution to the system

$$\lambda^\top b_{i_\ell \cdot} + \gamma s_{i_\ell} = r_{i_\ell} \qquad \ell = 1, \ldots, m+1$$

and the following sets of variables that generalize $I_0^{(i,j)}$, $I^{(i,j)}$, and $I_1^{(i,j)}$ from the KP:

$$\begin{aligned} I_0^\delta &= \{k : r_k < (\lambda^\delta)^\top b_{k\cdot} + \gamma^\delta s_k\} \\ I^\delta &= \{k : r_k = (\lambda^\delta)^\top b_{k\cdot} + \gamma^\delta s_k\} \end{aligned}$$

and

$$I_1^\delta = \{k : r_k > (\lambda^\delta)^\top b_{k\cdot} + \gamma^\delta s_k\}.$$

This then leads to the following extension of Theorem 6.1.3:

Theorem 6.2.3. *In order to solve MKP we may restrict ourselves to solutions x^δ, $\delta \in \Delta$, that are of the following form. For $k = 1, \ldots, n$:*

$$\begin{aligned} x_k^\delta &= 0 && \text{if } k \in I_0^\delta \\ x_k^\delta &\in [0,1] && \text{if } k \in I^\delta \\ x_k^\delta &= 1 && \text{if } k \in I_1^\delta. \end{aligned}$$

In a similar fashion as for the KP, we can now show that we can find a candidate solution for each $\delta \in \Delta$ by solving the problem

$$\text{maximize } \bar{R}^\delta + \sum_{j=1}^m \lambda_j^\delta (B_j - \bar{B}_j^\delta) + \gamma^\delta \sum_{k \in I^\delta} s_k x_k - g\left(\bar{S}^\delta + \sum_{k \in I^\delta} s_k x_k\right)$$

subject to (SP$^\delta$)

$$\begin{aligned} \sum_{k \in I^\delta} b_{kj} x_k &= B_j - \bar{B}_j^\delta \qquad j = 1, \ldots, m \\ x_k &\in [0,1] \qquad\qquad k \in I^\delta \end{aligned}$$

where

$$\bar{R}^\delta = \sum_{k \in I_1^\delta} r_k,\ \bar{S}^\delta = \sum_{k \in I_1^\delta} s_k, \text{ and } \bar{B}_j^\delta = \sum_{k \in I_1^\delta} b_{kj},\ j = 1, \ldots, m.$$

This problem is in fact an instance of a subclass of MKP that is obtained when $r = \alpha s$ for some $\alpha \in \mathbb{R}$, say MKP^0, in particular with $\alpha = \gamma^\delta$ and the set of decision variables corresponding to the index set I^δ. This problem is very similar to KP^0; the only change is that, in the first step, the set X should now read

$$X = \{x \in [0,1]^n : b_j^\top x = B_j,\ j = 1, \ldots, m\}.$$

We will denote the time required to solve a linear program of the form (6–6) or (6–7) with n variables and m constraints by $S(n, m)$. The second step still consists of a one-dimensional optimization problem of the form (6–8).

Theorem 6.2.4. *The solution $\bar{x} = x^{\bar{\delta}}$, where $\bar{\delta} = \arg\max\{r^\top x^\delta - g(s^\top x^\delta) : \delta \in \Delta\}$, is an optimal solution to MKP.*

Proof. We consider any optimal solution x^* to MKP. If we let δ correspond to the set of basic variables in the optimal solution to LMKP defined with $S = s^\top x^*$, we can show in an analogous fashion to Theorem 6.1.4 that x^δ is an optimal solution to MKP. □

Let us next discuss the runtime of the solution method. In a straightforward manner, we can conclude that the time required by the algorithm is $O(n^{m+1} \max\{n, m^3, S(n, m), \phi\})$ since the size of Δ is $O(n^{m+1})$ and we require: (i) $O(m^3)$ time to find $(\lambda^\delta, \gamma^\delta)$; (ii) $O(n)$ time to apply the Forcing rule; (iii) $O(S(n, m) + \phi)$ time to solve the subproblem. We can, however, improve the running time of the algorithm in a similar manner as we did

in the single-knapsack case, again replacing n by $\log n$ inside the $\max\{\cdot\}$-expression in the running time. We consider solving for all solutions $\delta \in \Delta$ with a sequence of m increasing item indices $\psi = (i_1, \ldots, i_m)$ in a particular order. We then generalize Δ_i to $\Delta_\psi = \{j : (\psi, j) \in \Delta\}$. If $\Delta_j \neq \emptyset$, the matrix with columns $(b_{i_k 1}, \ldots, b_{i_k m}, s_i)^\top$ for $k = 1, \ldots, m$ must, by definition, have rank m. Furthermore, we generalize D_i by defining D_ψ to be the set of items j such that if (λ, γ) satisfy $r_i = \lambda^\top b_{i\cdot} + \gamma s_i$ for all $i \in \psi$ then $r_j = \lambda^\top b_j + \gamma s_j$ as well. Note that we can construct D_ψ in $O(nm^3)$ time by Gaussian elimination.

For a given ψ, we should now order the items $j \in \Delta_\psi$ according to one of the elements of $(\lambda^{(\psi,j)}, \gamma^{(\psi,j)})$. Recall that, in the single-knapsack case, given a choice of item i, we expressed one of the elements of (λ, γ) linearly as a function of the other. This can be extended to the multi-knapsack problem by, given a vector of items ψ, finding an element of (λ, γ) that can be expressed in terms of all others. Put differently, we should find a component of (λ, γ) with the property that it is not constant among $(\lambda^{(\psi,j)}, \gamma^{(\psi,j)})$ for all $j \in \Delta_\psi$. We then sort the items $j \in \Delta_\psi$ according to that component. As in the case of the KP, in case of ties we only need to consider one of the corresponding items and reduce the set Δ_ψ accordingly.

It is not hard to generalize results obtained for the KP that say that, if we consider items in the order given above, each item $j \notin D_\psi$ appears in a subproblem to be solved at most once. In addition, the value of the corresponding variable is constant in subproblems that are considered earlier and later, respectively.

We can now derive the following result:

Theorem 6.2.5. *We can find an optimal solution to MKP in*

$$O\left(n^{m+1} \max\{\log n, m^3, S(n,m), \phi\}\right)$$

time.

Proof. We will first show that we can, for a fixed ψ, find the optimal solution to the problems $\mathrm{SP}^{(\psi,j)}$ for $j \in \Delta_\psi$ in $O\left(nm^3 + n\log n + nS(n,m) + n\phi\right)$ time. This can be seen as follows.

Firstly, we require $O(nm^3)$ time to determine the vectors $(\lambda^{(\psi,j)}, \gamma^{(\psi,j)})$ for all $j \in \Delta_\psi$. We next identify an element of these vectors to sort the items by, which requires $O(nm)$ time. Finally, we sort the items in Δ_ψ accordingly, requiring $O(n\log n)$ time. Step 2 requires $O(n)$ time to compute the parameters of problem $\mathrm{SP}^{(\psi,j_1)}$. Since each item $k \notin D_\psi$ is in $I^{(\psi,j_\ell)}$ for at most one value of ℓ, a total of $O(n)$ time is needed to update the subproblem parameters in all executions of Step 3.

Secondly, solving a subproblem of the form $\mathrm{SP}^{(\psi,j)}$ requires $O\left(S(n,m)\right)$ time to find the bounds using problems of the form (6–6) and (6–7) and $O(\phi)$ time to solve the one-dimensional optimization problem. This implies that the time required to solve all subproblems encountered in Steps 2 and 3 is $O\left(\sum_{j\in\Delta_\psi}(S(n,m)+\phi)\right) = O\left(nS(n,m) + n\phi\right)$, so that the total time required for a given ψ is indeed

$$O\left(nm^3 + n\log n + nS(n,m) + n\phi\right).$$

Defining $\Psi = \{\psi \subseteq \{1,\dots n\} : |\psi| = m\}$, this now yields that the total time required to find an optimal solution to KP can be found in

$$\begin{aligned} O\left(\sum_{\psi\in\Psi}(nm^3 + n\log n + nS(n,m) + n\phi)\right) &= O\left(n^m(nm^3 + n\log n + nS(n,m) + n\phi)\right) \\ &= O\left(n^{m+1}\max\{m^3, \log n, S(n,m), \phi\}\right) \end{aligned}$$

time. □

This improvement does not seem to have much effect on the running time, since typically $S(n,m)$ will dominate both m^3 and n. Typically, though, $S(n,m)$ will be a loose upper bound on the time required to solve the linear programs. The running time for a given instance of MKP will replace $S(n,m)$ with $S(\max_{\delta\in\Delta}|I^\delta|, m)$. In most cases,

$\max_{\delta\in\Delta} |I^\delta| \ll n$, so this improvement is worthwhile (one can expect that, typically, $|I^\delta| = m + 1$).

6.3 A Multiple-Choice Knapsack Problem

In this section, we consider the following class of nonlinear, nonseparable multiple-choice knapsack problems:

$$\text{maximize } \sum_{i=1}^{n}\sum_{j=1}^{v_i} r_{ij}x_{ij} - g\left(\sum_{i=1}^{n}\sum_{j=1}^{v_i} s_{ij}x_{ij}\right)$$

subject to (MCKP)

$$\sum_{i=1}^{n}\sum_{j=1}^{v_i} b_{ij}x_{ij} = B \tag{6–11}$$

$$\sum_{j=1}^{v_i} x_{ij} = 1 \qquad i = 1,\dots n \tag{6–12}$$

$$x_{ij} \geq 0 \qquad j = 1,\dots,v_i,\ i = 1,\dots n$$

where v_i is the number of variants of item i ($i = 1,\dots,n$). For convenience, we define $v = \max_{i=1,\dots,n} v_i$. Note that this class of problems also includes problems where the capacity constraint (6–11) is replaced by a constraint of the form

$$L \leq \sum_{i=1}^{n}\sum_{j=1}^{v_i} b_{ij}x_{ij} \leq B.$$

This follows from the fact that we may convert such a problem to a problem of the form MCKP by adding another item, say $n + 1$, with two variants, say 1 and 2. We set $r_{n+1,1} = r_{n+1,2} = s_{n+1,1} = s_{n+1,2} = b_{n+1,2} = 0$ and $b_{n+1,1} = B - L$. These problems are equivalent since, for any feasible solution to the inequality constrained problem, we may set

$$x_{n+1,1} = \frac{B - \sum_{i=1}^{n}\sum_{j=1}^{v_i} b_{ij}x_{ij}}{B - L} \text{ and } x_{n+1,2} = 1 - x_{n+1,1}$$

to obtain a solution to the equality constrained problem and, similarly, truncating any feasible solution x to the associated MCKP to the first n items only yields a feasible

solution to the inequality constrained problem. For the case of a one-sided inequality, it is not hard to extend this argument using the implicit upper (or lower) bound on $\sum_{i=1}^{n} \sum_{j=1}^{v_i} b_{ij} x_{ij}$.

Note that MCKP can be viewed as a special case of the extension of KP that allows for multiple equality constraints MKP that is studied in Section 6.2, namely with $\sum_{i=1}^{n} v_i = O(nv)$ items and $n + 1$ constraints. However, applying the procedure developed for this problem class would lead to a solution procedure that requires the solution of a number of one-dimensional optimization problems that is exponential in the number of items n. In this section, we will develop a tailored algorithm for MCKP that requires the solution of a number of such problems that is polynomial in both the number of items n and the number of variants per item v.

6.3.1 Structure of Candidate Solutions

As was the case for KP, there is an important family of linear programming problems that is closely related to MCKP:

$$\text{maximize} \quad \sum_{i=1}^{n} \sum_{j=1}^{v_i} r_{ij} x_{ij}$$

subject to (LMCKP)

$$\sum_{i=1}^{n} \sum_{j=1}^{v_i} b_{ij} x_{ij} = B \tag{6–13}$$

$$\sum_{i=1}^{n} \sum_{j=1}^{v_i} s_{ij} x_{ij} = S \tag{6–14}$$

$$\sum_{j=1}^{v_i} x_{ij} = 1 \qquad i = 1, \ldots n \tag{6–15}$$

$$x_{ij} \geq 0 \qquad j = 1, \ldots, v_i,\ i = 1, \ldots n$$

This linear programming problem can be used to study the structure of solutions to MCKP that we may restrict ourselves to without loss of optimality. In particular, we will first use it to characterize the ways in which items may be *split* in an optimal solution

to MCKP. Given a feasible solution x, we say that item i is a m-way split item if $|\{j = 1, \ldots, v_i : x_{ij} > 0\}| = m$.

Theorem 6.3.1. *There exists an optimal solution, say x^*, to MCKP such that no more than one of the following holds:*

(i) there are at most two two-way split items, i.e. there exists (i, j, k) and (i', j', k') such that $x^*_{ij}, x^*_{ik} > 0$ and $x^*_{i'j'}, x^*_{i'k'} > 0$;

(ii) there is at most one three-way split item, i.e. there exists (i, j, k, ℓ) such that $x^*_{ij}, x^*_{ik}, x^*_{i\ell} > 0$.

Proof. Theorem 3.0.1 shows that there exists an optimal solution, x^*, to MCKP that is also an optimal extreme point solution to LMCKP. Since there are $n + 2$ constraints in LMCKP, it follows that x^*, since it is a basic solution, will have no more than $n + 2$ positive components. By constraints (6–15) at least one variable associated with each item $i = 1, \ldots, n$ should be strictly positive, for a total of n positive variables. The remaining two positive variables may correspond to distinct items, leading to a solution of the form (i), or to the same item, leading to a solution of the form (ii). □

As for the LKP, we denote the dual multipliers of (6–13) and (6–14) by λ and γ. Moreover, we denote the dual multipliers for the constraints (6–15) by π_i $(i = 1, \ldots, n)$. Given a dual feasible solution (λ, γ, π) to the dual of LMCKP, the complementary slackness conditions include:

$$x_{ij}(\lambda b_{ij} + \gamma s_{ij} + \pi_i - r_{ij}) = 0 \qquad j = 1, \ldots, v_i,\ i = 1, \ldots n.$$

These conditions motivate the following *forcing rule* that determines the value of some primal variables for a given dual solution:

Definition 6.3.2 (Forcing Rule)**.** *Given a dual feasible solution (λ, γ, π) to the dual of LMCKP, the primal solution to LMCKP should satisfy, for all $i = 1, \ldots, n$:*

$$r_{ij} < \lambda b_i + \gamma s_i + \pi_i \quad \Rightarrow \quad x_{ij} = 0$$

$$r_{ij} = \lambda b_i + \gamma s_i + \pi_i \quad \Rightarrow \quad x_{ij} \in [0,1].$$

By Theorem 6.3.1, any basic feasible solution to LMCKP can be characterized by either two two-way split items or one three-way split item. We will first consider the first type of basic feasible solution defined by item i split between variants j and k, and item i' split between variants j' and k'. Let $\Delta^{(1)}$ be the collection of indices (i,j,k,i',j',k') that indeed correspond to a basic solution. This, in particular, means that for all $\delta = (i,j,k,i',j',k') \in \Delta^{(1)}$ the following system of equations has a unique solution $(\lambda^\delta, \gamma^\delta, \pi_i^\delta, \pi_{i'}^\delta)$:

$$\begin{aligned} r_{ij} &= \lambda b_{ij} + \gamma s_{ij} + \pi_i \\ r_{ik} &= \lambda b_{ik} + \gamma s_{ik} + \pi_i \\ r_{i'j'} &= \lambda b_{i'j'} + \gamma s_{i'j'} + \pi_{i'} \\ r_{i'k'} &= \lambda b_{i'k'} + \gamma s_{i'k'} + \pi_{i'}. \end{aligned}$$

Note that dual feasibility requires that

$$\pi_{\hat{\imath}}^\delta = \max_{\ell=1,\ldots,v_{\hat{\imath}}} \left(r_{\hat{\imath}\ell} - \lambda^\delta b_{\hat{\imath}\ell} - \gamma^\delta s_{\hat{\imath}\ell} \right) \qquad \hat{\imath} = i, i'.$$

If this is violated, δ cannot characterize a potential optimal basic feasible solution to some LMCKP and we may disregard it. We may determine the remaining values of π^δ from dual feasibility via

$$\pi_{\hat{\imath}}^\delta = \max_{\ell=1,\ldots,v_{\hat{\imath}}} \left(r_{\hat{\imath}\ell} - \lambda^\delta b_{\hat{\imath}\ell} - \gamma^\delta s_{\hat{\imath}\ell} \right) \qquad \hat{\imath} = 1, \ldots, n;\ \hat{\imath} \neq i, i'.$$

Similarly, we may consider the second type of basic feasible solutions that is defined by an item i split between variants j, k, and ℓ. Let $\Delta^{(2)}$ be the collection of indices (i,j,k,ℓ) that indeed correspond to a basic solution. This, in particular, means that for all $\delta = (i,j,k,\ell) \in \Delta^{(2)}$ the following system of equations has a unique solution $(\lambda^\delta, \gamma^\delta, \pi_i^\delta)$:

$$r_{ij} = \lambda b_{ij} + \gamma s_{ij} + \pi_i$$

$$\begin{aligned} r_{ik} &= \lambda b_{ik} + \gamma s_{ik} + \pi_i \\ r_{i\ell} &= \lambda b_{i\ell} + \gamma s_{i\ell} + \pi_i. \end{aligned}$$

As for the first type of candidate basic solutions, dual feasibility requires that

$$\pi_i^\delta = \max_{j=1,\ldots,v_i} \left(r_{ij} - \lambda^\delta b_{ij} - \gamma^\delta s_{ij} \right).$$

If this is violated, δ cannot characterize a potential optimal basic feasible solution to some LMCKP and we may disregard it. We may again determine the remaining values of π^δ follow dual feasibility via

$$\pi_{\hat{\imath}}^\delta = \max_{j=1,\ldots,v_i} \left(r_{\hat{\imath}j} - \lambda^\delta b_{\hat{\imath}j} - \gamma^\delta s_{\hat{\imath}j} \right) \qquad \hat{\imath} = 1, \ldots, n;\ \hat{\imath} \neq i.$$

Finally, let $\Delta = \Delta^{(1)} \cup \Delta^{(2)}$ correspond to the collection of all candidate basic feasible solutions that we need to consider. Summarizing, we have shown that, given a choice of split items as dictated by Theorem 6.3.1 that may occur in a basic feasible solution to LMCKP, there exists a unique choice of a dual multipliers (λ, γ, π) that will satisfy the complementary slackness conditions related to LMCKP. This fact will allow us to restrict attention to a relatively small number of candidate solutions to MCKP (one for each potential solution split) as compared to applying the algorithm that we developed for MKP in Section 6.2 to this class of problems, in which case we would have to consider $O\left((nv)^{n-1}\right)$ candidate solutions.

6.3.2 Constructing Full Candidate Solutions

We follow a structure that is similar to the solution method for KP. Given a candidate solution structure represented by $\delta \in \Delta$, we can determine the unique dual solution $(\lambda^\delta, \gamma^\delta, \pi^\delta)$ which corresponds to the chosen split. We then define the sets

$$I_i^\delta = \{j = 1, \ldots, v_i : r_{ij} = \lambda^\delta b_{ij} + \gamma^\delta s_{ij} + \pi_i^\delta\}.$$

In order to determine a candidate optimal solution to MCKP, say x^δ, we first observe that by the Forcing Rule (Definition 6.3.2) we know that $x_{ij}^\delta = 0$ if $j \notin I_i^\delta$. To determine the remaining components of x^δ we solve the following subproblem:

$$\text{maximize} \sum_{i=1}^{n} \sum_{j \in I_i^\delta} r_{ij} x_{ij} - g \left(\sum_{i=1}^{n} \sum_{j \in I_i^\delta} s_{ij} x_{ij} \right)$$

subject to (SP$^\delta$)

$$\sum_{i=1}^{n} \sum_{j \in I_i^\delta} b_{ij} x_{ij} = B \tag{6–16}$$

$$\sum_{j \in I_i^\delta} x_{ij} = 1 \qquad i = 1, \ldots, n \tag{6–17}$$

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

By substituting $r_{ij} = \lambda^\delta b_{ij} + \gamma^\delta s_{ij} + \pi_i^\delta$ for all $j \in I_i^\delta$ the objective of this problem reduces to

$$\text{maximize} \sum_{i=1}^{n} \pi_i^\delta + \lambda^\delta B + \gamma \sum_{i=1}^{n} \sum_{j \in I_i^\delta} s_{ij} x_{ij} - g \left(\sum_{i=1}^{n} \sum_{j \in I_i^\delta} s_{ij} x_{ij} \right).$$

Note that, if for some $i = 1, \ldots, n$ we have that $I_i^\delta = \{j_i\}$ we know that $x_{ij_i} = 1$. We could thus eliminate these variables, leading to an equivalent problem to SP$^\delta$ with fewer variables by appropriately defining $\bar{R}^\delta$, $\bar{S}^\delta$, and $\bar{B}^\delta$ in a similar fashion as we did for KP and MKP.

In conclusion, we may thus solve SP$^\delta$ by solving a problem from the class MCKP with $r = \alpha s$, which we will call MCKP^0. We can solve this class of problems in a similar fashion to KP^0 but with

$$X = \left\{ x \geq 0 : \sum_{i=1}^{n} \sum_{j=1}^{v_i} b_{ij} x_{ij} = B; \sum_{j=1}^{v_i} x_{ij} = 1, \, i = 1, \ldots n \right\}$$

(see Section 6.1.3). Note that solving SP$^\delta$ thus involves solving two linear multiple-choice knapsack problems to find appropriate lower and upper bounds using (6–6) and (6–7) as

well as solving a one-dimensional optimization problem of the form (6–8). We now show that the set Δ indeed contains an optimal solution to MCKP.

Theorem 6.3.3. *The solution* $\bar{x} = x^{\bar{\delta}}$*, where*

$$\bar{\delta} = \arg\max \left\{ \sum_{i=1}^{n} \sum_{j=1}^{v_i} r_{ij} x_{ij}^{\delta} - g\left(\sum_{i=1}^{n} \sum_{j=1}^{v_i} s_{ij} x_{ij}^{\delta} \right) : \delta \in \Delta \right\}$$

is an optimal solution to MCKP.

Proof. Consider an optimal solution x^* to MCKP and consider the LMCKP defined by $S^* = s^T x^*$. Let $\hat{x}$ denote a basic optimal solution to this LMCKP (which is also optimal to MCKP) and let δ define the split in $\hat{x}$. Since $\delta \in \Delta$ and the dual solution $(\lambda^{\delta}, \gamma^{\delta}, \pi^{\delta})$ is unique, we have that x^* is a feasible solution to SP^{δ} which implies that x^{δ} is optimal to MCKP. □

6.3.3 Solution Methods and Runtime Analysis

The running time of the proposed algorithm is $O(n^2 v^4 \max\{nv, \phi\})$, which can be seen as follows. The number of ways we may split an item between two variants is $O(nv^2)$ and number of ways we may split an item between three variants is $O(nv^3)$. We must examine all pairs of the former, which means that we need to consider a total of $O(n^2 v^4)$ candidate splits. Given a choice of split items, we require $O(nv)$ time to compute the remaining vector π and determine the sets I_i^{δ}. We then must solve two continuous multiple-choice knapsack problems, which can be done in $O(\sum_{i=1}^{n} |I_i^{\delta}|) = O(nv)$ time using either the algorithm of Dyer [45] or Zemel [161]. Finally, we must solve the corresponding one-dimensional global optimization problem in $O(\phi)$ time. This implies that the runtime to determine the candidate solution for a given δ is $O(nv + \phi)$. There are $O(n^2 v^4)$ number of pairs two-way splits and $O(n^2 v^3)$ three-way splits in Δ, so this yields a straightforward algorithm with a runtime of $O(n^2 v^4 \max\{nv, \phi\})$.

It turns out, for a given split (j, k) of item i, it is only necessary to examine this split with $O(v)$ splits of item i'. The straightforward algorithm examines $O(v^2)$ splits of item

i' for a given split (j, k) of item i, but as we will soon show, many of these splits lead to infeasible dual solutions. We will ultimately show the following theorem.

Theorem 6.3.4. *There exists an algorithm to solve MCKP with a running time of* $O(n^2v^3 \max\{\log n, v, \phi\})$.

6.3.3.1 Improved Algorithm Under an Assumption

We will first develop an algorithm with a running time of $O(n^2v^3 \max\{\log n, v, \phi\})$ under a technical assumption which will arise during our analysis. We will then show that we may remove this assumption and still obtain the same running time. Through the development of the algorithm, we will also gain more insight into the number of ways that we may split a particular item i' given that we choose to split item i between variants j and k.

Our improved algorithm will examine all solutions which split item i between variants j and k consecutively. In order for item i to be split between variants j and k, we must have:

$$r_{ij} = \lambda b_{ij} + \gamma s_{ij} + \pi_i \tag{6–18}$$

$$r_{ik} = \lambda b_{ik} + \gamma s_{ik} + \pi_i \tag{6–19}$$

where the vectors $(b_{ij}, s_{ij}, 1)$ and $(b_{ik}, s_{ik}, 1)$ are assumed to be linearly independent (since otherwise this split of item i cannot be part of a basic solution). This implies that either $b_{ij} \neq b_{ik}$ or $s_{ij} \neq s_{ik}$. For this discussion, we assume that $b_{ij} \neq b_{ik}$ although an equivalent argument can be made for the case when $b_{ij} = b_{ik}$ and $s_{ij} \neq s_{ik}$. Similar to the improved algorithm for KP, we will solve for λ and π_i as functions of γ in order for Equations (6–18)-(6–19) to hold. We will denote these functions by $\lambda^{ijk}(\gamma)$ and $\pi_i^{ijk}(\gamma)$. In particular, we must have that

$$\lambda^{ijk}(\gamma) = \frac{r_{ij} - r_{ik} - \gamma(s_{ij} - s_{ik})}{b_{ij} - b_{ik}} \tag{6–20}$$

$$\pi_i^{ijk}(\gamma) = \frac{r_{ik}b_{ij} - r_{ij}b_{ik} - \gamma(s_{ik}b_{ij} - s_{ij}b_{ik})}{b_{ij} - b_{ik}} \tag{6–21}$$

which are simply linear functions of γ. For any choice of γ, it is necessary that for all $\ell = 1, \ldots, v_i$ that

$$r_{i\ell} \leq \lambda^{ijk}(\gamma) b_{i\ell} + \gamma s_{i\ell} + \pi_i^{ijk}(\gamma) \tag{6–22}$$

in order for $(\lambda^{ijk}(\gamma), \gamma, \pi_i^{ijk}(\gamma))$ to be dual feasible. We will define the set

$$D_{ijk} = \{\ell = 1, \ldots v_i : r_{i\ell} = \lambda^{ijk}(\gamma) b_{i\ell} + \gamma s_{i\ell} + \pi_i^{ijk}(\gamma) \text{ for all } \gamma\}$$

which implies that if $\ell \in D_{ijk}$ then $x_{i\ell}$ will appear as a variable in all subproblems when we split item i between variants j and k. For any variant $\ell \notin D_{ijk}$ of item i, we may determine the value γ^ℓ such that equality holds in (6–22). Since the right-hand side of (6–22) is a linear function in terms of γ, we have that either (6–22) holds for all $\gamma \geq \gamma^\ell$ or for all $\gamma \leq \gamma^\ell$. In $O(v_i)$ time, we can determine the range of γ, $[\underline{\gamma}^{ijk}, \overline{\gamma}^{ijk}]$, in order for $(\lambda^{ijk}(\gamma), \gamma, \pi_i^{ijk}(\gamma))$ to be dual feasible. Note that for all $\gamma \in (\underline{\gamma}^{ijk}, \overline{\gamma}^{ijk})$ and there exists some $\ell \notin D_{ijk}$ that $r_{i\ell} > \lambda^{ijk}(\gamma) b_{i\ell} + \gamma s_{i\ell} + \pi_i^{ijk}(\gamma)$. Therefore, we need to only consider a three-way split of item i, $\delta = (i, j, k, \ell)$, for $\ell \notin D_{ijk}$ and $\gamma^\delta = \underline{\gamma}^{ijk}$ or $\gamma^\delta = \overline{\gamma}^{ijk}$. If multiple variants $\ell, \ell' \notin D_{ijk}$ have that $\gamma^\delta = \gamma^{\delta'} = \underline{\gamma}^{ijk}$ (or $\gamma^\delta = \gamma^{\delta'} = \overline{\gamma}^{ijk}$) where $\delta = (i, j, k, \ell)$ and $\delta' = (i, j, k, \ell')$, we have that $I_i^\delta = I_i^{\delta'}$. This implies that we need to only consider two three-way splits of item i involving variants j and k, one corresponding to $\gamma^\delta = \underline{\gamma}^{ijk}$ and one corresponding to $\gamma^\delta = \overline{\gamma}^{ijk}$, which we can solve for in $O(nv + \phi)$ time.

We now turn our attention to other items i' in MCKP. We can then express the remaining dual variables $\pi_{i'}$ as a piecewise linear convex function of γ for the remaining $\pi_{i'}$ variables by noting that complementary slackness and dual feasibility require

(i) for some $j' = 1, \ldots, v_{i'}$, $r_{i'j'} = \lambda^{ijk}(\gamma) b_{i'j'} + \gamma s_{i'j'} + \pi_{i'}^{ijk}(\gamma)$; and

(ii) for all $j' = 1, \ldots, v_{i'}$, $r_{i'j'} \leq \lambda^{ijk}(\gamma) b_{i'j'} + \gamma s_{i'j'} + \pi_{i'}^{ijk}(\gamma)$.

This implies that

$$\pi_{i'}^{ijk}(\gamma) = \max_{j'=1,\ldots,v_{i'}} \left(r_{i'j'} - \lambda^{ijk}(\gamma) b_{i'j'} - \gamma s_{i'j'} \right) \tag{6–23}$$

which is a piecewise linear convex function with at most $v_{i'}$ breakpoints. We will begin our discussion under the assumption:

Assumption 6.3.5. *For a fixed item i and variants j, k, there does not exist variants j', k' of an item $i' \neq i$ such that:*

$$r_{i'j'} - \lambda^{ijk}(\gamma) b_{i'j'} - \gamma s_{i'j'} = r_{i'k'} - \lambda^{ijk}(\gamma) b_{i'k'} - \gamma s_{i'k'}$$

for all $\gamma \in \mathbb{R}$.

To construct all potentially optimal solutions where item i is split between j and k and we split item i', we need to only consider splitting item i' into variants j' and k' such that the maximum in (6–23) is attained for both j' and k' since otherwise the solution $(\lambda^\delta, \gamma^\delta, \pi^\delta)$ (where $\delta = (i, j, k, i', j', k') \in \Delta$) is not dual feasible. Since we have (for now) assumed that Assumption 6.3.5 holds, this situation can only occur at the breakpoints of the function $\pi_{i'}^{ijk}(\gamma)$. Typically, exactly two variants of item i' will achieve the maximum in (6–23). Note that it is possible that more than two variants of item i' to achieve the maximum in (6–23) at some γ'. For any choice of variants j', k' achieving the maximum, we have that $I_{i'}^{\delta}$ (where $\delta = (i, j, k, i', j', k')$) is exactly the set of variants achieving the maximum in (6–23) for $\gamma^\delta = \gamma'$. This means that for any choice of variants j', k' achieving the maximum in (6–23), the subproblems that arise when solving for x^δ (with $\delta = (i, j, k, i', j', k')$) will be the same. Therefore, we need to only consider a single pair of variants of item i' at any breakpoint of $\pi_{i'}^{ijk}(\gamma)$. This implies that we need to consider at most $v_{i'}$ ways (each corresponding to a breakpoint of $\pi_{i'}^{ijk}(\gamma)$) to split item i'. We can determine these breakpoints in $O(v^2)$ time by first sorting the slopes of the $v_{i'}$ linear functions defining $\pi_{i'}^{ijk}(\gamma)$ and then determining the intervals where each linear function obtains the maximum.

We then sort *all* the values of γ corresponding to a breakpoint of any of the functions $\pi_{i'}^{ijk}(\gamma)$. Note that if any of these values, γ^δ ($\delta = (i, j, k, i', j', k')$) , are outside the range $[\underline{\gamma}^{ijk}, \overline{\gamma}^{ijk}]$, then we do not need to consider the split δ since $(\lambda^\delta, \gamma^\delta, \pi^\delta)$ is not dual feasible.

We will then proceed in a similar fashion as we did in the case of KP. We will consider the values of γ in the range in non-decreasing order. We calculate the sets I_i^δ for the split $\delta = (i, j, k, i', j', k')$ which corresponds to the smallest value of γ in the range. When we consider the next value of γ, exactly two things will occur. First, the forcing rule will dictate that $x_{i'j'}$ or $x_{i'k'}$ must equal zero. Second, there will be some other item $\hat{\imath}$ which had that (for the first value of γ) $|I_{\hat{\imath}}^\delta| = 1$ but now has that $|I_{\hat{\imath}}^\delta| > 1$. Note, though, as we vary γ, each variant of an item i' can enter and leave the set $I_{i'}^\delta$ at most once, corresponding to the smallest value of γ such that $\pi_{i'}^{ijk}(\gamma) = r_{i'j'} - \lambda^{ijk}(\gamma)b_{i'j'} - \gamma s_{i'j'}$ and the largest value of γ such that $\pi_{i'}^{ijk}(\gamma) = r_{i'j'} - \lambda^{ijk}(\gamma)b_{i'j'} - \gamma s_{i'j'}$. Therefore, as we vary γ, the time spent updating the sets I_i^δ (and hence updating the parameters of the subproblems) will be equal to $O(nv + nv) = O(nv)$.

Currently, we have analyzed the time required to compute and update the parameters of the subproblems that need to be solved involving variants j, k of item i. We now turn our attention to determining the time required to solve the subproblems. Recall that in our algorithm to solve KP, we had to pay attention to items which appeared in all subproblems involving a given item (the sets D_i). These sets played an important role in the complexity of the algorithm. We will pay attention to a similar set for a pair of variants of an item. For a particular choice of variants $j, k = 1, \ldots, v_i$, recall the set:

$$D_{ijk} = \{\ell : r_{i\ell} = \lambda^{ijk}(\gamma) + \gamma s_{i\ell} + \pi_i^{ijk}(\gamma) \text{ for all } \gamma\}.$$

Therefore, $x_{i\ell}$ will appear in all of the linear multiple choice knapsack problems during the course of the procedure to determine all solutions where we split item i between variants j and k. The definition of D_{ijk} is very similar to the definition of D_i in the case of KP. The procedure to solve for all splits of item i between variants j and k requires: (i) $O(v)$ time to determine the range $[\underline{\gamma}^{ijk}, \bar{\gamma}^{ijk}]$; (ii) $O(nv^2)$ time spent determining the breakpoints of the functions $\pi_{i'}^{ijk}(\gamma)$; (iii) $O(nv \log nv)$ time spent sorting all the values γ corresponding to breakpoints of the functions $\pi_{i'}^{ijk}(\gamma)$; (iv) $O(nv)$ time spent determining the initial sets

I_i^δ and the time entering/removing items from these sets between consecutive subproblems and changing the parameters of the subproblem; (v) $O(nv + nv|D_{ijk}| + nv\phi)$ time required to solve the subproblems. Each variant j' of an item i' will appear as a variable, i.e. $j' \in I_i^\delta$ and $|I_i^\delta| \neq 1$, at most twice and therefore will only add a constant amount of time in solving the linear multiple choice knapsack problems arising in the subproblem. We need to solve $O(nv)$ one-dimensional optimization problems. This implies that the total time required to solve for solutions where item i is split between variants j and k is $O(nv \max\{\log n, v, |D_{ijk}|, \phi\})$.

The sets D_{ijk} play an important role in the reduction of computational effort. The sets D_{ijk} partition the pairs of variants $j, k = 1, \ldots, v_i$ into mutually exclusive sets. In other words, if $\ell \in D_{ijk}$, then $k \in D_{ij\ell}$ and $j \in D_{ik\ell}$. Put differently, the sets D_{ijk} effectively form a partition of the set of pairs of variants of i, which can be indexed by a set $I_i \subseteq \{(j,k) : j, k = 1, \ldots v_i, j < k\}$ with the property that:

$$D_{ijk} \cap D_{i\ell p} = \emptyset \text{ for all } (j,k), (\ell,p) \in I_i \text{ and } \bigcup_{(j,k)\in I_i} D_{ijk} = \{(j,k) : j, k = 1, \ldots v_i, j < k\}.$$

Note that once we have solved for all splits of item i between variants j, k, it is not necessary to solve for all splits of item i between variants j, ℓ if $\ell \in D_{ijk}$. This will lead to a significant savings in computational effort in determining the solutions corresponding to all splits involving item i. In particular, the time required to solve for all splits involving item i is:

$$O\left(\sum_{(j,k)\in I_i} (nv \log n + nv^2 + nv|D_{ijk}| + nv\phi)\right) = O(nv^3 \max\{\log n, v, \phi\}).$$

Therefore, under Assumption 6.3.5, we can determine an optimal solution to MCKP in $O(n^2 v^3 \max\{\log n, v, \phi\})$ time.

6.3.3.2 Generalizing the Improved Algorithm

Next, we will analyze the improved algorithm for MCKP when Assumption 6.3.5 is violated. We will show that the running time is still $O(n^2 v^3 \max\{\log n, v, \phi\})$. In

particular, suppose that for a given split of item i between variants j and k, we have that for all γ:

$$r_{i'j'} - \lambda^{ijk}(\gamma)b_{i'j'} - \gamma s_{i'j'} = r_{i'k'} - \lambda^{ijk}(\gamma)b_{i'k'} - \gamma s_{i'k'}.$$

We first note that $b_{i'j'} \neq b_{i'k'}$, since the only point that equality would hold is $\gamma = (r_{i'j'} - r_{i'k'})/(s_{i'j'} - s_{i'k'})$. Therefore, rearranging the terms, we obtain:

$$\lambda^{i'j'k'}(\gamma) = \frac{r_{i'j'} - r_{i'k'} - \gamma(s_{i'j'} - s_{i'k'})}{b_{i'j'} - b_{i'k'}} = \lambda^{ijk}(\gamma)$$

which implies that for all $\hat{\imath} \neq i, i'$ that $\pi^{ijk}_{\hat{\imath}}(\gamma) = \pi^{i'j'k'}_{\hat{\imath}}(\gamma)$. It is easy to show (by a substitution and similar rearrangement) that for all γ:

$$r_{ij} - \lambda^{i'j'k'}(\gamma)b_{ij} - \gamma s_{ij} = r_{ik} - \lambda^{i'j'k'}(\gamma)b_{ij} - \gamma s_{ij}$$

Now consider the range, $[\underline{\gamma}, \bar{\gamma}]$, where the variants j', k' achieve the maximum in (6–23) of $\pi^{ijk}_{i'}(\gamma)$. Consider some $\delta = (i, j, k, \hat{\imath}, \hat{\jmath}, \hat{k})$ such that $\gamma^\delta \in (\underline{\gamma}, \bar{\gamma})$ and $(\lambda^\delta, \gamma^\delta, \pi^\delta)$ is dual feasible. This implies that we will need to solve the subproblem SP^δ. We have that $I^\delta_{i'} = \{j, k\}$, so $x_{i'j'}$ and $x_{i'k'}$ appear as variables in SP^δ. Since γ^δ is not a breakpoint of $\pi^{ijk}_{i'}(\gamma)$, these variables are unaccounted for in our complexity analysis above. Although the appearance of these variables require additional computational effort in solving SP^δ, we now detail that we save at least the same amount when considering splitting item i' between j' and k'. This would imply that the complexity of our algorithm is still $O(n^2v^3 \max\{\log n, v, \phi\})$. First, since $\pi^{ijk}_{\hat{\imath}}(\gamma) = \pi^{i'j'k'}_{\hat{\imath}}(\gamma)$, we have that γ^δ will be considered when we split item i' between variants j' and k'. Also, when we consider $\delta' = (i', j', k', \hat{\imath}, \hat{\jmath}, \hat{k})$, we will have that $I^{\delta'}_{\bar{\imath}} = I^\delta_{\bar{\imath}}$ for all $\bar{\imath} \neq i, i'$. Further, since $(\lambda^\delta, \gamma^\delta, \pi^\delta)$ is dual feasible, we have that for all $\ell = 1, \dots v_i$:

$$r_{ij} - \lambda^{ijk}(\gamma)b_{ij} - \gamma s_{ij} = \pi^{ijk}_i(\gamma^\delta) \geq r_{il} - \lambda^{ijk}(\gamma)b_{il} - \gamma s_{il}$$

which implies that $r_{ij} - \lambda^{i'j'k'}(\gamma)b_{ij} - \gamma s_{ij}$ and $r_{ik} - \lambda^{i'j'k'}(\gamma)b_{ik} - \gamma s_{ik}$ achieve the maximum in the definition of $\pi^{i'j'k'}_i(\gamma^{\delta'})$. This implies that both j, k are in the set I^δ_i. For any variant

$\ell \neq j, k$ of item i such that $\ell \in I_i^\delta$, it is easy to see that $\ell \in I_i^{\delta'}$. Since $\gamma^\delta \in (\underline{\gamma}, \bar{\gamma})$, it follows that $I_{i'}^\delta = \{j', k'\}$. This means that for any variant $l' \neq j', k'$ of item i' that:

$$\pi_{i'}^{i'j'k'}(\gamma) = r_{i'j'} - \lambda^{i'j'k'}(\gamma) b_{i'j'} - \gamma s_{i'j'} > r_{i'\ell'} - \lambda^{i'j'k'}(\gamma) b_{i'\ell'} - \gamma s_{i'l'}$$

and therefore $I_{i'}^{\delta'} = I_{i'}^\delta$. This implies that the subproblems SP^δ and $\text{SP}^{\delta'}$ are identical. Therefore, we need to only solve it once. Since $x_{i\hat{j}}$ and $x_{i\hat{k}}$ appear as variables in these subproblems, the computational savings of needing only to solve a single one of the subproblems exceeds the additional computational effort to account for $x_{i'j'}$ and $x_{i'k'}$ appearing as variables in SP^δ. This can be seen because in our complexity analysis of determining all splits involving variants j', k' of item i', we include the additional time required to solve the linear multiple-choice knapsack problems when $x_{i\hat{j}}$ and $x_{i\hat{k}}$ appear in $\text{SP}^{\delta'}$ but we do not actually solve these problems, since SP^δ and $\text{SP}^{\delta'}$ are equivalent.

Therefore, the running time of the improved algorithm remains the same, even if the instance of MCKP violates Assumption 6.3.5. This implies the following result:

Theorem 6.3.6. *We can find an optimal solution to MCKP in* $O(n^2 v^3 \max\{\log n, v, \phi\})$ *time*

6.4 Computational Testing

One of the main advantages of the proposed algorithms to solve KP and MCKP is that they require the solution of only a low-order polynomial number of one-dimensional global optimization problems *regardless* of the structure of the function g. In this section, we will illustrate the behavior of the algorithm by testing its performance and comparing it to the global optimization solver BARON. We have chosen to use this solver since it represents the state-of-the art in commercial global optimization software; as Neumaier et al. [108] said: "*Among the currently available global solvers, BARON is the fastest and most robust one*". For our experiments, we have used the BARON solver through its GAMS interface.

The test problems that we examine are motivated by market selection problems in supply chain optimization. In such problems, a company offers a product and should determine which fraction of demand for the product to satisfy in each of n available markets. The company receives a revenue of r_i if it chooses to serve the entire demand, s_i, in market i ($i = 1, \ldots, n$). The function g reflects the procurement cost of the total demand for the product. In addition, to ensure that the demand in market i indeed materializes, we assume that the company should invest an amount b_i (in the form of, e.g., advertising, market research, etc.), and has a budget of B available for this use. We let x_i denote the level of entry into market i ($i = 1, \ldots, n$). Then the company wishes to maximize its profits, i.e., total revenue $r^\top x$ less the procurement cost $g\left(s^\top x\right)$ subject to the constraint that the total marketing expenditures $b^\top x$ do not exceed B. the capital budget. In order to formulate this problem as an instance of the class KP, we introduce a slack variable x_{n+1} with $r_i = s_i = 0$ and $b_i = B$.

In addition, we considered extensions of such market selection problems in which there are several options available in each market, possibly arising due to different product designs or different advertising options. For example, consider a market with two forms of advertising media: television and newspaper. Given that a company chooses to enter the market, it can choose to advertise just on television, just in the newspaper, or both. In order to formulate this extension as an instance in the class MCKP, we introduce an additional market, say $n + 1$, with variants $v = 1, 2$ where $r_{n+1,1} = s_{n+1,1} = b_{n+1,1} = 0$ and $r_{n+1,2} = s_{n+1,2} = 0$ and $b_{n+1,2} = B$. Note that, in practice, it can be expected that the parameters corresponding to each of the variants are correlated, in the sense that a higher revenue variant should require more resources. For example, advertising in both forms of media will lead to higher revenue and higher demands but will require more capital. In other words, given a market i with choices (or variants) $v = 1, \ldots, v_i$, it is reasonable to expect that we can order the choices in such a way that $r_{ij} > r_{ik}$, $s_{ij} > s_{ik}$, and $b_{ij} > b_{ik}$ for any pairs j, k with $j < k$.

We considered a total of three forms for the procurement cost function g:

1. The first problem class that we consider is motivated by economies of scale in production planning, i.e., we assume that the total demand is produced, and the production function is a concave function of the production quantity. In particular, we consider the function

$$g(S) = \sqrt{CS}$$

for some constant $C > 0$, where S denotes the total demand to be satisfied. Note that this model arises if the production costs arise from an Economic Order Quantity model (see Harris [76] and Geunes et al. [67]), where $C = 2Kh$ if K is a fixed production cost and h a holding cost rate. Interestingly though, the same functional form of g arises in the presence of stochastic demands and risk pooling under newsvendor costs (see Taaffe et al. [149]); in this case, however, s_i represents the variance of demand in market i. Since C is a scaling parameter only, we have in our tests simply used $C = 1$.

2. The second problem class that we consider corresponds to situations in which there are again economies of scale in procurement costs, but this form only holds up to some production level β, which may represent a capacity. Beyond these levels, we assume that the company is able to outsource additional units of the product, but the outsourcing costs are a convex function of the amount outsourced. To model this situation, we have considered the consider the function

$$g(S) = \alpha (S - \beta)^3 + \beta^3$$

for some constants $\alpha, \beta > 0$. This function is concave on the interval $(-\infty, \beta]$ and convex on the interval $[\beta, \infty)$. In our tests, we have used the values $\alpha = 1/n^2$ and $\beta = n$, where the particular dependence on n ensures proper scaling among problem dimensions.

3. The form of g in the second problem class was chosen for both its general shape (concave at small values of the argument and convex at large values) as well as since it can be represented by a single expression. This is particularly important for BARON when used through its GAMS interface. However, from a practical point of view a more realistic model would have the concave part of the same form as in the first problem class, followed by a convex cost component for outsourcing. We therefore also considered the following function:

$$g(S) = \begin{cases} \sqrt{CS} & \text{if } 0 \leq S \leq \beta \\ (S-\beta)^2 + \sqrt{C\beta} & \text{if } S \geq \beta. \end{cases}$$

This function is continuous but not differentiable at β, and cannot be implemented with GAMS/BARON directly. We therefore reformulated the problem using a binary variable indicating whether or not S exceeds β, yielding a mixed-integer nonlinear problem to be solved by BARON. As in the first and second problem classes, we have set $C = 1$ and $\beta = n$.

We start our computational study with the problem class KP. For each of the three problem classes, we considered problems with n equal to 1,000, 5,000, 10,000, and 20,000. For each problem class and each value of n, we randomly generated 10 instances of KP as follows:

- The revenue r_i associated with market i is generated uniformly in the interval $(0, 50)$;
- the demand s_i in market i is generated uniformly in the interval $(0, 10)$;
- the expenditure b_i in market i is generated uniformly in $(0, 10)$.

In addition, we solved each problem instance with two different values for the budget: one corresponding to a tight budget ($B = .1 \times n \times 10 = n$) and one corresponding to a generous budget ($B = .4 \times n \times 10 = 4n$). Note that the budget is appropriately scaled by the number of markets n in order to obtain comparable instances among different problem dimensions.

Table 6-1 shows the results of the computational tests for KP, where the computation times are all in seconds on a Dell Power Edge 2600 with two Pentium 4 3.2 Ghz processors, 6 gigabytes of memory, and three Ultra320 15K RPM SCSI drives. The table clearly illustrates the insensitivity of the algorithm presented in this chapter to the functional form of g. In addition, it shows that the algorithm performs competitively with BARON for the easiest problem class, but as the form of the function g becomes more complex the results very clearly show the power of our approach. In fact, for some instances BARON was not able to solve the problem (or verify that the solution found was globally optimal) within the time limit of 3600 seconds (one hour) that we set. In particular, whenever the computation time has a superscript, the corresponding number represents the number of instances for which BARON was not successful (out of a total of 10 instances). In these cases, the times provided are averages over the *successful instances only*.

Table 6-1. Comparison of running times obtained with the algorithm for KP and BARON.

		Problem class 1		Problem class 2		Problem class 3	
n	B	this chapter	BARON	this chapter	BARON	this chapter	BARON
1,000	n	.39	.24	.60	5.74	.39	3.98
	$4n$	.39	.20	.49	3.73^{2}	.39	4.20
5,000	n	11.47	5.25	11.06	266.20	10.90	112.69
	$4n$	11.43	4.90	10.89	203.02^{1}	10.81	114.89
10,000	n	49.39	36.06	56.31	911.90	46.79	470.89^{1}
	$4n$	46.85	39.83	48.77	540.30	46.87	467.48^{2}
20,000	n	224.58	470.13	220.16	1914.85	213.89	2209.99
	$4n$	379.12	408.24	210.73	2188.88	207.92	1951.00

Next, we studied the performance of the improved algorithm for MCKP to BARON. To facilitate a comparison with the results obtained for KP as well as among different values of v, we chose, for each value of v, the number of markets n so that $n(v-1)$ is approximately equal to a corresponding value of n used for KP (where we note that the KP implicitly contains $v = 2$ variants). For example, for $n(v-1) = 10{,}000$ and $v = 4$, we set $n = 3{,}333$. In addition, again for consistency with the KP instances,

we let the v^{th} variant of each market correspond to not selecting the market at all, i.e., $r_{iv} = s_{iv} = b_{iv} = 0$. To obtain values of the parameters for each market for all other variants, we first independently generated $v - 1$ revenue, demand, and expenditure values from the same distributions that were used for the KP instances. We then associated the highest values in each parameter type with the first variant, the second highest values with the second variant, etc. Since the results obtained for the KP indicate that the performance of both algorithms is insensitive to the capacity, we only used a single, intermediate, capacity value of $B = .25 \times n \times 10 = 2.5n$ for MCKP.

Table 6-2 shows the results of the computational tests for MCKP. Although the performance of both approaches is still comparable for the easiest problem class, BARON was not able to solve even a single problem instance in the two harder classes with 5,000 variables and $v = 3$ or 10,000 variables or more within the CPU time limit of one hour. This is in stark contrast to the improved algorithm for MCKP, which was able to solve the problem instances of all problem classes and all dimensions tested in a very reasonable amount of CPU time.

Table 6-2. Comparison of running times obtained with the improved algorithm for MCKP and BARON.

		Problem class 1		Problem class 2		Problem class 3	
$n(v-1)$	v	this chapter	BARON	this chapter	BARON	this chapter	BARON
1,000	3	.82	.81	.99	150.99	.98	191.86
	4	.48	.77	.38	61.41	.20	38.21
5,000	3	24.05	23.98	25.41	—	20.85	—
	4	14.27	19.49	19.39	1212.14	5.34	1872.54
10,000	3	127.21	109.88	106.28	—	41.79	—
	4	73.26	146.66	69.30	—	25.00	—
20,000	3	195.21	253.27	469.45	—	185.91	—
	4	95.77	215.89	213.99	—	94.51	—

6.5 Chapter Summary and Future Research Directions

In this chapter, we have developed efficient algorithms to solve important classes of nonlinear knapsack, multi-knapsack, and multiple-choice knapsack problems. The algorithms developed are based on the analysis of a family of linear programs which are

closely related to the problems. Complementary slackness conditions and duality theory of linear programming are applied in a novel way to characterize a set of solutions which contains an optimal solution.

Rather than approach these classes of problems as single multi-dimensional global optimization problems, we solve the problems through a sequence of one-dimensional global optimization problems. In the cases of the knapsack and multiple-choice knapsack constraint systems, our algorithm requires the solution of only a polynomial number of one-dimensional problems. In the future, it would be interesting to identify other special types of problem structures that only require solving a polynomial number of one-dimensional optimization problems.

The classes of problems studied in this chapter have applications in supply chain optimization and financial engineering. In many of these applications, an integer version of the problems (with "all-or-nothing" choices) may also be of interest. In these cases, our algorithms can be used to provide an upper bounds on the objective function value at each node in a branch-and-bound tree. In a straightforward branch-and-bound scheme, we may need to apply our algorithm at many nodes. Therefore, it may be worthwhile to use ideas from this chapter to develop tailored branch-and-bound algorithms to solve integer versions of the problems studied in this chapter.

CHAPTER 7
INTEGRATING FACILITY LOCATION AND PRODUCTION PLANNING DECISIONS

Traditional facility location problems focus on determining a set of open facilities and assigning each customer to an open facility, where the goal is to minimize the sum of facility opening costs and connection costs of the customers to the facilities. When, as is often the case, the company needs to meet the demand of the customers in future time periods through production and inventory decisions at its facilities, using a traditional facility location model may be inappropriate since it cannot accurately represent all incurred costs. In particular, the connection costs often represent transportation costs only, and even if they attempt to capture production and inventory holding costs, this can only be done at a very coarse level. Therefore, using traditional facility location models may lead to very (and unnecessarily) high production and inventory costs. In this chapter, we introduce a new class of integrated facility location and production planning problems that generalizes traditional facility location problems by taking into account the demands of each customer in future time periods. Our goal is then to minimize the sum of facility opening costs, connection costs, and production/inventory costs.

The class of problems studied in Chapters 8 and 9 can be described as follows. We are given a set of n customers and T time periods where the demand of customer j in period t is given by d_{jt} ($j = 1, \ldots, n$, $t = 1, \ldots, T$). It will be convenient to also define the cumulative demand of customer j as $d_j = \sum_{t=1}^{T} d_{jt}$. We wish to assign a customer j to an open facility i and meet the demand of the customer through production and inventory decisions at the facility. There is a connection cost, c_{ij}, associated with facility i and customer j, which is expressed as a cost per unit of demand. Each facility has an opening cost of f_i, which we must pay if we assign any customers to the facility. Each facility i has a concave function representing the cost of producing p units in time period t, $P_{it}(p)$, and a concave function representing the cost of holding I units in time period t, $H_{it}(I)$. Our *uncapacitated facility location and production planning problem* (UFLPP) (where we have

chosen to drop a P in the acronym for the sake of exposition) can be formulated as:

$$\text{minimize } \sum_{i=1}^{m}\left(f_i y_i + \sum_{t=1}^{T}(P_{it}(p_{it}) + H_{it}(I_{it}))\right) + \sum_{i=1}^{m}\sum_{j=1}^{n} d_j c_{ij} x_{ij}$$

subject to

$$\begin{aligned}
\sum_{i=1}^{m} x_{ij} &= 1 && \text{for } j = 1,\ldots,n && (7\text{–}1)\\
x_{ij} &\leq y_i && \text{for } i = 1,\ldots,m,\ j = 1,\ldots,n && (7\text{–}2)\\
x_{ij} &\in \{0,1\} && \text{for } i = 1,\ldots,m,\ j = 1,\ldots,n && (7\text{–}3)\\
y_i &\in \{0,1\} && \text{for } i = 1,\ldots,m && (7\text{–}4)\\
I_{i,t-1} + p_{it} &= \sum_{j=1}^{n} d_{jt} x_{ij} + I_{it} && \text{for } i = 1,\ldots,m,\ t = 1,\ldots,T && (7\text{–}5)\\
I_{i0} &= 0 && \text{for } i = 1,\ldots,m && (7\text{–}6)\\
p_{it}, I_{it} &\geq 0 && \text{for } i = 1,\ldots,m,\ t = 1,\ldots,T. && (7\text{–}7)
\end{aligned}$$

Constraints (7–1)-(7–4) are traditional facility location problems and constraints (7–5)-(7–7) are production planning constraints at each facility to ensure that, in each time period, we meet the demand of all customers assigned to the facility. If we know the set of customers assigned to a given facility, then we simply need to solve an uncapacitated production planning problem at that facility. This production planning problem is a generalization of the classic economic lot-sizing problem (see Wagner and Whitin [157]) where concave production and inventory cost functions have replaced fixed-charge plus linear production costs and linear inventory costs. The UFLPP fits into the SCND framework by defining the facility cost function, $H_i(x_{i\cdot})$ to be zero if $x_{i\cdot}$ is the empty set and otherwise set it equal to the optimal solution value of the production planning problem

$$\text{minimize } f_i + \sum_{t=1}^{T}(P_{it}(p_{it}) + H_{it}(I_{it}))$$

subject to

$$\begin{aligned} I_{i,t-1} + p_{it} &= \sum_{j=1}^{n} d_{jt} x_{ij} + I_{it} && \text{for } i = 1, \ldots, m,\ t = 1, \ldots, T \\ I_{i0} &= 0 && \text{for } i = 1, \ldots, m \\ p_{it}, I_{it} &\geq 0 && \text{for } i = 1, \ldots, m,\ t = 1, \ldots, T. \end{aligned}$$

Note that in the UFLPP, we assign each customer to exactly one facility and that facility is required for meeting the demand of the customer *in each time period* over the horizon. However, in certain situations, it may be more appropriate that the demand of each customer in each time period is met by a single facility rather than the demand of the customer over the entire horizon is met by a single facility. In this case, we can formulate a similar problem to the UFLPP where we define binary variable x_{ijt} as the decision of serving customer j in time period t with production at facility i. This problem, which we refer to as the *uncapacitated facility location and production planning problem with dynamic assignments* (UFLPP-DA) can be formulated as

$$\text{minimize } \sum_{i=1}^{m} \left(f_i y_i + \sum_{t=1}^{T} (P_{it}(p_{it}) + H_{it}(I_{it})) \right) + \sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{t=1}^{T} d_{jt} c_{ij} x_{ijt}$$

subject to

$$\begin{aligned} \sum_{i=1}^{m} x_{ijt} &= 1 && \text{for } j = 1, \ldots, n,\ t = 1, \ldots, T && (7\text{–}8) \\ x_{ijt} &\leq y_i && \text{for } i = 1, \ldots, m,\ j = 1, \ldots, n,\ t = 1, \ldots, T && (7\text{–}9) \\ x_{ijt} &\in \{0, 1\} && \text{for } i = 1, \ldots, m,\ j = 1, \ldots, n,\ t = 1, \ldots, T && (7\text{–}10) \\ y_i &\in \{0, 1\} && \text{for } i = 1, \ldots, m && (7\text{–}11) \\ I_{i,t-1} + p_{it} &= \sum_{j=1}^{n} d_{jt} x_{ijt} + I_{it} && \text{for } i = 1, \ldots, m,\ t = 1, \ldots, T && (7\text{–}12) \\ I_{i0} &= 0 && \text{for } i = 1, \ldots, m && (7\text{–}13) \\ p_{it}, I_{it} &\geq 0 && \text{for } i = 1, \ldots, m,\ t = 1, \ldots, T. && (7\text{–}14) \end{aligned}$$

We note that the UFLPP-DA is somewhat related to the multi-period single-sourcing problem (see, for example, Romeijn and Romero Morales [125]). The main differences between the two problems are: (i) the absence of production capacities in the UFLPP-DA, (ii) the presence of facility opening decisions in the UFLPP-DA, and (iii) the presence of concave production and inventory costs in the UFLPP-DA rather than linear production and inventory costs in the multi-period single-sourcing problem.

It is not difficult to see that any instance of the UFLPP-DA can be converted to an equivalent instance of the UFLPP. In particular, rather than view customer j as a single customer with demand stream $(d_{j1}, d_{j2}, \ldots, d_{jT})$ in the UFLPP-DA, we view it as a set of customers, j_t for $t = 1, \ldots, T$, where the demand stream of customer j_t being given by $d_{j_t t'} = 0$ if $t' \neq t$ and $d_{j_t t} = d_{jt}$ if $t' = t$ in the UFLPP. This means that we can convert the UFLPP-DA to a UFLPP with $O(nT)$ customers. Further, note that if we are given an instance of the UFLPP such that every customer has exactly one non-zero demand period, this problem can be converted to an equivalent instance of the UFLPP-DA.

A popular stream of work for facility location problems is to develop approximation results for them. In Chapter 8, we examine approximation results for the UFLPP and the UFLPP-DA. We show that both these problems are as hard as the set cover problem and, therefore, it is unlikely that there exists constant factor approximation algorithms for the general problem. Therefore, it is appropriate to focus on approximation algorithms for special cases of the problem. These special cases come in two forms: (i) specialize the production and inventory cost structure and (ii) specialize the demand pattern of the customers. We show that for several production/cost structures both the UFLPP and the UFLPP-DA can be approximated within a constant factor. Further, we show that a special class of the UFLPP gives rise to a class of metric uncapacitated facility location problems where the facility cost function is a concave function in the amount of demand assigned to the facility. We develop a greedy algorithm for this problem with an approximation guarantee of 1.61. We then use the greedy algorithm together with the idea

of cost-scaling to provide an algorithm for this class of problems with an approximation guarantee of 1.52. We also discuss a new class of facility location problems that arise from a special class of the UFLPP-DA. In this new class of facility location problems, each customer requests a set of services from the facility and we must install at least one of these services at the facility which the customer is assigned.

In Chapter 9, we focus on issues around developing exact algorithms to solve the UFLPP and the UFLPP-DA. For the UFLPP, we show that for the special case when the production and inventory costs are that of an economic lot-sizing problem, reformulating the problems using the plant-location formulation of the lot-sizing problem (see Krarup and Bilde [92]) yields a tighter relaxation. We also show that the set-partitioning formulation of the problem (see Section 2.1 for a discussion) yields an even tighter relaxation on the problem. Therefore, we can expect that a branch and price algorithm will perform well on this instance of the UFLPP. For the UFLPP with general concave production and inventory costs, we discuss that, in general, we cannot determine if the relaxation of the set-partitioning formulation is tighter or weaker than the continuous relaxation of the UFLPP. This does not necessarily mean that the branch and price algorithm will be ineffective; there are many examples in the literature (Shen et al. [141], Shu et al. [144], and Huang et al. [85]) where we cannot say anything about the tightness of the set-partitioning formulation yet the branch and price algorithm is still highly effective. In these examples, one of the more important elements in the branch and price algorithm is the ability to effectively solve the pricing problem. The pricing problem that arises from the UFLPP is an interesting production planning problem with customer selection. Although it was recently shown that even with economic lot-sizing costs this problem is NP-hard, we discuss some polynomially-solvable subclasses of it.

CHAPTER 8
APPROXIMATION RESULTS FOR INTEGRATED FACILITY LOCATION AND PRODUCTION PLANNING PROBLEMS

In this chapter, we will focus on developing approximation results for the UFLPP and the UFLPP-DA. In particular, we say that an algorithm is an α-approximation algorithm for a minimization problem if the algorithm runs in polynomial time and returns a solution to the problem with cost no more than α times the optimal cost. We are very interested in developing constant factor approximation algorithms for the UFLPP and the UFLPP-DA, i.e., α does not depend on the size of the problem. Therefore, for the remainder of the chapter, we assume that the connection costs are symmetric, i.e. the cost of shipping from facility i to customer j is equal to the cost of shipping from customer j to facility i, and satisfy the triangle inequality, i.e. for any facilities i, i' and customers j, j', $c_{ij} \leq c_{i'j} + c_{i'j'} + c_{ij'}$. Although these assumptions on the connection costs are restrictive, they are common in developing constant factor approximation algorithms for facility location problems. In fact, for the non-metric UFLP, there cannot exist a constant factor approximation algorithm unless $NP \subseteq TIME[O(n^{\log \log n})]$.

In the UFLPP (and the UFLPP-DA), we have also assumed that production and inventory variables are uncapacitated. If the production variables are capacitated at each facility, it becomes NP-hard to even determine if the UFLPP (and the UFLPP-DA) has a feasible solution. This can be seen since if $T = 1$, then the UFLPP (and the UFLPP-DA) is a capacitated facility location problem with unsplittable demands. It is relatively straightforward to show that the Partition problem has a solution if and only if an associated capacitated facility location problem with unsplittable demands has a feasible solution. Therefore, a study of approximation algorithms for the UFLPP (and the UFLPP-DA) should either focus on models where the production and inventory variables are uncapacitated or constraints (7–3) (or constraints (7–10)) are relaxed to allow for splittable demands. In this chapter, we focus on the former class of problems.

In Section 8.1 of this chapter we show that the UFLPP is as hard as the set cover problem and conclude that, in general, it is highly unlikely that a constant factor approximation algorithm exists. Therefore, we will focus on approximating special cases of the UFLPP. The cases that we will study fall into two categories: (i) specializing the production and inventory cost structure at the facilities (Section 8.1); and (ii) specializing the demand pattern of the customers (Section 8.2). For problems belonging to (i), we will offer reductions from the UFLPP to previously studied variants of the metric UFLP. For the problems in (ii), we will show that they belong to a special class of facility location problems with facility cost functions that are concave functions of the amount of demand assigned to the facility. We will develop a 1.52-approximation algorithm for this concave cost facility location problem (CCFLP). In Section 8.3, we discuss approximation results for the UFLPP-DA. We show that the the UFLPP-DA is as hard as the set cover problem and develop analogous results of Section 8.1 for problems with specialized cost structures. We also discuss a new class of facility location problems that arise from a special case of the UFLPP-DA. We conclude the chapter in Section 8.4 with a summary and some future research directions.

8.1 Approximating the UFLPP with General Demands

Prior to developing an approximation algorithm for the UFLPP, it is necessary to determine the complexity of approximating the UFLPP. We begin with a result that shows that the UFLPP is as hard as the set cover problem. The definition of the Set Cover problem is: Given a collection of items $1, \ldots, n$ and a collection of sets $S_i \subseteq \{1, \ldots, n\}$ with associated cost f_i for $i = 1, \ldots, m$, determine a minimum cost set $\mathcal{S} \subseteq \{S_i : i = 1, \ldots, m\}$ such that for each $j \in \{1, \ldots, n\}$ there exists $S_i \in \mathcal{S}$ such that $j \in S_i$.

Theorem 8.1.1. *If there exists an α-approximation algorithm for the UFLPP, then there also exists an α-approximation algorithm for the set cover problem.*

Proof. Consider an instance of the set cover problem, and define an instance of the UFLPP as follows.

- The set of customers is $\{1, \ldots, n\}$ and the set of facilities is $\{1, \ldots, m\}$.
- Set $c_{ij} = 0$ for all facility/customer pairs (i, j) and the facility opening costs to f_i.
- Set the number of time periods to be $T = n$ and set the demand of customer j to be $d_{jt} = 0$ if $j \neq t$ and $d_{jt} = 1$ if $j = t$.
- Set the inventory cost function for each time period at each facility to be $H_{it}(I_{it}) = I_{it} \cdot \max_{i=1,\ldots,m}(f_i + 2)$.
- Define the production functions as $P_{it}(p_{it}) = 0$ if $t = j \in S_i$ and $P_{it}(p_{it}) = p_{it} \cdot \max_{i=1,\ldots,m}(f_i + 1)$ if $t = j \notin S_i$.

The inventory holding costs ensure that it is most cost-effective to satisfy demand in period t with production in period t. This implies that if we assign customer j to facility i, it will (optimally) cost $P_{it}(1)$. It is easy to see that it is always cheaper to assign customer j to a facility i such that $j \in S_i$ than to assign customer j to a facility i where $j \notin S_i$. Therefore, an optimal set of open facilities in the UFLPP will be a minimum cost cover. Now if we have an α-approximation algorithm for the UFLPP, we can easily convert the solution returned by the algorithm to a feasible cover with a cost no worse than the returned solution. This then would yield an α-approximation algorithm for the set cover problem. □

Since Feige [51] showed a hardness result about approximating the set cover problem, Theorem 8.1.1 implies that we cannot develop an approximation algorithm with a guarantee of better than $(1 - \epsilon) \log n$ unless $NP \subseteq TIME[O(n^{\log \log n})]$. Therefore, it is interesting to identify special cases of the UFLPP that can be approximated to within a constant factor. The following two theorems deal with two different cost structures for the UFLPP for which this is the case. Both results assume that the production cost functions and inventory cost functions are linear and can be written as $P_{it}(p_{it}) = b_{it}p_{it}$ $(i = 1, \ldots, m,\ t = 1, \ldots, T)$ and $H_{it}(I_{it}) = h_{it}I_{it}$ $(i = 1, \ldots, m,\ t = 1, \ldots, T)$ respectively. The first result, in addition, assumes that the production and inventory holding costs do not depend on the facility.

Theorem 8.1.2. *If there exists an α-approximation algorithm for the metric UFLP, then there exists a α-approximation algorithm for the class of instances of the UFLPP with linear and facility-invariant production and inventory holding costs.*

Proof. We will first define a UFLP based on this class of instances of the UFLPP. In this conversion, the facility opening costs will remain the same as the facility opening costs in the UFLPP. We will focus on defining the connection cost of a customer to a facility. Denote the unit production and inventory holding costs by $b_{it} = b_t$ and $h_{it} = h_t$ for $i = 1, \ldots, m$ and $t = 1, \ldots, T$, respectively. First, since production and inventory costs are linear and production/inventory is uncapacitated, we can determine the optimal cost of meeting demand in period t, which we denote C_t^*, through a simple recursion:

$$C_t^* = \min\{b_t, h_{t-1} + C_{t-1}^*\}.$$

Once we have C_t^* for $t = 1, \ldots, T$, the optimal production/inventory costs of customer j are given by $\sum_{t=1}^{T} d_{jt} C_t^*$. It is clear that if we are given a solution to the UFLPP where the production and inventory costs corresponding to customer j are higher than $\sum_{t=1}^{T} d_{jt} C_t^*$, then this solution cannot be optimal since we can lower the production/inventory costs. Therefore, in order to solve UFLPP, we can restrict ourselves to solutions where the production and inventory costs are $\sum_{t=1}^{T} d_{jt} C_t^*$.

We define a UFLP by defining the demands to be equal to the aggregate demands d_j and the connection costs to be $\bar{c}_{ij} = c_{ij} + \gamma_j$ where $\gamma_j = \sum_{t=1}^{T} d_{jt} C_t^* / d_j$. Now consider any pair of facilities i, i' and pair of customers j, j':

$$\bar{c}_{ij} = c_{ij} + \gamma_j \leq c_{i'j} + c_{i'j'} + c_{ij'} + \gamma_j \leq \bar{c}_{i'j} + \bar{c}_{i'j'} + \bar{c}_{i'j'}$$

so that the connection costs satisfy the triangle inequality. Given an assignment of customers to facilities in this facility location problem, consider the assignment costs:

$$\sum_{i=1}^{m} \sum_{j=1}^{n} d_j \bar{c}_{ij} x_{ij} = \sum_{i=1}^{m} \sum_{j=1}^{n} d_j (c_{ij} + \gamma_j) x_{ij} = \sum_{i=1}^{m} \sum_{j=1}^{n} d_j c_{ij} x_{ij} + \sum_{j=1}^{n} d_{jt} C_t^* x_{ij},$$

which are the connection costs plus the optimal production and inventory costs in the same assignment of customers to facilities in the UFLPP. Similarly, given an assignment of customers to facilities in the UFLPP, the connection costs plus the optimal production and inventory costs are the same as the assignment costs in the facility location problem. Therefore, this metric UFLP solves this special class of the UFLPP and the parameters of the problem can be determined in $O(nT + nm)$ time. □

The second result again deals with instances for which the production cost and holding cost functions are linear. However we now assume that an ordering of the facilities exists such that, in every time period, it is as cost-effective to produce (or hold) a unit of demand at facility i than it is at facility $i' > i$. Note that we do *not* make any assumptions regarding the facility opening costs, so it may be very expensive to open a facility with cheap production/inventory costs and very cheap to open a facility with expensive production/inventory costs.

Theorem 8.1.3. *There exists a 6-approximation algorithm for the class of instances of the UFLPP with linear production and inventory holding costs if there exists an ordering of the facilities such that, if i and i' are two facilities, then $i < i'$ implies that for all t, $b_{it} \leq b_{i't}$ and $h_{it} \leq h_{i't}$.*

Proof. We will convert this class of the UFLPP to a facility location problem with service-installation costs. This problem generalizes the facility location problem where each customer j has an associated service $g(j)$. We can install service $g(j)$ at facility i at a cost of $f_i^{g(j)}$. We then wish to assign customers to facilities while minimizing the sum of the facility opening costs, service installation costs, and connection costs. Shmoys et al. [142] offer a primal-dual approximation algorithm for this problem with an approximation guarantee of 6, when there exists an ordering of the facilities such that if $i < i'$ then for any service ℓ, we have $f_i^\ell \leq f_{i'}^\ell$. Note that this ordering makes no assumptions on the values of the facility opening costs.

In our conversion, the facility opening costs and connection costs remain the same as in the UFLPP. We now describe how to determine the service-installation costs at each facility. We can determine the optimal cost of meeting demand in period t at facility i, C^*_{it}, through a similar recursion as in the proof of Theorem 8.1.2 above. It is clear that if $i < i'$ then $C^*_{it} \leq C^*_{i't}$ for all $t = 1, \ldots, T$. Now define the (optimal) production/inventory costs associated with assigning customer j to facility i:

$$f_i^j = \sum_{t=1}^{T} C^*_{it} d_{jt}.$$

Similar to the proof of Theorem 8.1.2, we can restrict our search for the optimal solution to the UFLPP to solutions where the production and inventory costs of customer j is equal to f_i^j if j is assigned to facility i. We define $g(j) = j$ and require that service $g(j)$ must be installed at the facility to which customer j is assigned. It is clear that if we assign a customer j to a facility i that the service-installation cost, f_i^j, is the optimal production/inventory costs for meeting the demand of customer j at facility i. Similarly, if customer j is assigned to facility i in a solution to the UFLPP, it is clear that optimal production/inventory costs associated with this assignment will equal the service installation cost associated with customer j. This conversion can be done in $O(nmT)$ time and we can apply the algorithm of Shmoys et al. [142] to approximate this special case of the UFLPP within a factor of 6. □

8.2 Approximating the UFLPP with Seasonal Demands

In this section, we will study the important subclass of the UFLPP in which the demands follow a *seasonal pattern*. In particular, we say that a problem in the class has seasonal demands if each customer's aggregate demand is distributed among the T periods using a common (nonnegative) vector of multiplicative seasonal effects, $\sigma^\top = (\sigma_1, \ldots, \sigma_T)$, so that the demand of customer j in period t is given by $d_{jt} = \sigma_t d_j$. For convenience and without loss of generality we will assume that the seasonal effects are normalized so that $\sum_{t=1}^{T} \sigma_t = 1$.

Consider the optimal production and inventory holding costs required at facility i to meet a vector of demands equal to a nonnegative scalar multiple z of the vector of seasonal effects σ as given by the optimal solution value of the following optimization problem:

$$\text{minimize } \sum_{t=1}^{T} \left(P_{it}(p_{it}) + H_{it}(I_{it})\right)$$

subject to (PP(i))

$$\begin{aligned} I_{i,t-1} + p_{it} &= \sigma_t z + I_{it} \qquad \text{for } t = 1, \ldots, T \\ I_{i0} &= 0 \\ p_{it}, I_{it} &\geq 0 \qquad \text{for } t = 1, \ldots, T. \end{aligned}$$

We denote the optimal value function of this problem by $g_i(z)$. It is clear that we can restrict our search for the optimal solution to the UFLPP to solutions of the UFLPP where the production and inventory costs at a facility are equal to $g_i(\sum_{j=1}^{n} d_j x_{ij})$ since otherwise we could improve the cost of the current solution by altering the production and inventory variables. With a slight abuse of notation, we define the function

$$f_i(z) = \begin{cases} f_i + g_i(z) & \text{if } z > 0 \\ 0 & \text{if } z = 0. \end{cases}$$

We can then represent the total facility opening costs and optimal production and inventory costs of the UFLPP with seasonal demands as

$$\sum_{i=1}^{m} f_i \left(\sum_{j=1}^{n} d_j x_{ij} \right).$$

Let us now consider the connection costs associated with the UFLPP with seasonal demands. We have:

$$\sum_{j=1}^{n} \sum_{i=1}^{m} \left(\sum_{t=1}^{T} d_{jt} \right) c_{ij} x_{ij} = \sum_{j=1}^{n} \sum_{i=1}^{m} \left(d_j \sum_{t=1}^{T} \sigma_t \right) c_{ij} x_{ij} = \sum_{j=1}^{n} \sum_{i=1}^{m} d_j c_{ij} x_{ij}$$

since $\sum_{t=1}^{T} \sigma_t = 1$. Recalling that the connection costs c_{ij} are metric, the UFLPP is thus an instance of the following class of generalized metric facility location problems:

$$\text{minimize} \quad \sum_{i=1}^{m} f_i \left(\sum_{j=1}^{n} d_j x_{ij} \right) + \sum_{j=1}^{n} \sum_{i=1}^{m} d_j c_{ij} x_{ij}$$

subject to (P)

$$\begin{aligned} \sum_{i=1}^{m} x_{ij} &= 1 && \text{for } j = 1, \ldots, n \\ x_{ij} &\in \{0, 1\} && \text{for } i = 1, \ldots, m,\ j = 1, \ldots, n. \end{aligned}$$

The following lemma shows that in any instance of (P) resulting from the UFLPP with seasonal demands the functions f_i are concave by showing that the functions g_i are concave.

Lemma 8.2.1. *The functions g_i, $i = 1, \ldots, m$, are concave.*

Proof. It is well-known that the only candidate solutions that need to be considered for PP(i) are those that are characterized by a set of periods in which production takes place, together with the so-called zero-inventory property that says that the ending inventory in a period preceding a production period is equal to zero. It is easy to see that, given a particular choice of production periods, the value of PP(i) as a function of z is a concave function, so that g_i is the minimum of a family of concave functions, which implies that g_i itself is concave. □

We will refer to the problem class (P) with concave functions f_i as the concave cost facility location problem (CCFLP). Before studying this problem in more detail, note that it has other applications besides the UFLPP with seasonal demands. For example, Daskin et al. [39] and Shen et al. [141] consider a joint inventory-location problem, special cases of which belong to the class of the CCFLP. In Section 8.2.1, we will develop a 1.61-approximation algorithm for the CCFLP and use this algorithm and the idea of

cost-scaling to ultimately give a 1.52-approximation algorithm for it. In Section 8.2.2, we examine the problem (P) for other structures of the functions f_i, $i = 1, \ldots, m$.

8.2.1 Approximation Algorithms for the CCFLP

Before we begin developing an approximation algorithms for the CCFLP, it will be necessary to define the time required to evaluate the function $f_i(z)$ for some fixed value of z.

Definition 8.2.2. *We let ϕ denote the time required to evaluate the function $f_i(z)$ for a fixed z.*

It is important to note that this evaluation may not be trivial for all concave functions $f_i(z)$. For example, in the CCFLP that arises from the UFLPP with seasonal demands, we have that $\phi = O(T^2)$ for general concave production and holding cost functions and $\phi = O(T \log T)$ for fixed-charge plus linear production costs and linear holding costs (the cost structure of the economic lot-sizing problem).

Further, for the remainder of this section, we will assume that the demand level of the customers (d_j for $j = 1, \ldots, n$) are integral. We can make this assumption without loss of generality for the CCFLP with rational demand levels, since, if the demand levels are rational, we can multiply the demand levels of the customers by a large integer while dividing the connection costs and facility cost functions by the same large integer. In almost all situations, the base demand level of each customer will be rational, so, therefore, the integrality assumption is not very restrictive in practice.

We will begin by generalizing the 1.61-approximation algorithm of Jain et al. [87]. We now informally describe this algorithm. At any point in the algorithm we will have two sets of customers: connected customers and unconnected customers. Each customer will then make an offer to each facility. The offer of a connected customer to a facility is equal to the amount the customer would save in paying the connection to this facility as opposed to paying the connection cost to the facility it is currently assigned. The offer of an unconnected customer to a facility will be based on the customer's budget and the

connection cost to this facility. If a set of customers offer enough to cover the opening cost of a facility, the facility will open and each customer in the set will be assigned to it. If this event occurs, then the amount offered to the facility by a customer can be thought of as the customer's contribution to the opening of the facility. If no facility is offered enough to open, we raise the budget of each of the unconnected customers.

There are two crucial elements to this generalization that differ from the algorithm of Jain et al. [87]: (i) the idea of "contribution withdrawal", i.e., if a customer switches facilities, it withdraws some of its contribution to the facility it is initially connected to and offers it to the facility it switches to and (ii) efficiently solving a nonlinear fractional binary programming problem that is necessary at each iteration of the algorithm. We then use this algorithm to generalize the two-phase algorithm of Mahdian et al. [98] to the CCFLP, deriving an approximation algorithm for the CCFLP with a guarantee of 1.52.

We will now describe the notation used in our algorithm to approximate the CCFLP. Let $D = \{1, \ldots, n\}$ be the set of customers. At any point in the algorithm, $U \subseteq \{1, \ldots, n\}$ will denote the set of unconnected customers. Moreover, for any facility, i, A_i will denote the set of customers assigned to it by the algorithm so far, and $T_i = \sum_{j \in A_i} d_j$ will be the corresponding total demand currently assigned. There is a notion of time, τ, associated with the algorithm. For any $j \in U$, we set the budget of customer j at time τ equal to $\alpha_j = \tau$. Each customer will offer some money from its budget to a facility i, which we denote o_{ji}. The offer of customer j to facility i (where $j \notin A_i$) depends on whether customer j has been assigned to a facility earlier than time τ. In particular, at time τ, if $j \in U$, then $o_{ji} = d_j \max\{\alpha_j - c_{ij}, 0\}$; if $j \in A_{i'}$, then $o_{ji} = d_j \max\{w_{ji'} + c_{i'j} - c_{ij}, 0\}$ where $w_{ji'} = (f_{i'}(T_{i'}) - f_{i'}(T_{i'} - d_j))/d_j$. If $j \in A_{i'}$, then $w_{ji'}$ is the amount of customer j's contribution to facility i' that would be withdrawn from facility i' if customer j switches facilities. Note that the introduction of $w_{ji'}$ is significantly different than the algorithm of Jain et al. [87], which does not allow a customer to withdraw some of its contribution from the facility it is assigned. The idea behind the algorithm is to, as time progresses, assign a

subset of customers to a facility if the total offer by this set covers the additional facility costs. We are now in position to present the algorithm.

Greedy Algorithm

Step 0. Set $U = \{1, \ldots, n\}$, $A_i = \emptyset$, and $T_i = 0$ for all $i = 1, \ldots m$. Initialize $\tau = 0$.

Step 1. If $U = \emptyset$, terminate the algorithm. Otherwise, increase τ until there exists a facility i and a set $S \subseteq D \backslash A_i$ such that

$$\sum_{j \in S} o_{ji} = f_i \left(T_i + \sum_{j \in S} d_j \right) - f_i(T_i).$$

Step 2. Connect the clients in S to facility i, setting $A_i = A_i \cup S$ and $T_i = T_i + \sum_{j \in S} d_j$. For each $j \in S \cap U$, freeze $\alpha_j = \tau$. Set $U = U \backslash S$. For each $i' \neq i$, set $A_{i'} = A_{i'} \backslash S$ and $T_{i'} = \sum_{j \in A_{i'}} d_j$. Return to Step 1.

We will now show that we can determine the set S in Step 1 (and thereby the next time at which the Greedy Algorithm assigns a set of customers to a facility) can be found by solving a particular optimization problem.

Lemma 8.2.3. *The set S in Step 1 (and thereby the next time at which the Greedy Algorithm assigns a set of customers to a facility) can be found by solving the following minimization problem for each $i = 1, \ldots, m$:*

$$\min_{S_1 \in D \backslash (A_i \cup U), S_2 \in U} \frac{f_i(T_i + \sum_{j \in S_1 \cup S_2} d_j) - f_i(T_i) + \sum_{j \in S_1 \cup S_2} d_j a_j}{\sum_{j \in S_2} d_j} \qquad (FP(i))$$

where $a_j = c_{ij}$ if $j \in U$ and $a_j = -(c_{i'j} + w_{ji'} - c_{ij})$ if $j \in A_{i'}$.

Proof. Suppose that we arrive at Step 1 in the algorithm. We wish to determine the next time in which a set of customers (potentially) would be assigned to facility i. We have two types of customers that may be assigned to facility i: (i) customers currently connected to other facilities and (ii) customers currently unconnected. Let $S_1 \subseteq D \backslash (A_i \cup U)$ be a set of customers currently connected to other facilities and $S_2 \subseteq U$ be a set of customers currently unconnected. In order to determine the next time an assignment would occur at

facility i, we can solve

$$\text{minimize } \tau$$

subject to (SP(i))

$$\begin{aligned} \sum_{j \in S_1} d_j \max\{w_{ji'} + c_{i'j} - c_{ij}, 0\} & \\ + \sum_{j \in S_2} d_j \max\{\tau - c_{ij}, 0\} &= f_i\left(T_i + \sum_{j \in S_1 \cup S_2} d_j\right) - f_i(T_i) \qquad (8\text{–}1) \\ S_1 &\subseteq D \backslash (A_i \cup U) \\ S_2 &\subseteq U, \end{aligned}$$

since (8–1) ensures that the amount offered to a facility is equal to the additional amount incurred by the facility in serving customers in $S_1 \cup S_2$. Since we are minimizing τ, we can disregard solutions to this problem where we select a customer that does not offer anything to facility i. In other words, for potentially optimal solutions to SP(i), (8–1) can be rewritten as

$$\sum_{j \in S_1} d_j(w_{ji'} + c_{i'j} - c_{ij}) + \sum_{j \in S_2} d_j(\tau - c_{ij}) = f_i\left(T_i + \sum_{j \in S_1 \cup S_2} d_j\right) - f_i(T_i).$$

If we let $a_j = -(w_{ji'} + c_{i'j} - c_{ij})$ if $j \in D \backslash (A_i \cup U)$ and $a_j = c_{ij}$ if $j \in U$, then we see that (8–1) can be written as

$$\sum_{j \in S_2} d_j \tau = f_i\left(T_i + \sum_{j \in S_1 \cup S_2} d_j\right) - f_i(T_i) + \sum_{j \in S_1 \cup S_2} a_j d_j.$$

This means that we have a closed form expression for τ given S_1 and S_2 by dividing the previous equation by $\sum_{j \in S_2} d_j$. Therefore, we can write SP(i) as a problem FP(i):

$$\min_{S_1 \in D \backslash (A_i \cup U), S_2 \in U} \frac{f_i(T_i + \sum_{j \in S_1 \cup S_2} d_j) - f_i(T_i) + \sum_{j \in S_1 \cup S_2} d_j a_j}{\sum_{j \in S_2} d_j}.$$

If we let τ_i be the optimal solution to SP(i) or, equivalently, the optimal solution value of FP(i), then the next time that any set of customers can be connected to some

facility will be $\tau = \min_{i=1,\ldots,m} \tau_i$. Therefore, by solving FP(i) for facilities $i = 1, \ldots, m$, we can determine the next time an event occurs. □

Lemma 8.2.3 shows that it suffices to solve the problems FP(i) for $i = 1, \ldots, m$ in order to implement the Greedy Algorithm. Our next result deals with the complexity of solving a problem of the form FP(i).

Lemma 8.2.4. *The problem FP(i) can be solved in $O(n^2 \max\{\log n, \phi\})$ time, where the time required to evaluate $f_i(z)$ for a given z is ϕ.*

Proof. Let λ^* be the value of the optimal solution to FP(i). It can be shown that λ^* is equal to the value of λ for which there exists a non-trivial, i.e., $S_1 \cup S_2 \neq \emptyset$, optimal solution with value 0 to the following optimization problem:

$$\min_{S_1 \subseteq D \setminus (A_i \cup U), S_2 \subseteq U} f_i\left(T_i + \sum_{j \in S_1 \cup S_2} d_j\right) - f_i(T_i) + \sum_{j \in S_1} d_j a_j + \sum_{j \in S_2} d_j (a_j - \lambda). \qquad \text{(KP}(\lambda)\text{)}$$

Note further that the optimal solution of FP(i) is equal to a non-trivial optimal solution of KP(λ^*). We will use this fact, and the structure of the maximal optimal solution (i.e. the optimal solution where $|S_1 \cup S_2|$ is largest), in order to develop an efficient algorithm to solve FP(i).

Suppose that we wish to solve KP(λ) for a fixed λ. If we index the customers in $\{1, \ldots, n\} \setminus A_i$ in non-decreasing order of $\bar{a}_j(\lambda)$, where $\bar{a}_j(\lambda) = a_j - \lambda$ if $j \in U$ and $\bar{a}_j(\lambda) = a_j$ if $j \notin U$, it can be shown that there exists an optimal solution that selects the first ℓ, for some $\ell = 1, \ldots n - |A_i|$, customers in the order (see Shen et al. [141]). It can also be shown that if $\bar{a}_j(\lambda) = \bar{a}_{j'}(\lambda)$ for customers j, j', then there exists an optimal solution to KP(λ) where we select both j and j' or we select neither and therefore we can view j, j' as a single entity. In order to solve FP(i), it is sufficient to know the ordering of the customers based on $\bar{a}_j(\lambda^*)$, since we may then evaluate n potential solutions to FP(i) to determine the optimal solution. Therefore, we turn our attention to determining

the number of distinct orderings of the customers and, more importantly, the number of distinct candidate solutions arising for the orderings.

First, note that the ordering of the customers $j \in D\backslash(A_i \cup U)$ is independent of λ, i.e., it is only based on a_j. Similarly, the ordering of customers $j \in U$ is independent of λ. As we increase λ, new orderings only arise when a customer in $j \in D\backslash(A_i \cup U)$ and a customer $j \in U$ switch places. If there exists $j, j' \in D\backslash(A_i \cup U)$ (or $j, j' \in U$), such that $a_j = a_{j'}$, then $\bar{a}_j(\lambda) = \bar{a}_{j'}(\lambda)$ for any λ. Therefore, as mentioned above, we may essentially merge customers j and j', meaning that we will either select them both or select neither in evaluating candidate optimal solutions to FP(i). This immediately yields a bound of $O\left((n - |A_i| - |U|)|U|\right) = O(n^2)$ on the number of distinct orderings of the variables. Without further analysis, this leads to $O(n^3)$ candidate solutions for FP(i) by evaluating the solutions where we choose the first $\ell = 1, \ldots, n - |A_i|$ customers in each ordering. It turns out, however, that many of these candidate solutions are counted multiple times in this coarse analysis.

A deeper analysis of the number of candidate solutions that need to be considered to solve FP(i) is based on the following observations. Let $I(\lambda)$ be the indexing of the customers based on $\bar{a}_j(\lambda)$ and suppose that we know all the candidate solutions based on this ordering. Let $\bar{\lambda}$ be the next value that changes the indexing of the customers. If exactly two customers have swapped places in going from $I(\lambda)$ to $I(\bar{\lambda})$, we need to only evaluate a single new candidate solution. In particular, if customers ℓ and $\ell + 1$ swapped places, then we only need to consider the set consisting of the first $\ell - 1$ customers with the $(\ell + 1)^{\text{st}}$ customer in $I(\lambda)$. Therefore, it is necessary evaluate a number of candidate solutions equal to the number of swaps from $I(\lambda)$ to $I(\bar{\lambda})$ in order to evaluate all new candidate solutions in $I(\bar{\lambda})$. Further, if customers ℓ and $\ell + 1$ swapped places, in order to evaluate the the new candidate solution based on this swap, we need to know the sums $\sum_{j=1}^{\ell-1} d_j + d_{\ell+1}$ and $\sum_{j=1}^{\ell-1} a_j d_j + a_{\ell+1} d_{\ell+1}$. Once we know these sums, we need to evaluate

the function

$$f_i\left(T_i + \sum_{j=1}^{\ell-1} d_j + d_{\ell+1}\right)$$

in order to evaluate the candidate solution. Therefore, if we know the partial sums $\sum_{j=1}^{j'} d_j$ and $\sum_{j=1}^{j'} a_j d_j$ for $j' = 1, \ldots, n$ prior to the swap, we can evaluate the candidate solution in $O(\phi)$ time. Note further that we can update the new partial sums in $O(1)$ time. Therefore, given an ordering and all the partial sums corresponding to this ordering, if we perform a swap, we can evaluate a candidate solution and update the necessary partial sums based on this swap in $O(\phi)$ time.

The algorithm to solve FP(i) is then as follows. We view any $j, j' \in D\backslash(A_i \cup U)$ (or $j, j' \in U$) with $a_j = a_{j'}$ as a single entity. First, for each $j \in U$ and $j' \in D\backslash(A_i \cup U)$, we determine the value of $\lambda_{jj'}$ where $\bar{a}_j(\lambda_{jj'}) = \bar{a}_{j'}(\lambda_{jj'})$. It is only necessary to consider values of $\lambda_{jj'}$ that are non-negative. We sort these values in non-decreasing order, which requires $O(n^2 \log n)$ time. Then the actual algorithm starts by determining the indexing of the customers based on the values $\bar{a}_j(0)$ and evaluating the $O(n)$ corresponding candidate solutions. As we evaluate each of these candidate solutions, we record the sum of the demands for the first ℓ customers as well as the sum of the $a_j d_j$-values for the first ℓ customers, for all values of ℓ. At this point, we have a particular customer indexing and have evaluated all candidate solutions according to this indexing. We then continue the algorithm by examining the minimum value of $\lambda_{j'j''}$ not yet considered. This yields a new indexing by swapping the places of customer j' and j'' for each pair j', j'' that achieves the minimum value. This leads to a number of new candidate solutions that must be evaluated (in particular, one for each pair j', j'' for which $\lambda_{j'j''}$ attains the minimum). Since we have recorded the partial sums of demands and $a_j d_j$-values we can evaluate the new candidate solution and update the sums appropriately in $O(\phi)$ time. As we raise λ, each new candidate solution that must be evaluated is induced by a swapping of customers. Since there are $O(n^2)$ such swaps, the algorithm requires $O(n^2 \max\{\log n, \phi\})$ time. □

Since the Greedy Algorithm can be implemented in a way that solves FP(i) for $i = 1, \ldots, m$ at each iteration, and since we assign at least one unconnected customer in each iteration, Lemma 8.2.4 implies a running time of $O(mn^3 \max\{\log n, \phi\})$. We will now focus on determining the approximation guarantee of the algorithm. This analysis generalizes the algorithm for the metric uncapacitated facility location problem with an approximation guarantee of 1.61 presented in Jain et al. [87]. We begin with a property of the variables α_j.

Lemma 8.2.5. *The total cost of the solution produced by the Greedy Algorithm is no more than* $\sum_{j=1}^{n} d_j \alpha_j$.

Proof. At some point in time, consider facility i and A_i. We will first show that the total savings in disconnecting any subset $S \subseteq A_i$ is at least $\sum_{j \in S} d_j(w_{ji} + c_{ij})$. By the definition of w_{ji} and the concavity of f_i, we have

$$d_j w_{ji} = f_i(T_i) - f_i(T_i - d_j) \leq \frac{d_j}{\sum_{j \in S} d_j} \left(f_i(T_i) - f_i\left(T_i - \sum_{j \in S} d_j \right) \right)$$

implying

$$\begin{aligned} \sum_{j \in S} d_j(w_{ji} + c_{ij}) &\leq f_i(T_i) - f_i\left(T_i - \sum_{j \in S} d_j \right) + \sum_{j \in S} d_j c_{ij} \\ &= f_i(T_i) + \sum_{j \in A_i} d_j c_{ij} - f_i\left(T_i - \sum_{j \in S} d_j \right) - \sum_{j \in A_i \setminus S} d_j c_{ij} \end{aligned}$$

where the last term is the savings of disconnecting the customers in the set S. We will now prove our desired result by an inductive argument based on τ. We will show that the total cost of serving customers in $D \setminus U$ is at most $\sum_{j \in D \setminus U} d_j \alpha_j$ at time τ. It is clearly true for $\tau = 0$. Assume that it holds for time τ and consider the first $\tau' > \tau$ that an event occurs in the Greedy Algorithm, i.e. there exists a facility i and set $S \subseteq D \setminus A_i$ such that

$$\sum_{j \in S} o_{ji} = \sum_{j \in S} f_i\left(T_i + \sum_{j \in S} d_j \right) - f_i(T_i)$$

Let $S_1 \subset S$ be the set of clients that switched to i from another facility and $S_2 = S \backslash S_1$. The additional cost of serving customers in S at facility i is

$$\begin{aligned}
\sum_{j \in S} d_j c_{ij} + f_i \left(T_i + \sum_{j \in S} d_j \right) - f_i(T_i) &= \sum_{j \in S} (d_j c_{ij} + o_{ji}) \\
&= \sum_{j \in S_1} (d_j c_{ji} + o_{ji}) + \sum_{j \in S_2} (d_j c_{ji} + o_{ji}) \\
&= \sum_{j \in S_1} d_j (c_{ijj} + w_{ji_j}) + \sum_{j \in S_2} d_j \alpha_j
\end{aligned}$$

where i_j is the facility that $j \in S_1$ was previously connected. As previously shown, the cost in savings in disconnecting the customers $j \in S_1$ from the facilities they were previously connected to is at least $\sum_{j \in S_1} d_j (c_{ijj} + w_{ji_j})$ and therefore the cost is increased by at most $\sum_{j \in S_2} d_j \alpha_j$. Our result follows from the induction hypothesis. □

Consider the optimal solution to the CCFLP and let A_i^* denote the set of customers assigned to facility i. If we can show the existence of a pair of constants (R_f, R_c) such that for every i

$$\sum_{j \in A_i^*} d_j \alpha_j \leq R_f f_i \left(\sum_{j \in A_i^*} d_j \right) + R_c \sum_{j \in A_i^*} d_j c_{ij}$$

then the Greedy Algorithm is an (R_f, R_c)-approximation algorithm. We will now focus on a facility i and the customers indexed $1, \ldots, k$ such that $\alpha_1 \leq \alpha_2 \leq \cdots \leq \alpha_k$. We will let e_j denote the connection cost of the j-th customer in the ordering to facility i and $f = f_i(\sum_{j=1}^k d_j)$. For each $j \in A_i$, define the critical time of j as the time τ right before j is connected to a facility. For any customer $\ell < j$, we define $r_{\ell j} = \alpha_\ell = \alpha_j$ if ℓ has not been connected to a facility at the critical time of j and $r_{\ell j} = w_{\ell i_\ell} + c_{i_\ell \ell}$ if ℓ is connected to facility i_ℓ at the critical time of j. We will now derive a set of inequalities involving the variables α_j and $r_{\ell j}$. The concept of "contribution withdrawal" in our algorithm plays a very important role in deriving these inequalities.

Lemma 8.2.6. *For any pair of customers j, ℓ, $\alpha_j \leq r_{\ell j} + e_i + e_j$.*

Proof. Consider the critical time of customer j. If ℓ is not connected at this time, then this inequality holds trivially. Otherwise, let i' be the facility which ℓ is connected. By definition, we have

$$r_{\ell j} = c_{i'\ell} + \frac{f_{i'}(T_{i'}) - f_{i'}(T_{i'} - d_\ell)}{d_\ell} \geq c_{i'\ell} + \frac{f_{i'}(T_{i'} + d_j) - f_{i'}(T_{i'})}{d_j} \tag{8–2}$$

where the inequality holds due to the concavity of $f_{i'}$. It must be true that

$$\alpha_j \leq c_{i'j} + \frac{f_{i'}(T_{i'} + d_j) - f_{i'}(T_{i'})}{d_j} \tag{8–3}$$

since otherwise we would have connected customer j to facility i' at an earlier time. Combining equations (8–2) and (8–3) yields

$$\alpha_j \leq c_{i'j} + r_{\ell j} - c_{i'\ell}.$$

Applying the triangle inequality to this equation proves our desired result. □

Lemma 8.2.6 relates the triangle inequality to our algorithm. The following lemma derives a set of inequalities involving the facility opening cost based upon the concavity of the facility cost function.

Lemma 8.2.7. *For every $j = 1, \ldots, k$,*

$$\sum_{\ell=1}^{j-1} d_\ell \max\{r_{\ell j} - e_\ell, 0\} + \sum_{\ell=j}^{k} d_\ell \max\{\alpha_\ell - e_\ell, 0\} \leq f.$$

Proof. Consider the time which customer j gets connected to a facility, i.e. $\tau = \alpha_j$. Let A_i and T_i be the set of clients served at facility i and the total demand served at i. The amount that client ℓ offers to facility i is $d_\ell \max\{\alpha_j - e_\ell, 0\}$ if $\ell \geq j$ and $d_j \max\{r_{\ell j} - e_\ell, 0\}$ if $\ell < j$. Let D_1 and D_2 be the set of connected and unconnected customers in $\{1, \ldots k\}$. We have that

$$\sum_{\ell \in D_1 \setminus A_i} d_\ell \max\{r_{\ell j} - e_\ell, 0\} + \sum_{\ell \in D_2} d_\ell \max\{\alpha_\ell - e_\ell, 0\} \leq f_i\left(T_i + \sum_{\ell \in (D_1 \cup D_2) \setminus A_i} d_\ell\right) - f_i\left(T_i\right) \tag{8–4}$$

since otherwise we would have connected customer k to facility i at an earlier time. By the concavity of f_i and the fact that $T_i \geq \sum_{\ell \in D_1 \cap A_i} d_\ell$, we have that

$$\begin{aligned} & f_i\left(T_i + \sum_{\ell \in (D_1 \cup D_2) \setminus A_i} d_\ell\right) - f_i\left(T_i\right) \\ \leq \; & f_i\left(\sum_{\ell \in D_1 \cap A_i} d_\ell + \sum_{\ell \in (D_1 \cup D_2) \setminus A_i} d_\ell\right) - f_i\left(\sum_{\ell \in D_1 \cap A_i} d_\ell\right) \\ \leq \; & f_i\left(\sum_{\ell=1}^{k} d_\ell\right) - f_i\left(\sum_{\ell \in D_1 \cap A_i} d_\ell\right). \end{aligned} \tag{8–5}$$

If we combine equations (8–4) and (8–5), it yields that:

$$\sum_{\ell \in D_1 \setminus A_i} d_\ell \max\{r_{\ell j} - e_\ell, 0\} + \sum_{\ell \in D_2} d_\ell \max\{\alpha_\ell - e_\ell, 0\} \leq f_i\left(\sum_{\ell=1}^{k} d_\ell\right) - f_i\left(\sum_{\ell \in D_1 \cap A_i} d_\ell\right). \tag{8–6}$$

We now turn our attention to deriving an inequality for $\sum_{\ell \in D_1 \cap A_i} d_\ell \max\{r_{\ell,j} - e_\ell, 0\}$. Due to the concavity of f_i, we have for each $\ell' \in A_i$, that

$$f_i(T_i) - f_i(T_i - d_{\ell'}) \leq f_i\left(\sum_{\ell \in D_1 \cap A_i} d_\ell\right) - f_i\left(\sum_{\ell \in D_1 \cap A_i} d_\ell - d_{\ell'}\right) \leq \frac{d_{\ell'} f_i\left(\sum_{\ell \in D_1 \cap A_i} d_\ell\right)}{\sum_{\ell \in D_1 \cap A_i} d_\ell}.$$

It follows that

$$\begin{aligned} \sum_{\ell \in D_1 \cap A_i} d_\ell \max\{r_{\ell,j} - e_\ell, 0\} &= \sum_{\ell \in D_1 \cap A_i} d_\ell (r_{\ell,j} - e_\ell) = \sum_{\ell \in D_1 \cap A_i} d_\ell w_{\ell,i} \\ &= \sum_{\ell \in D_1 \cap A_i} \left(f_i(T_i) - f_i(T_i - d_\ell)\right) \\ &\leq \sum_{\ell' \in D_1 \cap A_i} \frac{d_{\ell'} f_i(\sum_{\ell \in D_1 \cap A_i} d_\ell)}{\sum_{\ell \in D_1 \cap A_i} d_\ell} = f_i\left(\sum_{\ell \in D_1 \cap A_i} d_\ell\right). \end{aligned} \tag{8–7}$$

Therefore, by combining equations (8–6) and (8–7), we have

$$\sum_{\ell \in D_1} d_\ell \max\{r_{\ell,j} - e_\ell, 0\} + \sum_{\ell \in D_2} d_\ell \max\{r_{\ell,j} - e_\ell, 0\} \leq f_i\left(\sum_{\ell=1}^{k} d_\ell\right) = f.$$

Our result follows by noticing that

$$\begin{aligned}
&\sum_{\ell=1}^{j-1} d_\ell \max\{r_{\ell j} - e_\ell, 0\} + \sum_{\ell=j}^{k} d_\ell \max\{\alpha_\ell - e_\ell, 0\} \\
&\quad = \sum_{\ell \in D_1} d_\ell \max\{r_{\ell,j} - e_\ell, 0\} + \sum_{\ell \in D_2} d_\ell \max\{r_{\ell,j} - e_\ell, 0\}.
\end{aligned}$$

□

Lemma 8.2.6 and Lemma 8.2.7 lead to our main result about the Greedy Algorithm.

Theorem 8.2.8. *For any $R_f \geq 1$, the Greedy Algorithm is an (R_f, R_c)-approximation algorithm, where R_c is an upper bound on the solution of*

$$\textit{maximize } \frac{\sum_{j=1}^{k} d_j \alpha_j - R_f f}{\sum_{j=1}^{k} d_j e_j}$$

subject to *(FLP)*

$$\begin{aligned}
\alpha_j &\leq \alpha_{j+1} && \text{for } j = 1, \ldots, k-1 \\
r_{\ell j+1} &\leq r_{\ell j} && \text{for } \ell = 1, \ldots, j-1,\ j = 2, \ldots, k \\
\alpha_j &\leq r_{\ell j} + e_j + e_\ell && \text{for } \ell = 1, \ldots, j-1,\ j = 2, \ldots, k \\
\sum_{\ell=1}^{j-1} d_\ell \max\{r_{\ell j} - e_\ell, 0\} & \\
+ \sum_{\ell=j}^{k} d_\ell \max\{\alpha_\ell - e_\ell, 0\} &\leq f && \text{for } j = 1, \ldots, k \\
\alpha_j, d_j, f, r_{\ell j} &\geq 0 && \text{for } \ell = 1, \ldots, j-1,\ j = 2, \ldots, k.
\end{aligned}$$

FLP is called the factor revealing linear program (see Jain et al. [87]). It was shown by Jain et al. [87] that, for the case of unit demands: $d_j = 1$ for $j = 1, \ldots, k$, if $R_f = 1.61$ then $R_c = 1.61$ and if $R_f = 1$ then $R_c = 2$. Further, Mahdian et al. [98] show that if $R_f = 1.11$ then $R_c = 1.78$, which will be important in developing a 1.52-approximation algorithm for the CCFLP. These results still hold for general integral d_j by replicating each α_j by d_j copies and each $r_{\ell j}$ by $d_\ell \times d_j$ copies. It can easily be seen that the replicated copies still satisfy the constraints of FLP. This leads to the following result.

Lemma 8.2.9. *The Greedy Algorithm is a* 1.61*-approximation algorithm for the CCFLP with a running time of* $O(mn^3 \max\{\log n, \phi\})$.

We can use the Greedy Algorithm and Theorem 8.2.8 to generalize the algorithm of Mahdian et al. [98].

Theorem 8.2.10. *There exists a* 1.52*-approximation algorithm for the CCFLP with a running time of* $O(mn^2 \max\{\log n, \phi\} \max\{m, n\})$.

Proof. We develop a two-phase algorithm for the CCFLP. In the first phase, we use the concept of scaling the facility costs that was introduced by Charikar and Guha [33]. Specifically, we scale up the facility opening costs by a factor of δ, i.e. $\bar{f}_i(\sum_{j=1}^n d_j x_{ij}) = \delta f_i(\sum_{j=1}^n d_j x_{ij})$, and apply the Greedy Algorithm. Given the solution returned by the first phase, we scale down the facility cost functions back to the original facility cost functions at the same rate. If at any point in this phase, we can reassign a set of customers to a different facility without increasing the cost of the solution, we perform the reassignment. If we are at a point in the second phase of the algorithm, say the facility cost functions are scaled up by a factor of δ^c, where no reassignments can be performed, then it can be shown that determining the next factor, δ^*, where a reassignment can be performed is equivalent to determining the maximum value of δ^* that satisfies

$$\sum_{j \in S} d_j \max\{w_{i_j j} + c_{i_j j} - c_{ij}, 0\} = \delta^* f_i(T_i + \sum_{j \in S} d_j) - \delta^* f_i(T_i)$$

for some facility $i = 1, \ldots, m$ and set $S \subseteq D \backslash A_i$. For a facility i, define the problem

$$K_i(\delta') = \max_{S \subseteq D \backslash A_i} \sum_{j \in S} d_j \max\{w_{i_j j} + c_{i_j j} - c_{ij}, 0\} - \delta' f_i(T_i + \sum_{j \in S} d_j) - f\delta' f_i(T_i)$$

where i_j is the facility j is currently assigned to and $w_{i_j j}$ is defined as above. For a fixed δ', this problem belongs to the same class as the problem KP(λ) in the proof of Lemma 8.2.4. If we sort the customers in $D \backslash A_i$ according to $d_j \max\{w_{i_j j} + c_{i_j j} - c_{ij}, 0\}$ in non-increasing order, then an optimal solution to $K_i(\delta')$ contains the first k customers in the ordering. Note that the ordering of the customers is independent of δ'. Define δ'_k to

be the value of δ' such that the solution containing the first k customers in the ordering has its objective value equal to 0. Therefore, the largest value of δ'_k will be the first time we could perform a reassignment at facility i. This implies that we can determine δ^* by applying the above procedure for each facility, in $O(mn \max\{\log n, \phi\})$. If $\delta^* \leq 1$, then we terminate the second phase of the algorithm. In each reassignment, at least one customer is switched to a different facility. Each customer can be reassigned to each facility a constant number of times throughout the second phase of the algorithm, and therefore we have a bound of $O(mn)$ reassignments. Therefore, the second phase of the algorithm runs in $O(m^2n^2 \max\{\log n, \phi\})$.

The derivation of the approximation guarantee of the algorithm uses the results of Mahdian et al. [98]. Mahdian et al. [98] showed that their two-phase algorithm is a $(R_f + \ln \delta + \epsilon, 1 + \frac{R_c-1}{\delta})$-approximation algorithm for any (R_f, R_c) given by Theorem 8.2.8. This analysis relied on deriving a factor revealing linear program by analyzing an algorithm that scales down δ in L discrete steps rather continuously. We can apply a similar analysis as this to show that our two-phase algorithm is a $(R_f + \ln \delta + \epsilon, 1 + \frac{R_c-1}{\delta})$-approximation algorithm. If we set $(R_f, R_c) = (1.11, 1.78)$ and $\delta = 1.504$, then we have a $(1.11 + \ln(1.504) + \epsilon, 1 + \frac{1.78-1}{1.504}) = (1.5181 + \epsilon, 1.518)$-approximation algorithm for the CCFLP. Therefore, the two-phase algorithm has a guarantee of 1.52. □

This immediately leads to the following result for the UFLPP with seasonal demands.

Corollary 8.2.11. *There exists a* 1.52*-approximation algorithm for the UFLPP with seasonal demands with a running time of* $O(mn^2 \max\{\log n, T^2\} \max\{m, n\})$ *in the case of concave production and holding cost functions and* $O(mn^2 \max\{\log n, T \log T\} \max\{m, n\})$ *for fixed-charge plus linear production costs and linear holding costs.*

Proof. For general concave production and holding cost functions, we can use the algorithm of Wagner [156] or Veinott [154] to evaluate the function $f_i(z)$ in $O(T^2)$ time. For fixed-charge plus linear production costs and linear holding costs, we can use the

algorithm of Aggarwal and Park [1], Federgruen and Tzur [47], or Wagelmans et al. [155] to evaluate the function $f_i(z)$ in $O(T \log T)$ time. □

8.2.2 Results on Generalizations of the CCFLP

In this subsection, we will discuss approximation algorithms and results for generalized facility location problems, (P), where the functions f_i, $i = 1, \ldots, m$, have different structures. In particular, we examine the problem (P) where the functions f_i, $i = 1, \ldots, m$, are subadditive. Recall that a function, say f, is called *subadditive* if $f(x + y) \leq f(x) + f(y)$ for any non-negative x and y. This class of problems was recently proposed by Gabor and van Ommeren [59]. They discuss three examples of facility location problems with stochastic demands that belong to this class of problems. Other examples of facility location problems with subadditive facility costs can be found in Gabor and van Ommeren [58] and Rodolakis et al. [121]. Gabor and van Ommeren [59] develop an approximation algorithm with a guarantee of $2(1 + \epsilon)$, for any $\epsilon > 0$. Also, for a special class of subadditive cost functions, they develop an approximation algorithm with a guarantee of 2. It was observed in both Gabor and van Ommeren [59] and Rodolakis et al. [121] that the concave envelope of a subadditive function is a 2-approximation of the function. Since the Greedy Algorithm that we developed above is a $(1, 2)$-approximation algorithm for the CCFLP, this yields a $(2, 2)$-approximation algorithm for the facility location problem with discrete subadditive cost functions, given that we can evaluate the concave envelope of a subadditive function at a single point in polynomial time. Note that this is less restrictive than needing to construct the entire concave envelope of a subadditive function in polynomial time.

Hajiaghayi et al. [75] also considered the problem (P) with unit demands but where the functions f_i are convex. They offered a polynomial-time exact algorithm for this problem. The next theorem shows that, for non-unit demands, there cannot exist an approximation algorithm for the variant of (P) where the cost functions are convex functions unless $P = NP$.

Theorem 8.2.12. *Consider the problem class (P) where the functions f_i are convex. There cannot exist a polynomial time $\alpha(n,m)$-approximation algorithm for this problem unless $P = NP$.*

Proof. We will show that the Partition Problem can be solved in polynomial time if we would have a polynomial time $\alpha(n,m)$-approximation algorithm for (P) where the functions f_i are convex. Recall that the Partition Problem is defined over a set of integers a_j. The question is: does there exist $x \in \{0,1\}^n$ such that $\sum_{j=1}^n a_j x_j = \frac{1}{2}\sum_{i=1}^n a_j$? We define an instance of (P) with n customers having demands $d_j = a_j$ $(j = 1,\ldots,n)$ and 2 facilities with all connection costs equal to 0, i.e. $c_{ij} = 0$ for $i = 1,2$ and $j = 1,\ldots,n$. We define the piecewise linear convex cost functions f_i for $i = 1,2$ as:

$$f_i(y) = \begin{cases} 1 & \text{if } 0 \le y \le \frac{1}{2}\sum_{j=1}^n d_j \\ (2\alpha(n,2)+1)\left(y - \frac{1}{2}\sum_{j=1}^n d_j\right) & \text{if } y \ge \frac{1}{2}\sum_{j=1}^n d_j. \end{cases}$$

If there exists a partition, say $S \subseteq \{1,\ldots,n\}$, then we may assign all customers in S to facility 1 and all customers in $\{1,\ldots,n\}\backslash S$ to facility 2 at a total cost of 2. Similarly, if the optimal solution to the (P) is 2, then the set of customers assigned to facility 1 form a partition. Otherwise, at least one unit of demand will pay the cost of the second segment of f_i, so the optimal cost is at least $2\alpha(n,2)+2$. Now suppose that we have a polynomial time $\alpha(n,2)$-approximation algorithm that we apply to this problem. If it returns a solution with cost no more than $2\alpha(n,2)$, then there exists a partition. If it does not, then there does not exist a partition. This implies the desired result. □

8.3 Approximating the UFLPP-DA

In this section, we discuss several approximation results for the UFLPP-DA. In particular, we first develop analogous results to those appearing in Section 8.1 for the UFLPP-DA. We then discuss another special class of the UFLPP-DA and show that it can be formulated as a problem belonging to a new class of facility location problems, which is a generalization of the facility location problem with service-installation costs (see Shmoys

et al. [142]). We first begin with several results that will help determine the complexity of approximating the general UFLPP-DA.

Lemma 8.3.1. *If there exists an α-approximation algorithm for the UFLPP-DA, then there also exists an α-approximation algorithm for the UFLPP where each customer has at most one period with non-zero demand.*

Proof. Suppose that we are given an instance of the UFLPP where customer j only has a non-zero demand in period j_t. Suppose that we define a nearly identical UFLPP-DA to this UFLPP, with the only exception being that the customer j does not need to be assigned to the same facility over the entire horizon. Given any solution to this UFLPP-DA, it is clear that we can convert this to a feasible solution to the UFLPP with the same cost by assigning all time periods $t \neq t_j$ of that customer to the facility which time period t_j is assigned since there is no demand in time periods $t \neq t_j$. Therefore, the optimal solution to the UFLPP-DA will have the same cost as the optimal solution to the UFLPP. Given the solution returned by an α-approximation algorithm for the UFLPP-DA, we can convert it to a solution to the UFLPP with equal cost and, therefore, have an α-approximation algorithm for this class of the UFLPP. □

Lemma 8.3.2. *If there exists an α-approximation algorithm for the UFLPP-DA, then there also exists an α-approximation algorithm for the set cover problem.*

Proof. This follows directly from Lemma 8.3.1 and the fact that in proof of Theorem 8.1.1, we reduced the Set Cover problem to a UFLPP where each customer has at most one non-zero demand period. □

Therefore, using the result of Feige [51], we cannot develop an approximation algorithm with a guarantee of better than $(1-\epsilon)\log n$ unless $NP \subseteq TIME[O(n^{\log\log n})]$ for the general UFLPP-DA. It is appropriate to then focus on special cases of the UFLPP-DA that can be approximated within a constant factor. The fact that we can convert any

UFLPP-DA to an equivalent instance of the UFLPP yields analogous results to Theorems 8.1.2 and 8.1.3.

Theorem 8.3.3. *If there exists an α-approximation algorithm for the metric UFLP, then there exists a α-approximation algorithm for the class of instances of the UFLPP-DA with linear and facility-invariant production and inventory holding costs.*

Proof. We can create an instance of the UFLPP with T customers (j_t for $t = 1, \ldots, T$) for each customer j where the demand stream of customer j_t is given by $d_{j_t t'} = 0$ if $t' \neq t$ and $d_{j_t t} = d_{jt}$. Therefore, by Theorem 8.3.3, we can convert the UFLPP-DA problem into a metric UFLP in $O(nT^2 + nmT)$ time since we have $O(nT)$ customers in the UFLPP. However, if we note that the time $O(nT^2)$ reflects the time required to compute $\sum_{t=1}^{T} C_t^* d_{\jmath t}$ for each customer $\jmath$ in the UFLPP, we see that this can be reduced to $O(nT)$ time since each customer has exactly one positive demand. Therefore, we can convert the UFLPP-DA problem into a metric UFLP problem in $O(nmT)$ time. □

Theorem 8.3.4. *There exists a 6-approximation algorithm for the class of instances of the UFLPP-DA with linear production and inventory holding costs if there exists an ordering of the facilities such that, if i and i' are two facilities, then $i < i'$ implies that for all t, $b_{it} \leq b_{i't}$ and $h_{it} \leq h_{i't}$.*

Proof. We can create an instance of the UFLPP with T customers (j_t for $t = 1, \ldots, T$) for each customer j where the demand stream of customer j_t is given by $d_{j_t t'} = 0$ if $t' \neq t$ and $d_{j_t t} = d_{jt}$. Therefore, by Theorem 8.3.3, we can convert the UFLPP-DA problem into a class of facility location problem with service installations (with a known 6-approximation algorithm) in $O(nmT^2)$ time since we have $O(nT)$ customers in the UFLPP. However, we may (again) improve the time required to perform this conversion by noting that $O(nmT^2)$ reflects the time required to compute $f_i^{\jmath} = \sum_{t=1}^{T} C_{it}^* d_{\jmath t}$ for each customer $\jmath$ and facility i in the UFLPP. Noting that for each customer in this particular UFLPP instance has exactly one positive demand, this time can be reduced to $O(nmT)$ time. □

Unfortunately, we cannot apply the results of Section 8.2 to the UFLPP-DA with seasonal demands, i.e., $d_{jt} = d_j\sigma_t$ for $j = 1, \ldots, n$ and $t = 1, \ldots, T$. This is due to the fact that if we are given an instance of the UFLPP-DA with seasonal demands, the conversion to a UFLPP destroys the seasonality. It is still an open question where the UFLPP-DA with seasonal demands admits a constant factor approximation algorithm.

We will now discuss another special class of the UFLPP-DA, in particular, we examine the UFLPP-DA with economic lot-sizing costs. In other words, the production cost functions are equal to $P_{it}(p_{it}) = a_{it} + b_{it}p_{it}$ and $H_{it}(I_{it}) = h_{it}I_{it}$. Note that we already know from Lemma 8.3.2, that this problem without production setup costs is as difficult as the set cover problem to approximate. However, we will present a general framework to formulate this problem as a new class of facility location problems and then discuss special cases that can possibly be approximated within a constant factor. It will be convenient to define H_{ist} to be the variable cost associated with producing a unit at facility i in time period s and hold it until time period t, i.e., $H_{ist} = b_{is} + \sum_{\tau=s}^{t-1} h_{i\tau}$.

By defining the variable $x_{ijst} = 1$ if we meet the demand of customer j in time period t with production at facility i in time period s, then the UFLPP-DA with economic lot-sizing costs can be formulated as

$$\min \sum_{i=1}^{m} f_i y_i + \sum_{i=1}^{m}\sum_{t=1}^{T} a_{it} z_{it} + \sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{t=1}^{T}\sum_{s=1}^{t} d_{jt}(c_{ij} + H_{ist}) x_{ijst}$$

subject to

$$\begin{aligned}
\sum_{i=1}^{m}\sum_{s=1}^{t} x_{ijst} &= 1 && \text{for } j = 1, \ldots, n,\ t = 1, \ldots, T \\
x_{ijst} &\leq y_i && \text{for } i = 1, \ldots, m,\ j = 1, \ldots, n,\ s = 1, \ldots, t,\ t = 1, \ldots, T \\
x_{ijst} &\leq z_{is} && \text{for } i = 1, \ldots, m,\ j = 1, \ldots, n,\ s = 1, \ldots, t,\ t = 1, \ldots, T \\
x_{ijst} &\in \{0, 1\} && \text{for } i = 1, \ldots, m,\ j = 1, \ldots, n,\ s = 1, \ldots, t,\ t = 1, \ldots, T \\
y_i &\in \{0, 1\} && \text{for } i = 1, \ldots, m \\
z_{it} &\in \{0, 1\} && \text{for } i = 1, \ldots, m,\ t = 1, \ldots, T.
\end{aligned}$$

This formulation of the UFLPP-DA with economic lot-sizing costs somewhat resembles a facility location problem with service-installation costs (see Shmoys et al. [142]). It is indeed a generalization of the facility location with service installation costs with two additional considerations: (i) if a customer is assigned to a facility, the facility can select one service from a set of services to install to serve the customer and (ii) we pay a certain amount for each customer depending on the service chosen for the customer at the facility. In the case of the UFLPP-DA with economic lot-sizing costs, the set of customers is given by the pairs (j,t) for $j = 1,\ldots,n$ and $t = 1,\ldots,T$. For a given customer (j,t), the choice to determine which service to install at the facility to which it is assigned is equivalent to determining the period in which we will produce the demand of the customer. We must pay the variable production and inventory costs associated with the customer and the time period we will serve the customer, i.e., if we will serve customer (j,t) in period s at facility i, then we pay $d_{jt}H_{ist}$.

In general, the new *facility location problem with customer service flexibility* (FLP-CSF) can be formulated as follows. We are given a set of facilities $i = 1,\ldots,m$, a set of services, $k = 1,\ldots,K$, and a set of customers $\ell = 1,\ldots,L$. The cost of opening facility i is f_i, the cost of installing service k at facility i is f_i^k, and the per-unit cost of serving customer ℓ with service k at facility i is $S_{ik\ell}$. The set of services that may serve customer ℓ is denoted by K_ℓ. The FLP-CSF is thus formulated as

$$\min \sum_{i=1}^{m} f_i y_i + \sum_{i=1}^{m}\sum_{k=1}^{K} f_i^k z_{ik} + \sum_{i=1}^{m}\sum_{\ell=1}^{L}\sum_{k\in K_\ell} d_\ell S_{ik\ell} x_{ik\ell} + \sum_{i=1}^{m}\sum_{\ell=1}^{L} d_\ell c_{i\ell} x_{ik\ell}$$

subject to

$$\begin{aligned}
\sum_{i=1}^{m} x_{i\ell} &= 1 && \text{for } \ell = 1,\ldots,L \\
x_{i\ell} &\le \sum_{k\in K_\ell} z_{ik} && \text{for } i = 1,\ldots,m,\ \ell = 1,\ldots,L \\
x_{i\ell} &\le y_i && \text{for } i = 1,\ldots,m,\ \ell = 1,\ldots,L \\
x_{i\ell} &\in \{0,1\} && \text{for } i = 1,\ldots,m,\ \ell = 1,\ldots,L
\end{aligned}$$

$$
\begin{aligned}
y_i &\in \{0,1\} \qquad \text{for } i = 1,\ldots,m \\
z_{ik} &\in \{0,1\} \qquad \text{for } i = 1,\ldots,m\, k = 1,\ldots,K.
\end{aligned}
$$

We are currently investigating special classes of the FLP-CSF that admit constant factor approximation algorithms. In particular, we are focusing on the case where the service installation costs and the customer service costs are facility-invariant, i.e., $f_i^k = f^k$ and $s_{ik\ell} = s_{k\ell}$ for $i = 1,\ldots,m$. This would correspond to the UFLPP-DA with economic lot-sizing costs where the lot-sizing costs are facility-invariant. We also plan to investigate a similar class of the FLP-CSF as we did for the UFLPP in Theorem 8.1.3 and the UFLPP-DA in Theorem 8.3.4. In particular, we will consider the FLP-CSF where there exists an ordering of the facilities such that if $i < i'$ in this order, we have that $f_i^k \leq f_{i'}^k$ and $s_{ik\ell} \leq s_{i'k\ell}$ for all $k = 1,\ldots,K$ and $\ell = 1,\ldots,L$. This would correspond to the UFLPP-DA with economic lot-sizing costs where the lot-sizing costs in any time period at facility i are less expensive than the lot-sizing costs in the same time period at facility i'.

8.4 Chapter Summary and Future Research Directions

In this chapter, we have studied approximating models for integrating facility location and production planning decisions. It was shown that, in general, these problems (the UFLPP and the UFLPP-DA) are as hard as the set cover problem. Therefore, we have focused on identifying special cases of this problem class that can be approximated within a constant factor. One of the classes of problems for which we derived a new approximation algorithm can be viewed as a metric facility location problem where the facility costs are a concave function of the amount of demand assigned to the facility. We developed a greedy algorithm for this class of functions that generalizes an algorithm of Jain et al. [87]. We then were able to use this greedy algorithm together with the idea of cost-scaling to develop an approximation algorithm with a guarantee of 1.52.

An important direction for future research is to identify additional problems in our class that can be approximated within a constant factor. For example, it is shown in

Section 8.3 that the UFLPP-DA with economic lot-sizing costs can be formulated as a new class of facility location problems with service-installation costs. We are currently investigating whether constant factor approximation algorithms exist for special cases of this new facility location problem (the FLP-CSF) which would include generalizations of the two problem classes of the UFLPP-DA described in Theorems 8.3.3 and 8.3.4 with production setup costs (i.e., the UFLPP-DA with economic lot-sizing costs). It will also be interesting to investigate whether other special classes of the UFLPP have constant factor approximation algorithms including generalizations of the problems described in Theorem 8.1.2 and Theorem 8.1.3 that include production setup costs.

CHAPTER 9
EXACT ALGORITHMS FOR INTEGRATED FACILITY LOCATION AND PRODUCTION PLANNING PROBLEMS

In this chapter, we examine issues around solving the UFLPP and the UFLPP-DA exactly. We will focus on developing a branch and price algorithm to solve the UFLPP since the UFLPP-DA can be formulated as a particular class of the UFLPP, so that many of the theoretical results for the UFLPP also hold for the UFLPP-DA. We will explore the tightness of certain formulations of the UFLPP as well solving the pricing problem that arises in the algorithm.

In its current formulation, the UFLPP is a large-scale nonlinear integer programming problem. By reformulating the UFLPP as a set-partitioning problem, we remove the nonlinearity from the formulation at the expense of adding a large (exponential, in fact) number of variables to the problem. However, despite the exponential number of variables in the set-partitioning formulation, branch and price algorithms (where we solve the relaxation of the set-partitioning formulation through column generation) have been successfully applied to several (nonlinear) assignment and location problems. As mentioned in Section 2.1, Savelsbergh [136] uses a branch and price approach for solving the Generalized Assignment Problem (GAP) by formulating it as a set-partitioning problem. Subsequently, set partitioning formulations have led to effective solution methods for several nonlinear assignment and facility location problems; see Freling et al. [57], Shen et al. [141], Huang et al. [85], Shu et al. [144], or Romeijn et al. [129]. In all of these applications, the critical factor that determines the efficacy of the approach is the ability to solve the associated pricing problem. Despite the fact that this problem is, in many cases, NP-hard, they do often allow for efficient solution approaches that allow instances of the set partitioning problem of significant dimension to be solved in reasonable time.

Recall that the pricing problem that arises in a branch and price algorithm for a SCND problem is a problem of the form

$$\max_{x \in \{0,1\}^n} r^\top x - H(x).$$

In the case of the UFLPP, this problem is of the form of a production planning and customer selection problem (PPCSP),

$$\max\ r^\top x - \sum_{t=1}^{T} (P_t(p_t) + H_t(I_t))$$

subject to

$$\begin{aligned} I_{t-1} + p_t &= \sum_{j=1}^{n} d_{jt} x_j + I_t && \text{for } t = 1, \ldots, T \\ I_0 &= 0 \\ x_j &\in \{0,1\} && \text{for } j = 1, \ldots, n \\ p_t, I_t &\geq 0 && \text{for } t = 1, \ldots, T. \end{aligned}$$

The PPCSP was recently shown to be NP-hard by Van den Heuvel et al. [77] even for the case where P_t and H_t are that of the classic economic lot-sizing problem. However, a number of special classes of this problem are polynomially solvable. We will discuss the PPCSP in more detail in Section 9.2.

The remainder of the section is organized as follows. In Section 9.1 we remind the reader of the set-partitioning formulation (SPF) of the UFLPP. We discuss an important special case of the UFLPP, in particular, the case when the production and inventory costs are that of the classic economic lot-sizing problem. For this special case, we show that the SPF yields a tighter relaxation than the UFLPP itself. We use insight from this result to discuss the relationship of the SPF and the UFLPP in general. In Section 9.2, we discuss the PPCSP in more detail. In particular, we discuss the complexity of the problem along with several polynomially solvable subclasses. We conclude the chapter in Section 9.3 with a summary and future research directions.

9.1 A Set-Partitioning Formulation of the UFLPP

It is easy to see that any feasible solution to the UFLPP can be viewed as a partition of the customers into m subsets, each of which is assigned to a facility. Denote the number of distinct subsets that can be feasibly assigned to facility i by L_i. In particular, L_i is equal to the number of subsets of $\{1, \ldots, n\}$. Furthermore, let the binary vector $\alpha_{i\cdot}^{\ell}$ represent the ℓ^{th} subset associated with facility i, where $\alpha_{ij}^{\ell} = 1$ if customer j belongs to the subset and 0 otherwise. (We also refer to $\alpha_{i\cdot}^{\ell}$ as the ℓ^{th} *column* associated with facility i.) Furthermore, let $h_i(\alpha_{i\cdot}^{\ell})$ denote the cost associated with serving the set of customers given by $\alpha_{i\cdot}^{\ell}$ from facility i. If we then define the decision variable y_i^{ℓ} to be equal to one if facility i serves the ℓ^{th} associated subset and 0 otherwise, the set-partitioning formulation of the UFLPP reads:

$$\text{minimize} \sum_{i=1}^{m} \sum_{\ell=1}^{L_i} h_i(\alpha_{i\cdot}^{\ell}) y_i^{\ell}$$

subject to (SPF)

$$\sum_{i=1}^{m} \sum_{\ell=1}^{L_i} \alpha_{ij}^{\ell} y_i^{\ell} = 1 \qquad \text{for } j = 1, \ldots, n \tag{9–1}$$

$$\sum_{\ell=1}^{L_i} y_i^{\ell} = 1 \qquad \text{for } i = 1, \ldots, m \tag{9–2}$$

$$y_i^{\ell} \in \{0, 1\} \qquad \text{for } i = 1, \ldots, m,\ \ell = 1, \ldots, L_i.$$

Without loss of generality, we assume that $\alpha_{i\cdot}^{1} = 0$ represents the empty set, so that clearly $h_i(\alpha_{i\cdot}^{1}) = h_i(0) = 0$. Otherwise,

$$h_i(\alpha_{i\cdot}^{\ell}) = f_i + \sum_{j=1}^{n} d_j c_{ij} \alpha_{ij}^{\ell} + g_i(\alpha_{i\cdot}^{\ell})$$

where $g_i(\alpha_{i\cdot}^{\ell})$ is equal to the optimal solution value of an associated production planning problem:

$$\text{minimize} \sum_{t=1}^{T} (P_{it}(p_{it}) + H_{it}(I_{it}))$$

subject to

$$\begin{aligned} I_{i,t-1} + p_{it} &= \sum_{j=1}^{n} d_{jt}\alpha_{ij}^{\ell} + I_{it} && \text{for } t = 1, \ldots, T \\ I_{i0} &= 0 \\ p_{it}, I_{it} &\geq 0 && \text{for } t = 1, \ldots, T. \end{aligned}$$

For convenience, throughout the remainder of this chapter, we will refer to v(Problem) as the value of the optimal solution to Problem. We will also refer to ProblemR as the problem that is obtained if the integrality constraints in Problem are relaxed. It is easy to see that v(SPF) is equal to v(UFLPP). Therefore, v(SPFR) is a lower bound on v(UFLPP). For the remainder of this section, we will explore the relationship between v(SPFR) and v(UFLPPR). We begin by examining the special class of the UFLPP with economic lot-sizing costs (Section 9.1.1). We then use insight from these results to discuss the relationship between v(SPFR) and v(UFLPPR) in general (Section 9.1.2).

9.1.1 Tightness of the SPF for the UFLPP with Lot-Sizing Costs

In this section, we will examine the UFLPP where the production and inventory costs are that of the classic economic lot-sizing problem. In particular, $P_{it}(p_{it})$ is given by a fixed-charge setup cost, a_{it}, and a per unit production cost, b_{it}. The inventory cost function, $H_{it}(I_{it})$, is simply given by a per unit holding cost, h_{it}. For this UFLPP, we can remove the nonlinearity from the problem by formulating a mixed-integer linear programming problem. An intuitive formulation using binary variables to represent production setup decisions and continuous variables to represent production and inventory decisions would require a constraint that includes a so-called "big-M" term to ensure that we cannot produce in a time period at a facility without setting up production in that time period at the facility. Such constraints usually prevent the efficient solution of the optimization problem due to the poor quality of its linear programming relaxation bound. Therefore, we will instead use an alternative formulation of the problem that is based on an extended formulation of the standard economic lot-sizing problem. For the

lot-sizing problem, this extended formulation actually guarantees that an integral optimal solution to its linear programming relaxation exists; see, e.g., Krarup and Bilde [92]. The relaxation of this formulation is therefore expected to provide a better lower bound on the optimal solution of the problem.

In particular, we define a binary decision variable z_{it} that is equal to 1 if there is a production setup at facility i in period t, and a binary decision variable x_{ijst} that is equal to 1 if we meet the demand of customer j in time period t using production in time period s at facility i. Furthermore, we define H_{ist} to be the variable cost associated with producing a unit at facility i in time period s and hold it until time period t, i.e., $H_{ist} = b_{is} + \sum_{\tau=s}^{t-1} h_{i\tau}$. We then can formulate our uncapacitated facility location and lot-sizing problem (UFLLSP) as

$$\text{minimize} \sum_{i=1}^{m} f_i y_i + \sum_{i=1}^{m}\sum_{j=1}^{n} d_j c_{ij} x_{ij} + \sum_{i=1}^{m}\sum_{s=1}^{T}\left(a_{is} z_{is} + \sum_{t=s}^{T} H_{ist}\left(\sum_{j=1}^{n} d_{jt} x_{ijst}\right)\right)$$

subject to

$$\begin{aligned}
\sum_{i=1}^{m} x_{ij} &= 1 && \text{for } j = 1,\ldots,n && (9\text{–}3)\\
x_{ij} &= \sum_{s=1}^{t} x_{ijst} && \text{for } i = 1,\ldots,m,\ j = 1,\ldots,n,\ t = 1,\ldots,T && (9\text{–}4)\\
x_{ijst} &\le z_{is} && \text{for } i = 1,\ldots,m,\ j = 1,\ldots,n,\ t = s,\ldots,T,\ s = 1,\ldots,T && (9\text{–}5)\\
y_i, x_{ij} &\in \{0,1\} && \text{for } i = 1,\ldots,m,\ j = 1,\ldots,n\\
x_{ijst}, z_{is} &\in \{0,1\} && \text{for } i = 1,\ldots,m,\ j = 1,\ldots,n,\ t = s,\ldots,T,\ s = 1,\ldots,T.
\end{aligned}$$

Constraints (9–4) ensure that we meet the demand of each customer through production and inventory decisions at the facility to which it is assigned, while constraints (9–5) ensure that a production setup takes place at a facility in any time period that production takes place. In the set-partitioning formulation of the UFLLSP, recall that for $\ell =$

$2, \ldots, L_i$, we define the cost function to be equal to

$$h_i(\alpha_{i\cdot}^{\ell}) = f_i + \sum_{j=1}^{n} d_j c_{ij} \alpha_{ij}^{\ell} + g_i(\alpha_{i\cdot}^{\ell})$$

where $g_i(\alpha_{i\cdot}^{\ell})$ is equal to the optimal solution value of an associated economic lot-sizing problem:

$$\text{minimize } \sum_{t=1}^{T} a_{it} z_{it} + \sum_{t=1}^{T} \sum_{s=t}^{T} H_{ist} \left(\sum_{j=1}^{n} d_{jt} \alpha_{ij}^{\ell} \right) x_{ist}$$

subject to

$$\begin{aligned}
\sum_{s=1}^{t} x_{ist} &= 1 && \text{for } t = 1, \ldots, T \\
x_{ist} &\leq z_{is} && \text{for } t = s, \ldots, T,\ s = 1, \ldots, T \\
x_{ist}, z_{is} &\in \{0, 1\} && \text{for } t = s, \ldots, T,\ s = 1, \ldots, T.
\end{aligned}$$

It turns out that the SPF of the UFLLSP yields a tighter relaxation to the UFLLSP than the straightforward relaxation.

Theorem 9.1.1. *v(SPFR) ≥ v(UFLLSPR).*

Proof. We will show that any feasible solution to SPFR can be converted to a solution to UFLLSPR with the same objective function value. To this end, we first, let $x_{ist}^{\ell}, z_{it}^{\ell}$ be the values of the variables in an optimal solution of the optimization problem corresponding to $H_i(\alpha_{i\cdot}^{\ell})$. Then, consider any feasible solution y to SPFR and define a corresponding solution to UFLLSPR as follows:

$$\begin{aligned}
y_i &= \sum_{\ell=2}^{L_i} y_i^{\ell} && i = 1, \ldots, m \\
x_{ij} &= \sum_{\ell=2}^{L_i} \alpha_{ij}^{\ell} y_i^{\ell} && i = 1, \ldots, m,\ j = 1, \ldots, n \\
x_{ijst} &= \sum_{\ell=2}^{L_i} x_{ist}^{\ell} \alpha_{ij}^{\ell} y_i^{\ell} && i = 1, \ldots, m,\ j = 1, \ldots, n,\ t = s, \ldots, T,\ s = 1, \ldots, T \\
z_{it} &= \sum_{\ell=2}^{L_i} z_{it}^{\ell} y_i^{\ell} && i = 1, \ldots, m,\ t = 1, \ldots, T.
\end{aligned}$$

We first show that this solution is feasible to UFLLSPR. First, recall for $j = 1, \ldots, n$, we have:

$$\sum_{i=1}^{m} x_{ij} = \sum_{i=1}^{m} \sum_{\ell=2}^{L_i} \alpha_{ij}^{\ell} y_i^{\ell} = 1$$

since y satisfies constraint (9–1), so that our solution to UFLLSPR satisfies constraints (9–3). For $i = 1, \ldots, m$, $j = 1, \ldots, n$, and $t = 1, \ldots, T$:

$$\sum_{s=1}^{t} x_{ijst} = \sum_{s=1}^{t} \sum_{\ell=2}^{L_i} x_{ist}^{\ell} \alpha_{ij}^{\ell} y_i^{\ell} = \sum_{\ell=2}^{L_i} \alpha_{ij}^{\ell} y_i^{\ell} \sum_{s=1}^{t} x_{ist}^{\ell} = \sum_{\ell=2}^{L_i} \alpha_{ij}^{\ell} y_i^{\ell} = x_{ij}$$

implying that our solution to UFLLSPR satisfies constraint (9–4). For $i = 1, \ldots, m$, $j = 1, \ldots, n$, and $t = s, \ldots, T$, and $s = 1, \ldots, T$:

$$x_{ijst} = \sum_{\ell=2}^{L_i} x_{ist}^{\ell} \alpha_{ij}^{\ell} y_i^{\ell} \leq \sum_{\ell=2}^{L_i} x_{ist}^{\ell} y_i^{\ell} \leq \sum_{\ell=2}^{L_i} z_{is}^{\ell} y_i^{\ell} = z_{is}$$

so that constraints (9–5) are satisfied. It is clear that all variables defined are non-negative and that for $i = 1, \ldots, m$, $y_i \leq 1$ by the definition of y_i and constraints (9–2) of SPFR. For a fixed $i = 1, \ldots, m$ and $j = 1, \ldots, n$, consider:

$$x_{ij} = \sum_{\ell=2}^{L_i} \alpha_{ij}^{\ell} y_i^{\ell} \leq \sum_{\ell=2}^{L_i} y_i^{\ell} \leq 1$$

where the inequality holds by constraints (9–2) of SPFR. This implies that for any $i = 1, \ldots, m$, $j = 1, \ldots, n$, $s = 1, \ldots, t$, $t = 1, \ldots, T$ that x_{ij} and x_{ijst} are less than or equal to 1. For fixed $i = 1, \ldots, m$ and $t = 1, \ldots, T$, consider:

$$z_{it} = \sum_{\ell=2}^{L_i} z_{it}^{\ell} y_i^{\ell} \leq \sum_{\ell=2}^{L_i} y_i^{\ell} \leq 1.$$

Therefore, the corresponding solution is feasible to UFLLPSR. Now observe that the solution value of y with respect to facility i in SPFR is equal to

$$\begin{aligned} \sum_{\ell=1}^{L_i} h_i(\alpha_{i\cdot}^{\ell}) y_i^{\ell} &= \sum_{\ell=2}^{L_i} h_i(\alpha_{i\cdot}^{\ell}) y_i^{\ell} \\ &= \sum_{\ell=2}^{L_i} \left(f_i + \sum_{j=1}^{n} d_j c_{ij} \alpha_{ij}^{\ell} + g_i(\alpha_{i\cdot}^{\ell}) \right) y_i^{\ell} \end{aligned}$$

$$
\begin{aligned}
&= \sum_{\ell=2}^{L_i} \left(f_i + \sum_{j=1}^{n} d_j c_{ij} \alpha_{ij}^{\ell} + \sum_{s=1}^{T} \left(a_{is} z_{is}^{\ell} + \sum_{t=s}^{T} H_{ist} \left(\sum_{j=1}^{n} d_{jt} \alpha_{ij}^{\ell} x_{ist}^{\ell} \right) \right) \right) y_i^{\ell} \\
&= f_i \left(\sum_{\ell=2}^{L_i} y_i^{\ell} \right) + \sum_{j=1}^{n} d_j c_{ij} \left(\sum_{\ell=2}^{L_i} \alpha_{ij}^{\ell} y_i^{\ell} \right) \\
&\qquad + \sum_{t=1}^{T} \left(a_{it} \left(\sum_{\ell=2}^{L_i} z_{it}^{\ell} y_i^{\ell} \right) + \sum_{s=1}^{t} \sum_{j=1}^{n} d_{jt} H_{st}^{i} \left(\sum_{\ell=2}^{L_i} \alpha_{ij}^{\ell} x_{ist}^{\ell} y_i^{\ell} \right) \right) \\
&= f_i y_i + \sum_{j=1}^{n} d_j c_{ij} x_{ij} + \sum_{s=1}^{T} \left(a_{is} z_{is} + \sum_{t=s}^{T} H_{ist} \left(\sum_{j=1}^{n} d_{jt} x_{ijst} \right) \right).
\end{aligned}
$$

This implies that any solution to SPFR has a corresponding solution to UFLLSPR with the same cost. However, since the reverse is not necessarily true, we conclude that we are optimizing over a larger feasible region in UFLLSPR. Therefore, we have that

$$v(\text{SPFR}) \geq v(\text{UFLLSPR}).$$

□

We will use the idea behind the proof of Theorem 9.1.1 to discuss the relationship between $v(\text{SPFR})$ and $v(\text{UFLPPR})$ for more general production and inventory cost functions.

9.1.2 Tightness of the SPF for the UFLPP

In this section, we will focus on comparing the tightness of the SPF, i.e., the value $v(\text{SPFR})$, and the tightness of the UFLPP, i.e., the value $v(\text{UFLPPR})$. It is not difficult to see that a similar idea as the one in the proof of Theorem 9.1.1 will convert any feasible solution to SPFR to a feasible solution to UFLPPR. In particular, let $p_{it}^{\ell}, I_{it}^{\ell}$ be the production and inventory levels in an optimal solution of the optimization problem corresponding to $H_i(\alpha_{i\cdot}^{\ell})$. Then, consider any feasible solution y to SPFR and define a corresponding solution to UFLPPR as follows:

$$
\begin{aligned}
y_i &= \sum_{\ell=2}^{L_i} y_i^{\ell} && i = 1, \ldots, m \\
x_{ij} &= \sum_{\ell=2}^{L_i} \alpha_{ij}^{\ell} y_i^{\ell} && i = 1, \ldots, m;\ j = 1, \ldots, n
\end{aligned}
$$

$$
\begin{aligned}
p_{it} &= \sum_{\ell=2}^{L_i} p_{it}^{\ell} y_i^{\ell} \qquad i = 1, \ldots, m;\ t = 1, \ldots, T; \\
I_{it} &= \sum_{\ell=2}^{L_i} I_{it}^{\ell} y_i^{\ell} \qquad i = 1, \ldots, m;\ t = 1, \ldots, T.
\end{aligned}
$$

By a similar argument as in the proof of Theorem 9.1.1, this solution satisfies constraints (7–1) and (7–2). We also have that for all $i = 1, \ldots, m$ and $t = 1, \ldots, T$,

$$
\begin{aligned}
I_{i,t-1} + p_{it} &= \sum_{\ell=2}^{L_i} I_{i,t-1}^{\ell} y_i^{\ell} + \sum_{\ell=2}^{L_i} p_{it}^{\ell} y_i^{\ell} = \sum_{\ell=2}^{L_i} (I_{i,t-1}^{\ell} + p_{it}^{\ell}) y_i^{\ell} \\
&= \sum_{\ell=2}^{L_i} \left(\sum_{j=1}^{n} d_{jt} \alpha_{ij}^{\ell} + I_{it} \right) y_i^{\ell} = \sum_{\ell=2}^{L_i} \sum_{j=1}^{n} d_{jt} \alpha_{ij}^{\ell} y_i^{\ell} + \sum_{\ell=2}^{L_i} I_{it} y_i^{\ell} \\
&= \sum_{j=1}^{n} d_{jt} \left(\sum_{\ell=2}^{L_i} \alpha_{ij}^{\ell} y_i^{\ell} \right) + I_{it} = \sum_{j=1}^{n} d_{jt} x_{ij} + I_{it},
\end{aligned}
$$

so that this solution satisfies constraints (7–5). Therefore, this solution is feasible to UFLPPR. Now observe that the solution value of y with respect to facility i in SPFR is equal to

$$
\begin{aligned}
\sum_{\ell=1}^{L_i} h_i(\alpha_{i\cdot}^{\ell}) y_i^{\ell} &= \sum_{\ell=2}^{L_i} h_i(\alpha_{i\cdot}^{\ell}) y_i^{\ell} \\
&= \sum_{\ell=2}^{L_i} \left(f_i + \sum_{j=1}^{n} d_j c_{ij} \alpha_{ij}^{\ell} + g_i(\alpha_{i\cdot}^{\ell}) \right) y_i^{\ell} \\
&= \sum_{\ell=2}^{L_i} \left(f_i + \sum_{j=1}^{n} d_j c_{ij} \alpha_{ij}^{\ell} + \sum_{t=1}^{T} (P_{it}(p_{it}^{\ell}) + H_{it}(I_{it}^{\ell})) \right) y_i^{\ell} \\
&= f_i (\sum_{\ell=2}^{L_i} y_i^{\ell}) + \sum_{j=1}^{n} d_j c_{ij} \left(\sum_{\ell=2}^{L_i} \alpha_{ij}^{\ell} y_i^{\ell} \right) + \sum_{t=1}^{T} \left(\sum_{\ell=2}^{L_i} P_{it}(p_{it}^{\ell}) y_i^{\ell} \right) + \sum_{t=1}^{T} \left(\sum_{\ell=2}^{L_i} H_{it}(I_{it}^{\ell}) y_i^{\ell} \right) \\
&= f_i y_i + \sum_{j=1}^{n} d_j c_{ij} x_{ij} + \sum_{t=1}^{T} \left(\sum_{\ell=2}^{L_i} P_{it}(p_{it}^{\ell}) y_i^{\ell} \right) + \sum_{t=1}^{T} \left(\sum_{\ell=2}^{L_i} H_{it}(I_{it}^{\ell}) y_i^{\ell} \right) \\
&\leq f_i y_i + \sum_{j=1}^{n} d_j c_{ij} x_{ij} + \sum_{t=1}^{T} (P_{it}(p_{it}) + H_{it}(I_{it})),
\end{aligned}
$$

where the last inequality holds since the functions P_{it} and H_{it} are concave and constraints (9–2). Therefore, any solution to the SPFR can be converted to a solution to the

UFLPPR with a cost greater than or equal to the cost in the SPFR. This implies that we cannot draw any conclusion about the relationship between v(SPFR) and v(UFLPPR). On one hand, any solution to SPFR has an equivalent solution with higher cost in UFLPPR. On the other hand, we are optimizing over a larger feasible region in UFLPPR so that we may still end up with a solution that is cheaper than the optimal solution to SPFR.

Despite the fact that we cannot establish that the SPFR is tighter than the UFLPPR, there is still an inherent advantage about solving the SPFR rather than the UFLPPR. This is due to the fact that the SPFR is linear (although the pricing problems are nonlinear) and the UFLPPR is nonlinear. Therefore, if we can solve the pricing problem for the SPFR effectively, it can be expected that the branch and price algorithm will perform well.

9.2 The Production Planning and Customer Selection Problem

In this section, we will consider the pricing problem associated with the UFLPP. In particular, we will examine the production planning and customer selection problem (PPCSP). This problem can be intuitively described as follows: Given a set of potential customers, determine a subset of customers that maximize the net profits given that all demand is for each selected customer is met. Van den Heuvel et al. [77] prove that the PPCSP is NP-hard even when the production and inventory cost structures are that of the classic economic lot-sizing problem.

Theorem 9.2.1. *The PPCSP is NP-complete.*

However, there are several special subclasses of the PPCSP that are solvable in polynomial time. In the remainder of this section, we will describe several polynomially solvable subclasses of the PPCSP and their relationship to branch and price algorithms for the UFLPP.

9.2.1 Linear Production and Inventory Costs

If production and inventory costs are linear, then the PPCSP can be solved by determining the profitability of each customer in isolation. We first compute the optimal

cost of meeting a unit of demand in time period t, C_t^*, and then determining if r_j is greater than $\sum_{t=1}^{T} d_{jt}C_t^*$. Note that this can be accomplished in $O(nT)$ time. If we were to solve the UFLPP with linear production and inventory costs through a branch and price algorithm, we actually do not need to solve *any* production planning and customer selection problems. This is due to the fact that the production/inventory costs associated with assigning a customer to a facility can be determined in isolation. In particular, if we let C_{it}^* denote the optimal cost of serving a unit of demand in time period t at facility i, then we know the production and inventory costs associated with assigning customer j to facility i is $\sum_{t=1}^{T} d_{jt}C_{it}^*$. We can determine these costs for all customer/facility pairs in $O(nmT)$ time. We then know that the total cost of assigning customer j to facility i in the UFLPP is equal to

$$\sum_{t=1}^{T} d_{jt}(C_{it}^* + c_{ij}).$$

Therefore, this class of the UFLPP would simply reduce to a traditional UFLP, so that any exact solution method developed for the UFLP can be used to solve this class of the UFLPP.

9.2.2 Seasonal Demand Patterns

We will consider the problem PPCSP when the customers all share a common seasonality pattern, i.e., $d_{jt} = d_j\sigma_t$ for all customers $j = 1, \ldots, n$ and time periods $t = 1, \ldots, T$. We define the function $g(z)$ in a similar fashion as we did in Section 8.2, i.e., $g(z)$ is the optimal solution value to the problem

$$\text{minimize} \sum_{t=1}^{T} (P_t(p_t) + H_t(I_t))$$

subject to

$$\begin{aligned} I_{t-1} + p_t &= \sigma_t z + I_t \qquad \text{for } t = 1, \ldots, T \\ I_0 &= 0 \\ p_t, I_t &\geq 0 \qquad \text{for } t = 1, \ldots, T. \end{aligned}$$

Therefore, the PPCSP with seasonal demand patterns can be formulated as

$$\max_{x\in\{0,1\}^n} \; r^\top x - g(\sum_{j=1}^{n} d_j x_j)$$

Huang et al. [85] show that for any concave function g, the optimal solution to this problem can be found in polynomial time. Since Lemma 8.2.1 proves that our function g is concave, we may apply the algorithm of Huang et al. [85] to solve the PPCSP with seasonal demands. In particular, we reorder the customers in nonincreasing order of the ratio $\frac{r_j}{d_j}$ and Huang et al. [85] show that an optimal solution exists to the PPCSP with seasonal demands that selects the first j customers in this ordering. The ordering of the customers require $O(n \log n)$ time and we need to solve n production planning problems, so we can solve the PPCSP with seasonal demands in $O(n \max\{\log n, T^2\})$. When the production and inventory costs are that of an economic lot-sizing problem, the runtime of the approach is $O(n \max\{\log n, T \log T\})$.

9.2.3 Customer-Specific Prices

We next consider the problem PPCSP where the demand for customer j in period t is given by

$$d_{jt} = \alpha_t - \beta_t d_j$$

where α_t and β_t are time-dependent coefficients of a linear price-demand response curve and d_j can be viewed as the price set for customer j. Note that we consider each of the customer prices to be fixed, i.e., they are not themselves decision variables in our problem. Note also that, more generally, d_j could represent any customer-dependent but stationary function of price, allowing for more general price-demand response curves than the linear one present above.

We define the function $g(k, z)$ to be optimal solution to the following production planning problem

$$\sum_{t=1}^{T} (P_t(p_t) + H_t(I_t))$$

subject to

$$\begin{aligned} I_{t-1} + p_t &= \alpha_t k - \beta_t z + I_t && \text{for } t = 1, \ldots, T \\ I_0 &= 0 \\ p_t, I_t &\geq 0 && \text{for } t = 1, \ldots, T. \end{aligned}$$

The PPCSP with customer-specific prices can then be formulated as

$$\max_{x \in \{0,1\}^n} r^\top x - g(\sum_{j=1}^{n} x_j, \sum_{n=1}^{n} d_j x_j).$$

Now suppose that in the optimal solution to the PPCSP, we know that exactly k customers are selected. Then the problem reduces to

$$\max\ r^\top x - g(k, \sum_{n=1}^{n} d_j x_j)$$

subject to (CSP(k))

$$\begin{aligned} \sum_{j=1}^{n} x_j &= k \\ x &\in \{0,1\}^n. \end{aligned}$$

As in Lemma 8.2.1, it is easy to see that $g(k, \cdot)$ is a concave function. Since the extreme points of the relaxation of CSP(k) are integral and the algorithm from Section 6.1 returns an extreme point solution if the function is concave, we can solve CSP(k) in $O(n^2 \max\{\log n, T^2\})$ time for general concave production and inventory cost functions and $O(n^2 \max\{\log n, T \log T\})$ for economic lot-sizing costs. This immediately implies that, by solving CSP(k) for $k = 1, \ldots, n$, we can find an optimal solution to the PPCSP with customer-specific prices in $O(n^3 \max\{\log n, T^2\})$ and $O(n^3 \max\{\log n, T \log T\})$ for general concave cost functions and economic lot-sizing costs, respectively.

However, we can reduce the running time of this algorithm by a factor of n by employing similarities between the n problems of the form CSP(k) that need to be solved.

This yields a generalization of the approach of Section 6.1 to solve the PPCSP with customer-specific prices. This result is summarized in the following theorem:

Theorem 9.2.2. *The PPCSP with customer-specific prices can be solved in* $O(n^2 \max\{\log n, T^2\})$ *for the case of concave production/inventory cost functions and* $O(n^2 \max\{\log n, T \log T\})$ *for the case of economic lot-sizing production and inventory costs.*

Proof. We will employ the similarities between the n subproblems CSP(k) ($k = 1, \ldots, n$) that need to be solved to specializes the approach of Section 6.1. First, note that all extreme points of the continuous relaxation of the feasible region of each of these problems is integral, and applying the approach of Section 6.1 to the relaxation of CSP(k) for some fixed k yields an integral solution. Therefore, we can, without loss of optimality, relax the integrality constraints; we will, for the sake of convenience, refer to the relaxation of the problem as CSP(k) also. Recall that, in the approach of Section 6.1, we construct a collection of candidate solutions indexed by $\Delta = \{(i,j) : i < j \text{ and } r_i \neq r_j\}$ and denoted by $x^{(i,j)}$ that is guaranteed to contain an optimal solution to CSP(k). We denote the solution to the system

$$\begin{aligned} \lambda + \gamma d_i &= r_i \\ \lambda + \gamma d_j &= r_j \end{aligned}$$

by

$$\gamma^{(i,j)} = \frac{r_i - r_j}{d_i - d_j} \text{ and } \lambda^{(i,j)} = \frac{r_j d_i - r_i d_j}{d_i - d_j}.$$

We then determine the following sets

$$\begin{aligned} I_0^{(i,j)} &= \{\ell : r_\ell < \lambda^{(i,j)} + \gamma^{(i,j)} d_\ell\} = \{\ell : x_\ell^{(i,j)} = 0\} \\ I^{(i,j)} &= \{\ell : r_\ell = \lambda^{(i,j)} + \gamma^{(i,j)} d_\ell\} \\ I_1^{(i,j)} &= \{\ell : r_\ell > \lambda^{(i,j)} + \gamma^{(i,j)} d_\ell\} = \{\ell : x_\ell^{(i,j)} = 1\} \end{aligned}$$

that help determine the partial solutions $x^{(i,j)}$. Interestingly, these sets are *independent* of the value of k and (as is shown in Section 6.1), the time required to determine the sets of $I_0^{(i,j)}$, $I^{(i,j)}$, and $I_1^{(i,j)}$ for all $(i,j) \in \Delta$ in $O(n^2 \log n)$.

It now remains to complete the candidate solutions for each value of k by determining the values of the variables in $I^{(i,j)}$; we will refer to the corresponding solutions as $x^{(i,j)}(k)$. These solutions can be found by solving the subproblems

$$\max \sum_{\ell \in I_1^{(i,j)}} r_\ell + \sum_{\ell \in I^{(i,j)}} r_\ell x_\ell - g\left(k, \sum_{\ell \in I_1^{(i,j)}} d_\ell + \sum_{\ell \in I^{(i,j)}} d_\ell x_\ell\right)$$

subject to $(\mathrm{SP}^{(i,j)}(k))$

$$\begin{aligned} \sum_{\ell \in I^{(i,j)}} x_\ell &= k - |I_1^{(i,j)}| \\ 0 \le x_\ell &\le 1 \qquad\qquad \ell \in I^{(i,j)} \end{aligned} \tag{9–6}$$

Recall that $\mathrm{SP}^{(i,j)}(k)$ can be solved by first solving two linear knapsack problems, namely,

$$\max \text{ (or min)} \sum_{\ell \in I^{(i,j)}} d_\ell x_\ell$$

subject to $(\mathrm{KP}^{(i,j)}(k))$

$$\begin{aligned} \sum_{\ell \in I^{(i,j)}} x_\ell &= k - |I_1^{(i,j)}| \\ 0 \le x_\ell &\le 1 \qquad\qquad \ell \in I^{(i,j)} \end{aligned}$$

and selecting the solution of these two problems with the best objective function value to $\mathrm{SP}^{(i,j)}(k)$.

It is important to note that $\mathrm{SP}^{(i,j)}(k)$ only has a feasible solution, and thus only needs to be considered, for $|I_1^{(i,j)}| \le k \le |I_1^{(i,j)}| + |I^{(i,j)}|$. We will proceed by first adapting the algorithm from Section 6.1 by, for each $(i,j) \in \Delta$, solving all relevant solutions $x^{(i,j)}(k)$ consecutively, for $k = |I_1^{(i,j)}|, \ldots, |I_1^{(i,j)}| + |I^{(i,j)}|$. Suppose now that we have found a solution

for some value $|I_1^{(i,j)}| \leq k < |I_1^{(i,j)}| + |I^{(i,j)}|$ and consider the value $k+1$. Assuming that we have recorded the sorting of the variables in the linear knapsack problems $\mathrm{KP}^{(i,j)}(k)$ that led to their optimal solutions, then we need to only add in the next highest (or next lowest) value in the sorting in order to determine the solutions to $\mathrm{KP}^{(i,j)}(k+1)$. We therefore conclude that, for a fixed $(i,j) \in \Delta$, we can solve for all candidate solutions $x^{(i,j)}(k)$ in $O(|I^{(i,j)}| \log |I^{(i,j)}| + |I^{(i,j)}|\phi)$ time (where ϕ is the time required to solve the production planning problem), which follows from the fact that (i) we need to sort the variables in $I^{(i,j)}$ to solve all the problems of the form $\mathrm{KP}^{(i,j)}(k)$, and (ii) we need to evaluate the objective function value to $\mathrm{SP}^{(i,j)}(k)$ for $O(|I^{(i,j)}|)$ solutions, each of which requires the solution of a production planning problem.

Typically, we can expect to have $|I^{(i,j)}| = 2$ for all $(i,j) \in \Delta$. If this is the case, we obtain the desired result immediately from the fact that $|\Delta| = O(n^2)$. However, we may have $|I^{(i,j)}| > 2$ so that the situation becomes more complex. In general, let us define the sets $\Delta_i = \{j : (i,j) \in \Delta\}$ for $i = 1, \ldots, n$. Now if, for a given i, no variable occurs in $I^{(i,j)}$ for more than one $j \in \Delta_i$, we have that

$$O\left(\sum_{j \in \Delta_i} |I^{(i,j)}|\right) = O(n)$$

so that the desired result would follow again. Finally, Section 6.1 shows that, for a given i, the only variables that can occur in more than one of the sets $I^{(i,j)}$ are the ones for which the revenue/demand pair is identical to that of customer i. Denoting, for each i, the set of such customers by D_i, we obtain $\sum_{j \in \Delta_i} |I^{(i,j)}| = O(n + n|D_i|)$ and, since it is easy to see that the sets D_i are disjoint, we obtain the desired result:

$$\begin{aligned} O\left(\sum_{j=1}^{n} \sum_{j \in \Delta_i} (|I^{(i,j)}| \log |I^{(i,j)}| + |I^{(i,j)}|\phi)\right) &= O\left((\log n + \phi) \sum_{j=1}^{n} \sum_{j \in \Delta_i} |I^{(i,j)}|\right) \\ &= O(n^2 \max\{\log n, \phi\}). \end{aligned}$$

□

9.2.4 Customers Have Exactly One Non-Zero Demand Period

Recall that in the reformulation of the UFLPP-DA problem as a UFLPP, we end up with a UFLPP problem where each customer has exactly one non-zero demand period (where there is at most n customers in each time period with positive demand). The pricing problem that arises in this UFLPP is a PPCSP where each customer has exactly one non-zero demand period. Geunes et al. [66] considered a joint pricing and lot-sizing model where the demand to be satisfied depends on the price level. They assume that the revenue function in each period is piecewise linear and concave and show that the problem can be interpreted as an order selection problem. This means that this model is a special case of the PPCSP where each customer has exactly one period with positive demand. Geunes et al. [66] show that if the production and inventory costs are economic lot-sizing costs, this problem can be solved in $O(JT^2)$ time where J is the maximum number of customers having a positive demand in any single time period. Therefore, for the pricing problem that arises in solving the UFLPP-DA, we can solve the resulting PPCSP in $O(nT^2)$ time. We are currently investigating the complexity of the PPCSP where each customer has exactly one period with positive demand under more general concave production and inventory cost functions.

9.3 Chapter Summary and Future Research Directions

In this chapter, we have examined issues around solving the UFLPP to optimality. We have focused on the theoretical foundations of a branch and price algorithm to solve the it. We have shown that for the UFLPP with economic lot-sizing costs, the set-partitioning formulation of the problem yields a tighter relaxation than the mixed-integer formulation of the UFLPP inspired by the formulation of the economic lot-sizing of Krarup and Bilde [92]. Although, in general, we cannot conclude that a relationship between the tightness of the set-partitioning formulation and the continuous relaxation of the UFLPP, solving the UFLPP through a branch and price algorithm has the inherent advantage that we reduce the size of the nonlinear optimization problems

that we must solve. In the branch and price algorithm, we must repeatedly solve a production planning and customer selection problem. This problem is NP-hard, however, we discuss several practically important subclass of the PPCSP that can be solved in polynomial time. We are currently planning on implementing the branch and price algorithm to see its performance in practice. We also plan on investigating problems of the form UFLPP that are cyclic, i.e., the starting and ending inventories at each facility do not have to be zero, however, their levels must be the same.

CHAPTER 10
A SIMPLEX ALGORITHM FOR MINIMUM-COST NETWORK-FLOW PROBLEMS IN INFINITE NETWORKS

Network-flow problems constitute a very important class of problems in many application areas. For example, many production planning problems can be viewed as minimum-cost network-flow problems in appropriately defined networks. More generally, sequential decision making problems in which a sequence of decisions has to be made over time, such as equipment replacement problems and capacity expansion problems, can often be formulated as network-flow problems. If the planning horizon and action space of such a problem are finite, the resulting network formulation can be expected to be finite. However, in many cases there does not exist a natural finite planning horizon. We then obtain an infinite horizon optimization problem or, more specifically, a minimum-cost network-flow problem in an infinite network, i.e., a network with infinitely many nodes and arcs. In this chapter, we develop a network simplex method for solving such problems for a large class of infinite networks with countably many nodes and arcs. In addition, we do so in a non-standard but intuitive way that employs to the greatest extent possible the extensive knowledge of finite-dimensional linear programming and network optimization.

In this chapter, we study a class of infinite-dimensional minimum-cost network-flow problems. We employ the so-called natural dual optimization problem, which is analogous to the familiar finite-dimensional linear programming dual. Although this generally means that weak duality does not automatically hold, we show that a transversality condition similar to the one derived by Romeijn et al. [131] and Romeijn and Smith [130] yields both weak and strong duality. This circumvents the necessity of dealing explicitly with abstract dual spaces. We then develop a simplex method by characterizing basic primal solutions, constructing complementary dual solutions, computing reduced costs, and using these results to define an infinite-dimensional pivot operation. We derive convergence results and, as a consequence, show the existence of a pair of primal and dual solutions that satisfy strong duality. With this sound theoretical basis, we analyze a class of network-flow

problems where the flow balance constraints are inequalities rather than equalities. For this class, we show that there exist basic feasible solutions with the property that, from these, we may perform pivot operations in a finite amount of time. We show that if the network-flow problem is feasible, we can restrict ourselves to only basic solutions of this form, thereby ensuring that each pivot can be performed in finite time. We next develop a Phase I algorithm for this class of problems that returns a basic feasible solution that satisfies the desired property. Finally, we develop a hybrid Phase I and Phase II algorithm for which all iterations can be performed in a finite amount of time and derive convergence properties of this algorithm.

This chapter is organized as follows. In Section 10.1 we formally define our class of minimum-cost network-flow problems and lay the mathematical foundations for the infinite-dimensional simplex method by deriving duality results. In Section 10.2 we develop the infinite-dimensional simplex method and analyze its convergence properties. In Section 10.3 we analyze a class of problems for which the simplex method can be implemented in such a way that all pivots take only a finite amount of time. In Section 10.4 we conclude the chapter by discussing future research directions.

10.1 Problem Definition and Mathematical Foundations

10.1.1 An Infinite-Dimensional Minimum-Cost Network-Flow Problem

Consider an infinite directed network $G = (N, A)$, where $N = \{1, 2, 3, \ldots\}$ denotes the set of nodes while $A \subseteq N \times N$ denotes the set of directed arcs. We make the regularity assumption that each node in the network has finite in- and out- degree. In this chapter, we study a large class of *minimum-cost network-flow problems* defined on such networks. We first define a vector of costs $c \in \mathbb{R}^{|A|}$ (with typical element c_{ij}) that represents the unit costs of a shipment along the arcs of the network. Furthermore, we define a vector of integral supplies $b \in \mathbb{Z}^{|N|}$ (with typical element b_i) that represents the net amounts to be shipped from the nodes in the network. In particular, this means that (i) if $b_i > 0$ we say that i is a supply node and we need to ship the supply from node i;

(ii) if $b_i < 0$ we say that i is a demand node and the quantity $-b_i$ is its demand which needs to be delivered to node i; (iii) if $b_i = 0$ we say that i is a transshipment node. Finally, let $u \in \mathbb{Z}_+^{|A|}$ be a nonnegative vector of integral upper bounds on the flow on each of the arcs. This assumption implies that the feasible region of our problem as we will formulate it later is compact. Moreover, to ensure continuity of the objective function we assume that $\sum_{(i,j)\in A} |c_{ij}|u_{ij} < \infty$. Note that this condition, which we will refer to as the norm-assumption, is often satisfied naturally in network-flow problems that represent planning periods over time where costs are discounted and flows are uniformly bounded.

Our goal in this chapter is to develop a simplex algorithm that finds or approximates a minimum-cost solution that conserves flow at each node in the network. To allow for a formal analysis of, and the development of a simplex algorithm for, the minimum-cost network-flow problem described above, we will define a partial ordering of the nodes in the network G through the concept of *layers*. In particular, we will construct a sequence of sets L_n $(n = 1, 2, \ldots)$ where L_n denotes the set of nodes in the n^{th} layer. Moreover, we let $\bar{L}_n \equiv \cup_{n'=1}^{n} L_{n'}$ denote the set of all nodes in the first n layers. The essential property of a layering of the network G is that for all arcs that have *exactly* one node in $\bar{L}_n$ that node is in L_n. This implies that the only arcs that connect $\bar{L}_n$ to $N \backslash \bar{L}_n$ either originate or end in L_n. We can construct such a layering of the network by choosing an arbitrary node $\hat{\imath} \in N$ and choosing $L_1 \equiv \{\hat{\imath}\}$. We then recursively define L_{n+1} as follows:

$$L_{n+1} = \{j \in N \setminus \bar{L}_n : \exists\, i \in \bar{L}_n \text{ such that } (i,j) \in A \text{ or } (j,i) \in A\}.$$

We will also define, for each layer $n = 1, 2, \ldots$, the set of arcs A_n that are completely contained in that layer

$$A_n \;\equiv\; \{(i,j) \in A : i, j \in L_n\},$$

as well as the set of arcs that go from layer n to layer $n+1$

$$A_{n,n+1}^F \;\equiv\; \{(i,j) \in A :\; i \in L_n,\; j \in L_{n+1}\},$$

the set of arcs that go from layer $n+1$ to layer n

$$A^B_{n,n+1} \equiv \{(i,j) \in A : \ i \in L_{n+1}, \ j \in L_n\},$$

and the set of arcs that connect layers n and $n+1$

$$A_{n,n+1} \equiv A^F_{n,n+1} \cup A^B_{n,n+1}.$$

Finally, it will be convenient to define the set containing all arcs in layers $1, \ldots, n$ as well as the arcs connecting layer n to layer $n+1$:

$$\bar{A}_n = \{(i,j) \in A : i \in \bar{L}_n \text{ or } j \in \bar{L}_n\}.$$

Note that the procedure given above does not necessarily yield a unique layering of the network since a different choice of starting node $\hat{\imath}$ may lead to a different layering. The procedure therefore serves as just one way to define a layering of the network. In many applications, such as production planning or lot-sizing problems with a single supply and demand node in each period, a natural ordering of the network with the desired property may exist. Figure 10-1 shows an example of a network decomposed into layers.

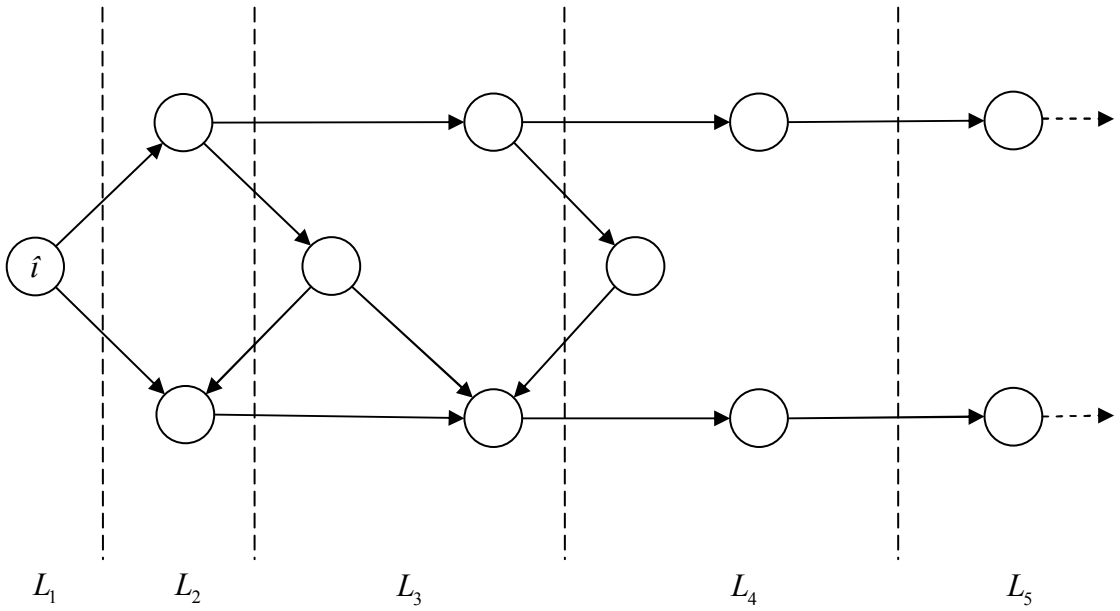

Figure 10-1. An illustration of layers.

10.1.2 Problem Formulation

In order to model the minimum-cost network-flow problem sketched above, we let $x \in \mathbb{R}^{|A|}$ (with typical element x_{ij}) denote a vector of flows in the network. With this notation, we can formulate the minimum-cost network-flow problem (P) in G as:

$$\text{minimize } C(x) \equiv \sum_{(i,j)\in A} c_{ij}x_{ij}$$

subject to

$$\begin{aligned} \sum_{j\in N:(i,j)\in A} x_{ij} - \sum_{j\in N:(j,i)\in A} x_{ji} &= b_i \qquad \text{for all } i \in N \qquad (10\text{–}1)\\ x_{ij} &\leq u_{ij} \qquad \text{for all } (i,j) \in A \qquad (10\text{–}2)\\ x_{ij} &\geq 0 \qquad \text{for all } (i,j) \in A. \end{aligned}$$

Constraints (10–1) are the usual *flow-balance constraints* while constraints (10–2) bound the flow on individual arcs. For convenience, we will denote the feasible region of the primal problem by $\mathcal{P}$. Our regularity assumptions on the structure of the network ensure that the flow-balance constraints are well defined. Under our assumptions on the problem data, Schochetman and Smith [137] show that the function C is well defined, i.e.,

$$C(x) = \lim_{n\to\infty} \sum_{(i,j)\in \bar{A}_n} c_{ij}x_{ij}$$

exists for all $x \in \mathcal{P}$ and is continuous with respect to the product topology on $\mathcal{P}$.

10.1.3 Duality

To guide a simplex algorithm towards the optimal solution it is important to establish a dual problem, say (D), to (P). Romeijn et al. [131] and Romeijn and Smith [130] define the natural dual of inequality-constrained linear programming problems. Extending this idea to our problem would yield the following dual problem:

$$\text{maximize } B(\pi,\theta) \equiv \limsup_{n\to\infty} \left(\sum_{i\in \bar{L}_n} b_i\pi_i - \sum_{(i,j)\in \bar{A}_n} u_{ij}\theta_{ij} \right)$$

subject to

$$\begin{aligned} \pi_i - \pi_j - \theta_{ij} &\leq c_{ij} && \text{for all } (i,j) \in A \\ \theta_{ij} &\geq 0 && \text{for all } (i,j) \in A. \end{aligned}$$

Here the vector π contains the dual variables (or multipliers) associated with constraint set (10–1) while the vector θ contains the dual variables (or multipliers) associated with the constraint set (10–2). Note that the dual objective function B can be written as

$$B(\pi, \theta) = \sum_{i \in N} b_i \pi_i - \sum_{(i,j) \in A} u_{ij} \theta_{ij}$$

if the infinite sums converge. Romeijn et al. [131] and Romeijn and Smith [130] provide a sufficient condition under which there exists a weak duality relationship between (P) and its natural dual problem. Unfortunately, rewriting our optimization problem using inequality constraints does not satisfy this condition. As an example, consider a minimum-cost network-flow problem in a network with nodes $i = 1, 2, \ldots$, arcs $(i, i+1)$ for $i = 1, 2, \ldots$, and demands $b_1 = 1$ and $b_i = 0$ for $i = 2, 3, \ldots$:

$$\text{minimize} \quad \sum_{i=1}^{\infty} \left(\frac{1}{2}\right)^i x_{i,i+1}$$

subject to

$$\begin{aligned} x_{1,2} &= 1 && \\ x_{i,i+1} - x_{i-1,i} &= 0 && \text{for } i = 2, 3, \ldots \\ x_{i,i+1} &\leq 1 && \text{for } i = 1, 2, \ldots \\ x_{i,i+1} &\geq 0 && \text{for } i = 2, 3, \ldots \end{aligned}$$

whose candidate dual would be:

$$\text{maximize} \quad -\pi_1 - \liminf_{n \to \infty} \sum_{i=1}^{n} \theta_{i,i+1}$$

subject to

$$\begin{aligned} \pi_i - \pi_{i+1} - \theta_{i,i+1} &\leq \left(\frac{1}{2}\right)^i && \text{for } i = 1, 2, \ldots \\ \theta_{i,i+1} &\geq 0 && \text{for } i = 1, 2, \ldots \end{aligned}$$

There is only one feasible solution to the primal problem: $x_{i,i+1} = 1$ for $i = 1, 2, \ldots$ with cost 1. Moreover, any solution of the form $\pi_i = \Pi$ and $\theta_{i,i+1} = 0$ for $i = 1, 2, \ldots$ is a feasible solution to the candidate dual with cost $-\Pi$. So for any $\Pi < -1$ we have a feasible solution to the candidate dual whose cost exceeds that of the optimal primal solution. In fact, we have that the candidate dual is unbounded. We conclude that the primal and candidate dual problem do not satisfy weak duality. We overcome this problem using a concept of *transversality* (see also Romeijn et al. [131] and Romeijn and Smith [130]):

Definition 10.1.1 (Transversality)**.** *Let $x \in \mathbb{R}^{|A|}$ and $\pi \in \mathbb{R}^{|N|}$. These vectors are said to satisfy the* transversality condition *if*

$$\lim_{n\to\infty} \left(\sum_{(i,j)\in A^F_{n,n+1}} x_{ij}\pi_j - \sum_{(i,j)\in A^B_{n,n+1}} x_{ij}\pi_i \right) = 0.$$

We now define the set of dual vectors π that satisfy the transversality condition with *all* primal feasible solutions:

$$\mathcal{T} = \left\{ \pi \in \mathbb{R}^{|N|} : \lim_{n\to\infty} \left(\sum_{(i,j)\in A^F_{n,n+1}} x_{ij}\pi_j - \sum_{(i,j)\in A^B_{n,n+1}} x_{ij}\pi_i \right) = 0 \text{ for all } x \in \mathcal{P} \right\}.$$

Adding the constraint that $\pi \in \mathcal{T}$ now yields yields our dual problem (D):

$$\text{maximize } B(\pi, \theta) = \limsup_{n\to\infty} \left(\sum_{i\in\bar{L}_n} b_i\pi_i - \sum_{(i,j)\in\bar{A}_n} u_{ij}\theta_{ij} \right)$$

subject to

$$\pi_i - \pi_j - \theta_{ij} \leq c_{ij} \qquad \text{for all } (i,j) \in A \tag{10–3}$$

$$\theta_{ij} \geq 0 \qquad \text{for all } (i,j) \in A \tag{10–4}$$

$$\pi \in \mathcal{T}. \tag{10–5}$$

For convenience, we will denote the feasible region of the dual problem by $\mathcal{D}$. The following theorem then shows that the pair of problems (P) and (D) satisfy weak duality.

Theorem 10.1.2 (Weak duality)**.** *Let $x \in \mathcal{P}$ be a feasible solution to the primal problem (P) and let $(\pi, \theta) \in \mathcal{D}$ be a feasible solution to the dual problem (D). Then*

$$B(\pi, \theta) \leq C(x).$$

Proof. Starting with the dual objective function we obtain:

$$\begin{aligned} B(\pi,\theta) &= \limsup_{n\to\infty} \left(\sum_{i\in\bar{L}_n} b_i\pi_i - \sum_{(i,j)\in\bar{A}_n} u_{ij}\theta_{ij} \right) \\ &\leq \limsup_{n\to\infty} \left(\sum_{i\in\bar{L}_n} b_i\pi_i - \sum_{(i,j)\in\bar{A}_n} x_{ij}\theta_{ij} \right) \end{aligned}$$

by constraint (10–2),

$$= \limsup_{n\to\infty} \left(\sum_{i\in\bar{L}_n} \left(\sum_{(i,j)\in A} x_{ij} - \sum_{(j,i)\in A} x_{ji} \right) \pi_i - \sum_{(i,j)\in\bar{A}_n} x_{ij}\theta_{ij} \right)$$

by constraint (10–1),

$$\begin{aligned} &= \limsup_{n\to\infty} \left(\sum_{(i,j)\in\bar{A}_n\setminus A^B_{n,n+1}} x_{ij}\pi_i - \sum_{(j,i)\in\bar{A}_n\setminus A^F_{n,n+1}} x_{ji}\pi_i - \sum_{(i,j)\in\bar{A}_n} x_{ij}\theta_{ij} \right) \\ &= \limsup_{n\to\infty} \left(\sum_{(i,j)\in\bar{A}_n} (\pi_i - \pi_j - \theta_{ij})\, x_{ij} + \sum_{(i,j)\in A^F_{n,n+1}} x_{ij}\pi_j - \sum_{(i,j)\in A^B_{n,n+1}} x_{ij}\pi_i \right) \\ &\leq \limsup_{n\to\infty} \left(\sum_{(i,j)\in\bar{A}_n} c_{ij}x_{ij} + \sum_{(i,j)\in A^F_{n,n+1}} x_{ij}\pi_j - \sum_{(i,j)\in A^B_{n,n+1}} x_{ij}\pi_i \right) \end{aligned}$$

by constraint (10–3),

$$\leq \limsup_{n\to\infty} \sum_{(i,j)\in\bar{A}_n} c_{ij}x_{ij} + \limsup_{n\to\infty} \left(\sum_{(i,j)\in A^F_{n,n+1}} x_{ij}\pi_j - \sum_{(i,j)\in A^B_{n,n+1}} x_{ij}\pi_i \right)$$

$$= \limsup_{n\to\infty} \sum_{(i,j)\in\bar{A}_n} c_{ij}x_{ij}$$

by constraint (10–5),

$$= C(x)$$

which proves the desired result. □

We next extend the notion of complementary slackness to our class of problems.

Definition 10.1.3 (Complementary slackness)**.** *Let x be a primal solution and let (π, θ) be a dual solution (where both are potentially infeasible). These solutions are said to satisfy* complementary slackness *if*

$$x_{ij}(c_{ij} - \pi_i + \pi_j + \theta_{ij}) = 0 \qquad \text{for all } (i,j) \in A \qquad (10\text{–}6)$$

$$\pi_i \left(b_i - \sum_{j\in N:(i,j)\in A} x_{ij} + \sum_{j\in N:(j,i)\in A} x_{ji} \right) = 0 \qquad \text{for all } i \in N \qquad (10\text{–}7)$$

$$\theta_{ij}(u_{ij} - x_{ij}) = 0 \qquad \text{for all } (i,j) \in A. \qquad (10\text{–}8)$$

This enables us to show that if a primal feasible solution and a dual feasible solution satisfy complementary slackness, they are optimal to their respective problems.

Theorem 10.1.4. *Let $x \in \mathcal{P}$ be a feasible solution to the primal problem (P) and let $(\pi, \theta) \in \mathcal{D}$ be a feasible solution to the dual problem (D). If this pair of solutions satisfies complementary slackness then $C(x) = B(\pi, \theta)$, x is an optimal solution to (P), and (π, θ) is an optimal solution to (D).*

Proof. Starting from the primal objective function value, we have

$$\begin{aligned} C(x) &= \limsup_{n\to\infty} \sum_{(i,j)\in\bar{A}_n} c_{ij}x_{ij} \\ &= \limsup_{n\to\infty} \sum_{(i,j)\in\bar{A}_n} (\pi_i - \pi_j - \theta_{ij})x_{ij} \end{aligned}$$

by the complementary slackness condition (10–6),

$$
\begin{aligned}
&= \limsup_{n\to\infty} \left(\sum_{(i,j)\in \bar{A}_n} x_{ij}\pi_i - \sum_{(j,i)\in \bar{A}_n} x_{ji}\pi_i - \sum_{(i,j)\in \bar{A}_n} \theta_{ij} x_{ij} \right) \\
&= \limsup_{n\to\infty} \left(\sum_{i\in \bar{L}_n} \left(\sum_{(i,j)\in A} x_{ij} - \sum_{(j,i)\in A} x_{ji} \right) \pi_i \right. \\
&\qquad \left. - \sum_{(j,i)\in A^F_{n,n+1}} x_{ji}\pi_i + \sum_{(i,j)\in A^B_{n,n+1}} x_{ij}\pi_i - \sum_{(i,j)\in \bar{A}_n} \theta_{ij} x_{ij} \right) \\
&= \limsup_{n\to\infty} \left(\sum_{i\in \bar{L}_n} b_i\pi_i - \sum_{(i,j)\in \bar{A}_n} u_{ij}\theta_{ij} - \sum_{(i,j)\in A^F_{n,n+1}} x_{ij}\pi_j + \sum_{(i,j)\in A^B_{n,n+1}} x_{ij}\pi_i \right)
\end{aligned}
$$

by the complementary slackness conditions (10–7) and (10–8),

$$
= \limsup_{n\to\infty} \left(\sum_{i\in \bar{L}_n} b_i\pi_i - \sum_{(i,j)\in \bar{A}_n} u_{ij}\theta_{ij} \right)
$$

by constraint (10–5),

$$
= B(\pi, \theta).
$$

Combining this with the weak duality result of Theorem 10.1.2 we obtain that x and (π, θ) are optimal solutions to (P) and (D), respectively. □

Constraint (10–5) seems, at first sight, somewhat restrictive and definitely inconvenient to deal with. The following proposition and corollary characterize two sets of mild sufficient conditions under which this constraint can be replaced by a somewhat stronger but easier and more intuitive one. Moreover, the results in Section 10.3 illustrate that even weaker conditions suffice for a large class of minimum-cost network-flow problems.

Proposition 10.1.5. *Suppose that the total flow on arcs between consecutive layers is uniformly bounded, i.e., there exists some S such that $\sum_{(i,j)\in A_{n,n+1}} x_{ij} \le S$ for all $n = 1, 2, \ldots$ and all feasible flows $x \in \mathcal{P}$. Then constraint (10–5) in the dual problem (D) can be replaced by*

$$
\lim_{n\to\infty} \max\{|\pi_i| : \ i \in L_n\} = 0. \tag{10–9}
$$

Proof. Let $x \in \mathcal{P}$ be a feasible solution to (P) and let (π, θ) be a dual solution that satisfies constraints (10–3), (10–4), and (10–9). We will show that $\pi \in \mathcal{T}$ so that constraint (10–5) is satisfied as well. First, note that

$$\begin{aligned}
&\left| \limsup_{n\to\infty} \sum_{(i,j)\in A^F_{n,n+1}} x_{ij}\pi_j - \sum_{(i,j)\in A^B_{n,n+1}} x_{ij}\pi_i \right| \\
&\leq \left| \limsup_{n\to\infty} \sum_{(i,j)\in A^F_{n,n+1}} x_{ij}\pi_j + \sum_{(i,j)\in A^B_{n,n+1}} x_{ij}\pi_i \right| \\
&\leq \limsup_{n\to\infty} \left(\sum_{(i,j)\in A_{n,n+1}} x_{ij} \right) \cdot \max\{|\pi_i| : i \in L_{n+1}\}.
\end{aligned}$$

The result now follows since

$$\limsup_{n\to\infty} \sum_{(i,j)\in A_{n,n+1}} x_{ij} \leq S < \infty.$$

□

The following corollary provides a mild sufficient condition under which the result of Proposition 10.1.5 is ensured.

Corollary 10.1.6. *Suppose that the upper bounds for arcs between layers as well as the number of arcs between successive layers are uniformly bounded, i.e., there exists some $U < \infty$ such that $u_{ij} \leq U$ for all $(i,j) \in A_{n,n+1}$ for all $n = 1, 2, \ldots$ and some $M < \infty$ such that $|A_{n,n+1}| \leq M$ for all $n = 1, 2, \ldots$. Then constraint (10–5) in the dual problem (D) can be replaced by constraint (10–9).*

Proof. The assumption in the corollary says that the amount of flow between layers n and $n+1$ is bounded from above by MU. This means that we should either have $\sum_{(i,j)\in A_{n,n+1}} x_{ij} \leq MU$, so that Proposition 10.1.5 applies with $S = MU$, or the primal problem (P) is infeasible. □

In the remainder of this chapter we will deal with situations where constraint (10–5) can indeed be replaced by constraint (10–9). Note that since constraint (10–9) is, in

general, stronger than constraint (10–5), the results of Theorem 10.1.2 and Theorem 10.1.4 still hold with the modified dual constraint in such cases.

The results obtained thus far do not imply that *strong duality* holds, i.e., the existence of an $x \in \mathcal{P}$ and $(\pi, \theta) \in \mathcal{D}$ such that $C(x) = B(\pi, \theta)$. In the next section we will address this issue through the development of our infinite network simplex method. In particular, we will show that there exists a pair of optimal primal and dual solutions that satisfy strong duality.

10.2 An Infinite Network Simplex Method

The network simplex method for finite-dimensional problems is of course based on the fact that an optimal extreme point solution exists. Romeijn et al. [128] show that the feasible region $\mathcal{P}$ of (P) contains an extreme point. However, it does not immediately follow that the optimization problem (P) has an extreme point optimal solution. In general, more restrictive assumptions on the space that the primal (and dual) optimization problems are posed in are required to conclude this directly. However, in the remainder of this section we will nevertheless develop a simplex method that moves through a sequence of extreme points of $\mathcal{P}$. Analogous to the finite-dimensional case, we construct a complementary dual solution to a given basic feasible (extreme point) solution to (P). We use this dual solution and the structure of primal basic feasible solutions to characterize reduced costs and the pivot step. Finally, we show that the sequence of extreme points generated by the algorithm converges in value to the optimal solution value of (P) and also study the issue of solution convergence.

10.2.1 Components of the Simplex Method

10.2.1.1 Extreme Points and Basic Primal Solutions

In finite minimum-cost network-flow problems, an extreme point solution can be characterized by a decomposition of the flow variables into basic and nonbasic variables where the (basic) arcs corresponding to the basic variables form a spanning tree in the graph. When each of the nonbasic variables is fixed at either its upper or its lower bound,

the values of the basic variables are uniquely defined. If these values satisfy their bound constraints the solution is a basic feasible solution. The network simplex method now proceeds by pivoting between basic feasible solutions until an optimal solution is found. A crucial component of a simplex method for infinite minimum-cost network-flow problems is therefore the characterization of extreme point solutions to that problem through basic and nonbasic variables or arcs. Romeijn et al. [128] provide such a characterization for the class of network-flow problems that we study in this chapter. They show that a feasible solution x is an extreme point solution if and only if the graph induced by the so-called free arcs, i.e., the arcs corresponding to variables for which the bound constraints are strictly satisfied, has the property that each node lies on no more than one infinite path. In addition, they showed that the free arc graph for each extreme point can be extended such that each node i lies on exactly one infinite path in the extended graph. We may then view this extended free arc graph as a basic characterization of the extreme point, where the variables corresponding to arcs in the extended free arc graph are basic and the other variables are nonbasic. It will be useful to form an intuitive analogy between the basic arc graph in the infinite network and the basic arc graph in a finite network. In particular, we could interpret the basic arc graph in the infinite network as a spanning tree "rooted" at a virtual node at infinity. Note that we could, in principle, also choose any node $i \in N$ as the root of this tree, as long as we then interpret infinite paths as being connected to one another through the virtual node at infinity.

10.2.1.2 Complementary Dual Solutions

This intuitive interpretation of the basic arc graph is useful in determining a complementary dual solution corresponding to a given basic feasible solution x. Following the theory on finite minimum-cost network-flow problems, we could choose any node of the network to be the root of the corresponding spanning tree and find all dual multipliers π by setting the value of π at the root node equal to zero and computing the other values uniquely through the equations $\pi_i - \pi_j = c_{ij}$ for all basic arcs (i, j). In the infinite case,

however, we will not choose any of the "true" nodes $i \in N$ to be the root node of the tree but, rather, choose the virtual node at infinity to be the "root". Note that we may interpret the value

$$\lim_{n\to\infty} \max\{|\pi_i| : \ i \in L_n\}$$

to be the "dual multiplier" of this virtual node, and setting it equal to zero will then ensure that constraint (10–9) is satisfied. It remains to be shown that this choice uniquely determines the values of all true dual multipliers π. We will do this as follows: we first construct a dual solution with the property that the dual multipliers along an infinite path in the basic arc graph converge, in some sense, to zero; we next show that this solution indeed satisfies constraint (10–9).

Recall that, in the basic arc graph, each node in the network has exactly one infinite path. However, these paths are not necessarily directed. Let B be a set of basic arcs corresponding to a basic feasible solution and let the infinite path in the basic arc graph from some node $i_0 \in N$ visit the sequence of unique nodes $i_0 - i_1 - i_2 - \cdots$, where $\delta_k = 0$ if $(i_k, i_{k+1}) \in B$ and $\delta_k = 1$ if $(i_{k+1}, i_k) \in B$ $(k = 0, 1, 2, \ldots)$. We can then define the following candidate set of dual multipliers $\tilde{\pi}$ for the nodes on this path. We start by setting $\tilde{\pi}_{i_0} = 0$. Next, we iteratively set

$$\tilde{\pi}_{i_{k+1}} = \begin{cases} \tilde{\pi}_{i_k} - c_{i_k i_{k+1}} & \text{if } (i_k, i_{k+1}) \in B \\ \tilde{\pi}_{i_k} + c_{i_{k+1} i_k} & \text{if } (i_{k+1}, i_k) \in B \end{cases} \qquad \text{for } k = 0, 1, 2, \ldots.$$

Equivalently, this means that we set

$$\tilde{\pi}_{i_k} = \sum_{\ell=0}^{k-1} \left(\delta_\ell c_{i_{\ell+1} i_\ell} - (1 - \delta_\ell) c_{i_\ell i_{\ell+1}} \right) \qquad \text{for } k = 0, 1, 2, \ldots.$$

Now note that

$$\sum_{\ell=0}^{\infty} \left| \delta_\ell c_{i_{\ell+1} i_\ell} - (1 - \delta_\ell) c_{i_\ell i_{\ell+1}} \right| \leq \sum_{(i,j)\in A} |c_{ij}| < \infty$$

so that $\lim_{k\to\infty} \tilde{\pi}_{i_k} \equiv \bar{\pi}(i_0)$ exists and is finite. We then define

$$\pi_{i_k} = \tilde{\pi}_{i_k} - \bar{\pi}(i_0) \qquad \text{for } k = 0, 1, 2, \ldots$$

so that $\lim_{k\to\infty} \pi_{i_k} = 0$. Even if some of the infinite paths in the basic arc graph overlap, this characterization will be consistent since (i) there exists only a single infinite path from each node; (ii) the basic arc graph is acyclic; and (iii) for any two nodes i_0 and j_0 whose infinite paths overlap, the contribution of the common arcs in the infinite paths to the values of $\bar{\pi}(i_0)$ and $\bar{\pi}(j_0)$ is the same.

It remains to be shown that this solution satisfies constraint (10–9). This does not immediately follow from the results so far since infinite paths in the basic arc graph do not have to pass through the layers in increasing order. Therefore, we define m_n to be the smallest index of a layer with the property that at least one infinite path from nodes in layers L_n and beyond pass through it. Noting that $A\backslash\bar{A}_{m_n-1}$ is the set of arcs entirely contained in layers $m_n, m_n + 1, \ldots$ this implies that

$$\max\{|\pi_i| : \ i \in L_n\} \leq \sum_{(i,j)\in A\backslash\bar{A}_{m_n-1}} |c_{ij}|.$$

Since $\sum_{(i,j)\in A} |c_{ij}| < \infty$, the result now follows if $\lim_{n\to\infty} m_n = \infty$. It is easy to see that $\lim_{n\to\infty} m_n$ exists since m_n is nondecreasing with respect to n. Now suppose that this limit is finite, say m. This means that, for all $n = m, m + 1, \ldots$, there must exist a node $i \in L_n$ with the property that the infinite path associated with i passes through $\bar{L}_m$. In other words, there are an infinite number of nodes with the property that their infinite path in the basic arc graph passes through L_m. On the other hand, since the number of nodes in L_m is finite and the degree of all these nodes is finite there can only be a finite number of *distinct* infinite paths that pass through L_m. Therefore, there exists a node in L_m that lies on two distinct infinite paths (thereby essentially forming what could be called a doubly-infinite path). However, this contradicts the nature of the basic arc graph. We thus conclude that constraint (10–9) is satisfied.

Finally, we select the following values for the dual multipliers θ:

$$\theta_{ij} = \begin{cases} 0 & \text{if } x_{ij} < u_{ij} \\ -c_{ij} + \pi_i - \pi_j & \text{if } x_{ij} = u_{ij} \end{cases} \qquad \text{for } (i,j) \in A.$$

We thus have constructed a dual solution (π, θ) that satisfies complementary slackness with respect to the basic feasible solution x. We will often refer to x and its dual solution (π, θ) as a primal/dual pair or to (π, θ) as the dual complement of x.

10.2.1.3 Reduced Costs

From the results in the previous section it is easy to see that the dual complement of any basic feasible solution to the primal problem satisfies the dual constraints (10–3) and (10–4) (at equality) for all arcs $(i,j) \in B$. However, consider a nonbasic arc $(i,j) \notin B$. We should then consider two cases:

(i) Suppose that $x_{ij} = 0$. Then constraint (10–4) is satisfied for arc (i,j), but constraint (10–3) may be violated, i.e., we may have that $c_{ij} - \pi_i + \pi_j < 0$.

(ii) Suppose that $x_{ij} = u_{ij}$. Then constraint (10–3) is satisfied for arc (i,j), but constraint (10–4) may be violated, i.e., we may have that $\theta_{ij} = -c_{ij} + \pi_i - \pi_j < 0$ or, equivalently, $c_{ij} - \pi_i + \pi_j > 0$.

Analogous to minimum-cost network-flow problems in finite networks, we now define the *reduced cost* associated with arc $(i,j) \in A$ as

$$c_{ij}^{\pi} = c_{ij} - \pi_i + \pi_j.$$

It immediately follows that $c_{ij}^{\pi} = 0$ if $(i,j) \in B$. Moreover, the complementary dual solution is infeasible if and only if there exists some arc $(i,j) \notin B$ such that (i) $x_{ij} = 0$ and $c_{ij}^{\pi} < 0$ or (ii) $x_{ij} = u_{ij}$ and $c_{ij}^{\pi} > 0$. On the other hand, if the complementary dual solution is feasible, Theorem 10.1.4 says that it, as well as the corresponding primal solution, is optimal.

10.2.1.4 Pivot Operations

We can now develop a pivot operation in which we move from a particular basic feasible solution to an adjacent basic feasible solution in the primal problem. Here, analogous to the finite-dimensional case, we define two basic feasible solutions to be adjacent if the basic arc graph of one can be obtained from the basic arc graph of the other by removing exactly one arc and replacing it by exactly one other arc. Then, a pivot operation is performed as follows. First, a nonbasic arc that violates the optimality conditions from Section 10.2.1.3 is selected. If this arc is added to the basic arc graph of the current solution one of two things will happen:

1. A (finite) cycle is created in the basic arc graph. In that case, we proceed exactly as in the finite network simplex method and change the flow along this cycle until one of the basic arcs in the cycle reaches one of its bounds. That arc is then removed from the basis, a new basic feasible solution is obtained, and a new dual complement is determined. We will call this a *finite cycle pivot*.
2. Two infinite paths are connected in the basic arc graph, creating a doubly-infinite path. As in the previous situation, we may now augment the flow along this doubly-infinite path while ensuring feasibility of the solution until one of the arcs in the doubly-infinite path reaches one of its bounds. This arc is then removed from the basis, thereby breaking the doubly-infinite path, a new basic feasible solution is obtained, and a new dual complement is determined. Note that, using the previously discussed informal interpretation of infinite paths leading to a virtual node at infinity, we may interpret a doubly-infinite path as an "infinite cycle". We will therefore call this an *infinite cycle pivot*.

10.2.2 Simplex Method

We now present our network simplex algorithm for an infinite network, given a basic feasible starting solution x and its dual complement (π, θ). Our simplex algorithm will proceed in iterations that correspond to the layers of the network. In particular, this

means that, in iteration n of the algorithm, we only consider nonbasic arcs for entry into the basis that are in $\bar{A}_n$, i.e., in one of the first n layers of the network. In the description of the algorithm below, x will at all times denote the "current solution".

Infinite Network Simplex Method

Step 0. Set $n = 1$.

Step 1. Find an arc in $(i, j) \in \bar{A}_n$ that violates the optimality conditions. If such an arc exists, go to Step 2; otherwise, go to Step 3.

Step 2. Perform a pivot to enter arc (i, j) into the basis and update the primal solution x and the dual variables of the nodes affected by the pivot. Return to Step 1.

Step 3. Set $x^n = x$, set $n = n + 1$, and return to Step 1.

It is clear that only after no nonbasic arc can be found in $\bar{A}_{n-1}$ (i.e., arcs in layers $1, \ldots, n-1$ and arcs connecting layer $n-1$ to layer n) will we consider nonbasic arcs in A_n (i.e., arcs contained within layer n) for entry into the basis. However, note that this does not mean that in iteration n we do not consider arcs in $\bar{A}_{n-1}$ at all. After a pivot step in which an arc from A_n has entered the basis, it is very well possible that a nonbasic arc in $\bar{A}_{n-1}$ becomes attractive for entry into the basis. In addition to the above structure of our algorithm, we will also assume that we use some pivot rule (such as Bland's rule) within each of the main iterations $n = 1, 2, \ldots$ to avoid cycling during the course of an iteration of our algorithm.

10.2.2.1 Value Convergence

In this section we will show that the network simplex method converges in value to the optimal cost. Recall that x^n denotes the solution obtained when no eligible arcs exist in $\bar{A}_n$. Further, we will let (π^n, θ^n) denote its dual complement. In other words, x^n is the current solution when we let $n = n + 1$. First, note that $C(x^n) \geq C(x^m)$ for $n < m$ since a pivot step will never increase the cost of the solution. Since we have assumed that an anti-cycling pivot selection rule is applied, the solution x^{n+1} will be reached in a finite

number of iterations starting from x^n since there are only a finite number of variables in the first $n+1$ layers.

Before turning to the issue of convergence of the sequences $\{C(x^n) : n = 1, 2, \ldots\}$ and $\{x^n : n = 1, 2, \ldots\}$ we first need to establish some preliminary results regarding the primal feasible region and a space in which the complementary dual solutions lie.

Lemma 10.2.1. *The primal feasible region $\mathcal{P}$ is compact in the product topology.*

Proof. This follows immediately from the fact that the feasible region is bounded and the Tychonoff Product Theorem (see, e.g., Munkres [105]). □

Lemma 10.2.2. *The closure of the set*

$$\{(\pi, \theta) \in \mathbb{R}^{|N|} \times \mathbb{R}^{|A|} : (\pi, \theta) \text{ is the dual complement of some extreme point in } \mathcal{P}\}$$

is compact in the product topology.

Proof. From the discussion in Section 10.2.1.2 it immediately follows that any complementary dual solution satisfies

$$-\infty < -\sum_{(i,j)\in A} |c_{ij}| \leq \pi_i \leq \sum_{(i,j)\in A} |c_{ij}| < \infty.$$

Bounds on the vector θ can easily be derived based on these bounds on π so that the result follows from the Tychonoff Product Theorem (see, e.g., Munkres [105]). □

We are now ready to prove one of our main convergence results as well as strong duality for our primal and dual problems.

Theorem 10.2.3 (Value Convergence/Strong Duality)**.** *The network simplex algorithm converges with respect to cost, that is*

$$\lim_{n\to\infty} C(x^n) = C^*$$

where C^* is the optimal solution value to the primal problem. Further, the primal and dual optimal solutions are attained, i.e., there exist an $x^* \in \mathcal{P}$ and $(\pi^*, \theta^*) \in \mathcal{D}$ that satisfy complementary slackness and are therefore optimal for their respective problems.

Proof. The results of Lemmas 10.2.1 and 10.2.2 imply that there exists a convergent subsequence of the primal/dual pairs $\{(x^n, \pi^n, \theta^n) : n = 1, 2, \ldots\}$, say indexed by $\{n_m : m = 1, 2, \ldots\}$. Let

$$\lim_{m \to \infty} (x^{n_m}, \pi^{n_m}, \theta^{n_m}) = (x^*, \pi^*, \theta^*).$$

Since for all $m = 1, 2, \ldots$, x^{n_m} and $(\pi^{n_m}, \theta^{n_m})$ satisfy complementary slackness, it follows that x^* and (π^*, θ^*) satisfy complementary slackness as well. In addition, from the description of the simplex method it can be seen that (π^*, θ^*) is dual feasible so that the pair x^* and (π^*, θ^*) satisfy strong duality by Theorem 10.1.4 and are optimal to their respective problems. Furthermore, since the sequence of costs $C(x^n)$ is nonincreasing and there is a subsequence of $\{x^n : n = 1, 2, \ldots\}$ that converges to an optimal cost solution we have

$$\lim_{n \to \infty} C(x^n) = C(x^*) = C^*$$

by the continuity of C. □

10.2.2.2 Solution Convergence

The results in the previous section show that our simplex method can obtain a solution that approximates the optimal cost arbitrarily closely. However, this does not imply that the sequence of solutions generated converges as well. In fact, we will show in this section that this cannot in general be ensured, but it does follow in case the optimal solution is unique. To this end, we first conclude from the proof of Theorem 10.2.3 that any limit point of the sequence of solutions generated by the simplex method is optimal.

Corollary 10.2.4. *Let $\bar{x}$ be the limit of a convergent subsequence of $\{x^n : n = 1, 2, \ldots\}$, i.e.,*

$$\lim_{m \to \infty} x^{n_m} = \bar{x}$$

for some subsequence $\{n_m : m = 1, 2, \ldots\}$. Then $\bar{x}$ is an optimal solution to (P). □

Theorem 10.2.5 (Solution Convergence)**.** *If x^* is the unique optimal solution to (P) then*

$$\lim_{n \to \infty} x^n = x^*.$$

Proof. Suppose the sequence $\{x^n : n = 1, 2, \ldots\}$ does not converge. This implies that there exists some $(i, j) \in A$ such that x_{ij}^n does not converge. In particular, this means that, for each $\epsilon > 0$, there exists a subsequence, say $\{n_m : m = 1, 2, \ldots\}$, such that $|x_{ij}^{n_m} - x_{ij}^*| > \epsilon$ for all $m = 1, 2, \ldots$. However, since $\mathcal{P}$ is compact there exists a subsequence of $\{x^{n_m} : m = 1, 2, \ldots\}$ that converges, say to $\bar{x}$. It is easy to see that $\bar{x}_{ij} \neq x_{ij}^*$ so that $\bar{x} \neq x^*$. On the other hand, Corollary 10.2.4 shows that $\bar{x}$ is an optimal solution to (P). This contradicts the uniqueness of the optimal solution so that the result follows. □

Establishing sufficient conditions for uniqueness of the optimal solution to an infinite-horizon optimization problem is notoriously hard, as discussed in Ryan and Bean [135], and most of the literature on infinite-horizon optimization has dealt with deriving results (such as the existence of solution or forecast horizons) under this assumption (see, for example, Bean and Smith [15, 16], Bés and Sethi [21], and Schochetman and Smith [139]). Theorem 10.2.5, which states that the Infinite Network Simplex Method converges solution-wise when the optimal solution is unique, is a further example of such a result. However, due to the additional network-flow structure in our problem class, we can establish the following sufficient condition under which the optimal solution is indeed unique:

Condition 10.2.6. *Define the cost of a finite or infinite undirected cycle in the graph $G = (N, A)$ as the sum of the arc costs in the cycle (or the negative thereof if the direction*

of the arc is reversed). If there does not exist any zero-cost undirected cycle then the optimal solution to (P) is unique.

The sufficiency of this condition follows immediately from the fact that it implies that, if we start with an optimal extreme point solution, then there does not exist a pivot that leaves the cost of the solution unchanged. Due to the convexity of $\mathcal{P}$, it follows that there cannot exist an alternative optimal solution.

Example

We now present an example in which the optimal solution is nonunique and the simplex method constructs a sequence of solutions that has two convergent subsequences, one corresponding to each of the alternative optimal solutions – so that the sequence of solutions does not converge. The problem is illustrated in Figure 10-2. The supplies are $b_1 = 1$ and $b_i = 0$ for $i = 2, 3, \ldots$. The costs of the arcs are given as follows: $c_{12} = c_{13} = 1$; $c_{2k,2k+2} = c_{2k+1,2k+3} = \frac{1}{2^k}$ for $k = 1, 2, \ldots$; $c_{26} = 0$; $c_{6k+2,6k+6} = -c_{6k,6k+2}$ for $k = 1, 2, \ldots$; and $c_{6k-1,6k+3} = -c_{6k-3,6k-1}$ for $k = 1, 2, \ldots$. The upper bounds are chosen to be $u_{ij} = 2$ for all $(i, j) \in A$ so that they are nonbinding in any feasible solution. The layers will then be defined as follows: $L_1 = \{1\}$, $L_2 = \{2, 3\}$, $L_3 = \{4, 5, 6\}$, $L_4 = \{7, 8, 9\}$, etc.

It is easy to see that this problem has two alternative optimal solutions. The first solution, say x^*, is to set $x^*_{1,2} = x^*_{6k-4,6k} = x^*_{6k,6k+2} = 1$ for $k = 1, 2, \ldots$ and $x^*_{ij} = 0$ for all other arcs. The second solution, say x^{**}, is to set $x^{**}_{1,3} = x^{**}_{6k-3,6k-1} = x^{**}_{6k-1,6k+3} = 1$ for $k = 1, 2, \ldots$ and $x^{**}_{ij} = 0$ for all other arcs. The value of both solutions is 1. Now suppose that we start the simplex method with the following initial solution: the basic variables are $x_{13} = x_{2k+1,2k+3} = 1$ for $k = 1, 2, \ldots$ and $x_{2k,2k+2} = 0$ for $k = 1, 2, \ldots$ and the remaining variables are nonbasic. The cost of this solution is equal to 2.

- *Iteration 1*:

  There is no nonbasic variable in layer 1 with negative reduced cost so the current solution is saved as x^1.

- *Iteration 2*:

In the first pivot, the nonbasic variable x_{26} enters the basis while the basic variable x_{46} leaves the basis, which leaves the actual solution and solution value unchanged. Next, the nonbasic variable x_{12} enters the basis while the basic variable x_{13} leaves the basis. The new solution has basic variables $x_{12} = x_{26} = x_{2k,2k+2} = 1$ for $k = 3, 4, \ldots$ and $x_{2k+1,2k+3} = 0$ for $k = 1, 2, \ldots$ while the remaining variables are nonbasic. The cost of this solution is equal to $1\frac{1}{4}$. There are now no nonbasic variables in layers 1 and 2 with negative reduced cost so the current solution is saved as x^2.

- *Iteration 3*:

  In the first pivot, the nonbasic variable x_{59} enters the basis while the basic variable x_{79} leaves the basis, which leaves the actual solution and solution value unchanged. Next, the nonbasic variable x_{13} enters the basis while the basic variable x_{12} leaves the basis. The new solution has basic variables $x_{13} = x_{35} = x_{59} = x_{2k+1,2k+3} = 1$ for $k = 4, 5, \ldots$ and $x_{24} = x_{26} = x_{2k,2k+2} = 0$ for $k = 3, 4, \ldots$. The cost of this solution is equal to $1\frac{1}{8}$. There are now no nonbasic variables in layers 1–3 with negative reduced cost so the current solution is saved as x^3.

- *Iteration n: $n \geq 4$*:

  These iterations proceed in a similar fashion.

It is easy to see that $\lim_{n\to\infty} x^{2n} = x^*$ and $\lim_{n\to\infty} x^{2n+1} = x^{**}$.

10.3 A Class of Problems with Finite-time Pivots

In Section 10.2 we developed a network simplex algorithm for solving minimum-cost network-flow problems in infinite networks described in Section 10.1. However, there are two obstacles that may in general prevent this algorithm to be implementable and used to approximate the solution to such problems:

(i) Even though each iteration of the simplex algorithm consists of only a finite number of pivots, each individual pivot may require an infinite number of operations.

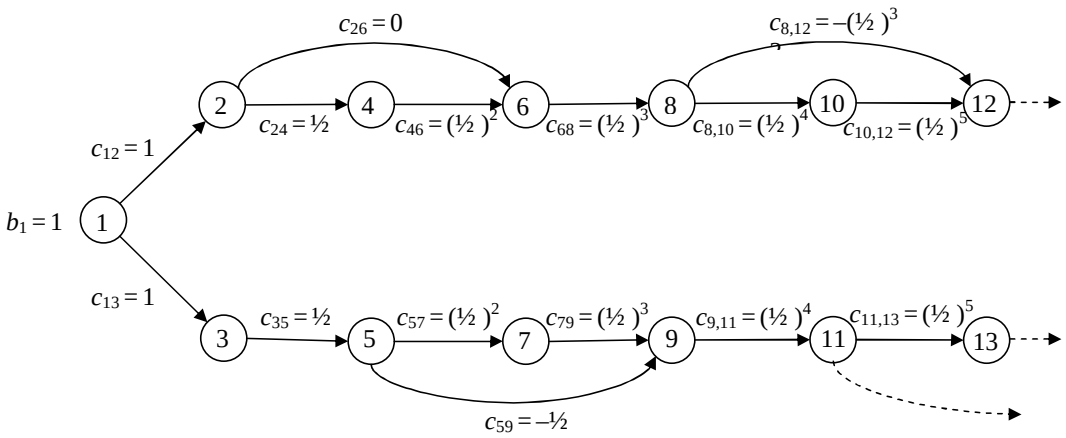

Figure 10-2. An example with multiple optimal extreme point solutions.

(ii) The algorithm requires a basic feasible initial solution, and it is not clear how such a solution may be obtained.

In this section, we will study a large class of network-flow problems for which both these obstacles can be overcome. In particular, we consider graphs G with the property that for each node $i \in N$, the set of its predecessors P_i (i.e., all nodes j with the property that there exists a directed path from j to i) is finite. We refer to this as the *finite predecessor assumption*. Note that this assumption will often be satisfied in sequential decision problems over time. For this class of graphs we will study problems for which the flow balance *equality* constraints (10–1) are replaced by *inequality* constraints, representing that it is feasible to (i) deliver more units than required to demand nodes; and (ii) use fewer than the supplied units from supply nodes. This class of problems clearly encompasses problems for which this change can be made without loss of optimality but also problems that are naturally formulated in this way. For example, in production planning (or lot-sizing) problems over an infinite horizon we can without loss of optimality assume that the demand in each period will be satisfied at equality; moreover, the supply at the supply nodes can be viewed as a production capacity and does not need to be fully used. We will reformulate the problem as an equivalent problem that falls within the problem class introduced in Section 10.1. We will then characterize the extreme points of the equivalent

problem for which pivot operations only require a finite amount of time and show that we can restrict attention to only these extreme points during the course of the simplex algorithm. We will next formulate a Phase I minimum-cost network-flow problem with the property that a basic optimal solution to this problem corresponds to a basic feasible solution to the actual problem of the desired type. We conclude this section and the chapter by proposing a hybrid algorithm that performs the Phase I algorithm for finding an initial basic feasible solution and the (say Phase II) algorithm for finding the optimal solution in parallel.

10.3.1 Problem Definition

We consider an almost identical problem setting as in Section 10.1 with the only modification being that the flow balance constraints (10–1) are replaced by

$$\sum_{j\in N:(i,j)\in A} x_{ij} - \sum_{j\in N:(j,i)\in A} x_{ji} \leq b_i \qquad \text{for all } i \in N. \tag{10–1$'$}$$

We will assume that the underlying network to the problem satisfies the finite predecessor assumption. We can then construct a layering of the network that has the property that $A^B_{n,n+1} = \emptyset$ for all n (and thus $A_{n,n+1} = A^F_{n,n+1}$) as follows. Recalling that we have the relationship $\bar{L}_n = \cup_{n'=1}^{n} L_{n'}$, we start by choosing a node $\hat{\imath}$ and setting $L_1 \equiv P_{\hat{\imath}}$. We next recursively define L_{n+1} as the set of nodes *not* in $\bar{L}_n$ that

(i) are connected by an arc to a node in layer n:

$$L'_{n+1} = \left\{ j \notin \bar{L}_n : \exists\, (i,j) \in A \text{ with } i \in \bar{L}_n \right\},$$

(ii) or are a predecessor of a node in L'_{n+1}.

More formally:

$$L_{n+1} = L'_{n+1} \cup \left(\cup_{i\in L'_{n+1}} P_i \backslash \bar{L}_n \right).$$

This implies that if $i \in \bar{L}_n$ then $P_i \subseteq \bar{L}_n$, so there do not exist any arcs (j,i) where $j \in L_{n+1}$ and $i \in L_n$, which in turn implies that $A^B_{n,n+1} = \emptyset$. In the remainder of this section we will assume that we have a layering of this form.

We will next reformulate this problem to fall within the framework of Section 10.1. A common approach in finite networks would be to simply add slack variables to the inequality constraints. These variables would then be interpreted as flows to a sink node that absorbs all excess supplies. However, in the case of an infinite network this sink node has infinite in-degree and, moreover, its demand would not be well defined. Therefore, we instead use an *infinite directed path* to represent the surplus flow: for every node $i \equiv i_0$ we introduce artificial (transshipment) nodes $i_1, i_2, i_3, \ldots$ and artificial (costless) arcs $(i_\ell, i_{\ell+1})$ for $\ell = 0, 1, 2, \ldots$. We will refer to this infinite path as the *path to infinity (PTI)* corresponding to node i. Figure 10-3 shows a network where the white nodes represent the nodes in the original graph and the grey nodes are the PTIs.

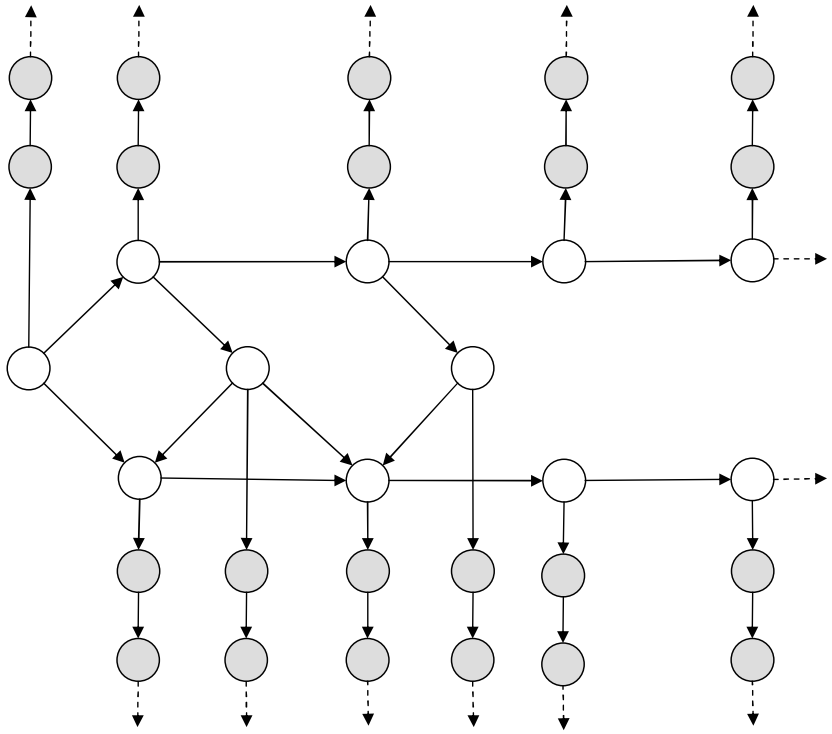

Figure 10-3. A reformulated network-flow problem.

We thus obtain the following minimum-cost network-flow problem (P′):

$$\text{minimize} \sum_{(i,j)\in A} c_{ij} x_{ij}$$

subject to

$$\begin{aligned}
\sum_{j\in N:(i,j)\in A} x_{ij} + x_{ii_1} - \sum_{j\in N:(j,i)\in A} x_{ji} &= b_i && \text{for all } i \in N \\
x_{i_\ell i_{\ell+1}} - x_{i_{\ell-1} i_\ell} &= 0 && \text{for all } i \in N;\ \ell = 1, 2, \ldots \\
0 \le x_{ij} &\le u_{ij} && \text{for all } (i,j) \in A \\
0 \le x_{i_{\ell-1} i_\ell} &\le \sum_{j\in P_i \cup \{i\}} b_j && \text{for all } i \in N;\ \ell = 1, 2, \ldots \quad (10\text{–}10)
\end{aligned}$$

For convenience, we will denote the graph corresponding to this reformulated problem by $G' = (N', A')$. Note that the upper bounds in constraints (10–10) are redundant but are included to ensure that (P′) falls within our class of minimum-cost network-flow problems described in Section 10.1. In particular, the predecessor sets of the nodes in N have not changed and it is easy to see that the predecessor set of i_ℓ contains only a finite number of elements more than P_i (for all $i \in N$ and $\ell = 1, 2, \ldots$). Since the artificial arcs are costless it immediately follows that the assumption on the cost function is satisfied as well. Moreover, as we will see later, we can assume without loss of generality that the dual variables $\pi_{i_\ell} = 0$ and $\theta_{i_{\ell-1} i_\ell} = 0$ for $i \in N$ and $\ell = 1, 2, \ldots$. Finally, since G' satisfies the finite predecessor condition it is easy to see that we can create a layering for which all arcs connecting layer n to layer $n+1$ are forward arcs, i.e., $A^B_{n,n+1} = \emptyset$. To ensure that our weak and strong duality results apply to (P′), we make the following assumption:

Assumption 10.3.1. *There exists some finite constant S' with the property that we may without loss of optimality restrict ourselves to solutions that satisfy*

$$\sum_{(i,j)\in A_{n,n+1}} x_{ij} \le S' \qquad \text{for } n = 1, 2, \ldots.$$

This assumption together with the observation regarding the dual variables corresponding to the artificial arcs and nodes then implies that constraint (10–5) in the dual of (P′) can be replaced by constraint (10–9). Note that this assumption is weaker than the assumption in Proposition 10.1.5 applied to (P′) since we only consider

flows between layers on the non-artificial arcs in G'. The assumption is immediately satisfied if, for example, the upper bounds on the arcs in A are bounded by U and the number of non-artificial arcs between consecutive layers is bounded by M (which is in fact the assumption in Corollary 10.1.6 applied to the graph G but much weaker than that assumption applied to the extended graph G').

10.3.2 Finite-time Pivots and the Finite Partition Property

The goal of this section is to characterize extreme points of (P$'$) for which we can perform pivot operations in finite time. First, note that, due to the addition of artificial nodes and arcs to the original network, an extreme point solution may have many different basic representations. In particular, if arc (i, i_1) does not carry any flow we can choose any of the arcs $(i_{\ell-1}, i_\ell)$ (for $\ell = 1, 2, \ldots$) to be nonbasic while the other artificial arcs are basic. However, as a convention, we will always choose all artificial arcs for $\ell = 2, 3, \ldots$ to be basic and only allow (i, i_1) to be nonbasic. (This in fact means that we do not really use or need most of the artificial arcs explicitly. However, their presence ensures that we can apply the results from Sections 10.1 and 10.2.) With this convention, a given set of basic arcs B' corresponding to a basic feasible solution implies a natural decomposition of the original set of nodes N into a countable number of equivalence classes of nodes that are connected through arcs in the basis. Each of these equivalence classes is of one of the following types:

(i) it is finite, i.e., all nodes in the class are connected to an infinite path consisting of arcs in $A' \setminus A$ (i.e., artificial arcs);

(ii) it is infinite, i.e., all nodes in the class are on an infinite path consisting of arcs in A (i.e., original arcs).

Basic feasible solutions that *only* contain equivalence classes of type (i) are of particular importance. We will say that such solutions satisfy the *finite partition property*:

Definition 10.3.2. *A basic feasible solution x to (P') is said to satisfy the* finite partition property *if its set of basic arcs B decomposes the node set N into a countable number of connected sets each containing a finite number of nodes.*

Given that a basic feasible solution x satisfies the finite partition property, any pivot operation in the infinite network simplex algorithm requires only a finite number of operations. This follows from the fact that adding a nonbasic arc to the basic set B' corresponding to x either creates a cycle or a doubly-infinite path. In the former case the result follows immediately. In the latter case, the doubly-infinite path contains only a finite number of arcs from the original network since the current basic solution satisfies the finite partition property. Further, note that on a particular node's PTI, the amount of flow and the upper bound on each arc are the same for each artificial arc. Therefore, we only need to consider the arcs on the path for which at least one node is in the original network. This implies that the pivot requires only a finite number of operations. Moreover, we immediately obtain that the basic solution reached after the pivot will again satisfy the finite partition property. It thus follows that we can implement each pivot in the Infinite Network Simplex Method in finite time provided we start the algorithm with a solution that satisfies the finite partition property. In the next section we will develop a Phase I algorithm that finds such a solution (thereby showing, by construction, that such a solution always exists). Further, note that the value of individual elements of a complementary dual solution to a basic feasible primal solution x that satisfies the finite partition problem can be found in finite time. In particular, since any infinite path in the basic solution consists of only a finite number of nodes and arcs in G, the values $\bar{\pi}(i_0)$ and the dual values of all nodes in a particular equivalence class of the set N can be found in finite time. As we mentioned before, this then immediately ensures that the dual variables corresponding to artificial nodes and arcs satisfy $\pi_{i_\ell} = 0$ and $\theta_{i_{\ell-1}i_\ell} = 0$ for $i \in N$ and $\ell = 1, 2, \ldots$.

10.3.3 Finding a Feasible Solution: Phase I

In Section 10.2 we have assumed that an initial basic feasible solution to the minimum-cost network-flow problem is available. In finite networks, we can easily formulate a Phase I minimum-cost network-flow problem in an extended network. This could be achieved, for example, by adding a transshipment node to the network and arcs from all nodes with $b_i > 0$ to the new node and from the new node to all nodes with $b_i < 0$. In addition, an arc from the new node to a sink node will account for any excess supply. A trivial basic solution is found by letting all new arcs be basic and used to supply all demands. The basis is then extended with original arcs having flow 0. When the costs of the new arcs are positive (say equal to 1) and the costs of all original arcs are set to zero, any basic optimal solution to the Phase I problem provides a basic feasible solution to the original problem.

In the infinite-dimensional case the situation is more complicated since the new transshipment node would have infinite in- and out- degree and transship an infinite amount of flow. In addition, we are interested in finding a basic feasible solution that satisfies the finite partition property. We therefore propose to construct an extended network-flow problem that will produce such a solution by employing the layering of the nodes and the arcs in graph G' as described in Section 10.1.1. Note that this layering corresponds to a layering of the underlying graph G as well. In the following, if we use the layering of G we will use the notation of Section 10.1.1 while if we use the layering of G' we will add a $'$ to the relevant symbol. The idea of the Phase I problem is to allow the satisfaction of all demands in a layer through supplies in the same layer, so that we may determine an initial solution to the Phase I problem easily. To this end, we extend the node set N' by adding an artificial supply node s^n with supply $S' + \sum_{i \in L_n} \max(b_i, 0)$ to each layer L_n ($n = 1, 2, \ldots$). In addition, we create an arc from node s^n to each demand node $i \in L_n$ (i.e., to each node with $b_i < 0$) as well as a PTI for s^n (consisting of nodes s^n_ℓ ($\ell = 1, 2, \ldots$) and corresponding arcs) with appropriate upper bounds. Observe that,

intuitively, the new nodes and arcs provide "shortcuts" to satisfying the demands in each layer. Assuming (P′) has a feasible solution, Assumption 10.3.1 implies that a feasible solution to the Phase I problem exists in which all demands are supplied within the corresponding layer, by setting $x_{s^n,i} = -b_i$ for all i where $b_i < 0$. A basis for this solution consists of, for each $n = 1, 2, \ldots$, (i) a spanning tree in the subnetwork induced by the set L_n, (ii) the PTIs of the supply and transshipment nodes in L_n, and (iii) the PTIs of the demand nodes in L_n except for their first arc.

We next construct a cost structure for the extended graph with the property that any basic optimal solution to the Phase I problem is a basic feasible solution to (P′) that satisfies the finite partition property. In addition, this cost structure will ensure that applying the Infinite Network Simplex Method to the Phase I problem yields a sequence of solutions that converges to its optimal solution. We start by letting the cost of all arcs in A' as well as the PTI of nodes s^n be equal to zero. In addition, we make the cost of all arcs (s^n, i) for all $i \in L_n$ such that $b_i > 0$ and for all $n = 1, 2, \ldots$ equal to $c'_{s^n i} = 1$. Any solution with cost 0 is then feasible to (P′). To find an optimal solution that satisfies the finite partition property, we next reimpose costs on arcs in $A_{n,n+1}$ for all $n = 1, 2, \ldots$. To this end, denote these arcs by $a_1, a_2, \ldots$ and assume that they are ordered in such a way that arcs between earlier layers have lower numbers than arcs between later layers: if $a_j \in A_{n,n+1}$ and $a_k \in A_{m,m+1}$ then $m > n$ implies that $k > j$. We then define

$$U_\ell = \sum_{\ell'=1}^{\ell} u_{a_{\ell'}}$$

and set the costs of these arcs equal to

$$c'_{a_\ell} = \alpha^{U_\ell} \qquad \text{for } a_\ell \in \bigcup_{n=1}^{\infty} A_{n,n+1}$$

where $\alpha < \frac{1}{2}$. This means that satisfying a unit of demand in layer n from an original supply node costs less than 1 so that it is always advantageous to supply a unit of demand through the original network rather than from the artificial supply nodes. In addition,

since the costs on arcs between layers are decreasing in n it is always advantageous to supply a demand through the latest possible layer.

The potential problem with this choice of cost function is that it does not satisfy the norm-assumption imposed in Section 10.1.1 that ensures that (i) a dual complementary solution to any primal basic feasible solution is well defined; and (ii) the primal objective function is continuous. However, we will see that these obstacles may be overcome for the Phase I problem. First, note that since the initial basic feasible solution to the Phase I problem has the finite partition property its dual complement is well defined. This means that we can apply the simplex algorithm developed in Section 10.2. The following theorem then not only shows that value and solution convergence hold in the sense that the results of Theorem 10.2.3 and Corollary 10.2.4 extend to the Phase I problem but, in addition, that the sequence of solutions generated by the infinite network simplex method converges. Further, we assume that after iteration n, if there exists a node i such that $b_i < 0$ and $x_{s_n,i} > 0$ then we terminate the algorithm since this implies that the original problem is infeasible.

Theorem 10.3.3. *Let x^n be the sequence of solutions generated by the network simplex algorithm applied to the Phase I problem starting from the initial solution described above. Then this sequence is convergent, i.e.,*

$$\lim_{n\to\infty} x^n = \tilde{x}$$

where $\tilde{x}$ is an optimal solution to the Phase I problem and, moreover, a basic feasible solution to (P') that satisfies the finite partition property.

Proof. Since the primal feasible region for the Phase I problem is compact we have that the sequence of solutions $\{x^n : n = 1, 2, \ldots\}$ contains a convergent subsequence. Denote the limit of such a convergent subsequence by $\bar{x}$.

Secondly, note that the initial solution contains no basic arcs in any of the sets $A_{n,n+1}$. This means that the first n iterations of the simplex algorithm will not affect

the flow on any arc that is not in $\bar{A}_n$. This observation combined with the cost structure of the Phase I problem implies that no dual variable will exceed 1 in absolute value throughout the algorithm. This means that the claim of Lemma 10.2.2 holds and, in turn, the result of Corollary 10.2.4 as well. This proves that $\bar{x}$ is a basic feasible solution to (P′).

Next, suppose that $\bar{x}$ does not satisfy the finite partition property. This means that there exists an infinite path of basic arcs in A. Moreover, due to the cost structure of the Phase I problem that ensures that it is optimal to satisfy demand from later layers, none of the arcs on this infinite path carry zero flow. But since the cost of sending a unit of flow along this path is strictly positive and the flows are integral this contradicts the optimality of $\bar{x}$. This proves that $\bar{x}$ satisfies the finite partition property.

Finally, suppose that the sequence of solutions $\{x^n : n = 1, 2, \ldots\}$ does not converge. This implies that there exists a subsequence $\{n_m : m = 1, 2, \ldots\}$ such that

$$\lim_{m\to\infty} x^{n_m} = \tilde{x} \neq \bar{x}.$$

Now suppose that there exists at least one arc in $\cup_{n=1}^{\infty} A_{n,n+1}$ on which $\bar{x}$ and $\tilde{x}$ differ, and let $\ell = \arg\min\{\ell' = 1, 2, \ldots : \bar{x}_{a_{\ell'}} \neq \tilde{x}_{a_{\ell'}}\}$ denote the smallest index of such an arc. Without loss of generality suppose that $\bar{x}_{a_{\ell'}} < \tilde{x}_{a_{\ell'}}$. Then

$$\sum_{\ell'=1}^{\infty} c'_{a_{\ell'}} \bar{x}_{a_{\ell'}} - \sum_{\ell'=1}^{\infty} c'_{a_{\ell'}} \tilde{x}_{a_{\ell'}} = c'_{a_\ell}(\bar{x}_{a_\ell} - \tilde{x}_{a_\ell}) + \sum_{\ell'=\ell+1}^{\infty} c'_{a_{\ell'}}(\bar{x}_{a_{\ell'}} - \tilde{x}_{e_{\ell'}}).$$

Moreover,

$$\begin{aligned}
\sum_{\ell'=\ell+1}^{\infty} c'_{a_{\ell'}}(\bar{x}_{a_{\ell'}} - \tilde{x}_{e_{\ell'}}) &\leq \sum_{\ell'=\ell+1}^{\infty} c'_{a_{\ell'}} u_{a_{\ell'}} \\
&= \sum_{\ell'=\ell+1}^{\infty} \alpha^{U_\ell} u_{a_{\ell'}} \\
&\leq \sum_{v=U_\ell+1}^{\infty} \alpha^v = \frac{\alpha^{U_\ell+1}}{1-\alpha} < \frac{\alpha^{U_\ell+1}}{\alpha} = \alpha^{U_\ell} = c'_{a_\ell}
\end{aligned}$$

since $0 < \alpha < \frac{1}{2}$, so that

$$\sum_{\ell'=1}^{\infty} c'_{a_{\ell'}} \bar{x}_{a_{\ell'}} \quad < \quad \sum_{\ell'=1}^{\infty} c'_{a_{\ell'}} \tilde{x}_{a_{\ell'}}$$

which contradicts the optimality of $\tilde{x}$. Therefore, we know that $\tilde{x}$ and $\bar{x}$ coincide on all arcs a_ℓ ($\ell = 1, 2, \ldots$). Now since the flow vectors are basic and therefore integral this implies that, for all $n = 1, 2, \ldots$, there exists some k_n with the property that any flows on arcs in $\cup_{n'=1}^{n-1} A_{n',n'+1}$ are not affected beyond iteration k_n of the Infinite Network Simplex Method. This, in turn, implies that beyond iteration k_n there will be no candidate pivot in $\bar{A}_n$ which yields that $\bar{x} = \tilde{x}$. □

In theory, the results of this section show that we can find a basic feasible solution that satisfies the finite partition property by solving a Phase I network-flow problem. This solution may then be used as the starting solution for the actual network-flow problem that we intend to solve, which we refer to as the Phase II problem. However, the major pitfall of this approach is of course the fact that it will take an infinite amount of time to find the starting solution for the Phase II problem, thereby preventing the practical implementation of the method. In the next section we will develop a hybrid Phase I/II simplex algorithm. One of the results obtained as the byproduct of the proof of Theorem 10.3.3 will be important in the development of this hybrid algorithm and we therefore state it as a corollary.

Corollary 10.3.4. *There exists a nondecreasing sequence* $\{k_n : n = 1, 2, \ldots\}$ *with the property that any flows on arcs in* $\bar{A}_n$ *are not affected beyond iteration* k_n *of the Infinite Network Simplex Method applied to the Phase I problem.*

10.3.4 A Hybrid Phase I/II Simplex Algorithm

The idea behind the hybrid Phase I/II simplex algorithm is based on Corollary 10.3.4, which says that the simplex algorithm applied to the Phase I problem yields a sequence of solutions with the property that the flows up to any layer eventually lock into the flows in the optimal solution to the problem. More precisely, let us denote the sequence of solutions that is generated by the Phase I algorithm by $\{y^n : n = 1, 2, \ldots\}$ and let

$y = \lim_{n\to\infty} y^n$ be the corresponding feasible solution to the original problem. Then, for all n there exists a value k_n such that the solution y^n restricted to the arcs in $\bar{A}_n$ is not affected in iterations $k_n + 1, k_n + 2, \ldots$. Now suppose that after performing the first k_n iterations of Phase I we consider any basic feasible solution to the Phase II problem, say x, that coincides with y^n on the arcs in $\bar{A}_n$ (which is guaranteed to exist since y is such a solution). Next, perform the first iteration of Phase II. If all pivots performed in this iteration only affect flow on arcs in $\bar{A}_n$, these pivots are precisely the pivots that we would have performed starting with initial solution y. Moreover, in this case the actual values of x beyond layer n are irrelevant for these Phase II pivots. Unfortunately, we will typically not know the sequence $\{k_n : n = 1, 2, \ldots\}$ and therefore we do not know when it would be valid to pause Phase I and perform an iteration of Phase II. The hybrid algorithm that we will propose can be viewed as providing a mechanism by which the values in this sequence are guessed and possible errors in these guesses are corrected for.

In order to best understand the hybrid algorithm, we first present a naive approach to applying Phases I and II in parallel. We will study convergence results for the resulting algorithm and then demonstrate that the hybrid algorithm is in fact a more efficient implementation of this naive approach.

Let $x^0 \equiv y$ be the starting point of Phase II in a sequential Phase I/II approach, and let $\{x^m : m = 0, 1, 2, \ldots\}$ denote the corresponding sequence of iterates. Now consider the solution y^n obtained after the first n iterations of Phase I and denote a corresponding, potentially infeasible, solution to Phase II by $x^{n,0}$. Using this solution as a starting point we now perform the first n iterations of the simplex algorithm applied to the Phase II problem, which yields the sequence of solutions $\{x^{n,m} : m = 1, \ldots, n\}$. Note that, since there are no basic arcs between layers n and $n + 1$ in this solution and since nonbasic arcs beyond layer n are not considered in the first n iterations, the fact that the flows beyond layer n are potentially infeasible does not cause any problems. Essentially, the algorithm

generates a sequence of solutions that can be visualized as follows:

$$\begin{pmatrix} x^{1,1} & - & - & - & - \\ x^{2,1} & x^{2,2} & - & - & - \\ x^{3,1} & x^{3,2} & x^{3,3} & - & - \\ x^{4,1} & x^{4,2} & x^{4,3} & x^{4,4} & - \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}.$$

The following theorem shows that the solutions $x^{n,m}$, for $n = m, m+1, \ldots$, approximate the solution x^m in the sense that the rows of this matrix converge elementwise to the sequence of simplex iterates $\{x^m : m = 1, 2, \ldots\}$ that would be obtained if Phase I and II were performed sequentially. In the following, it will be convenient to define, for any solution $\bar{y}$ ($\bar{x}$) to Phase I (Phase II), $\bar{y}(\tilde{A})$ ($\bar{x}(\tilde{A})$) to be the solution restricted to the arcs in $\tilde{A}$.

Theorem 10.3.5. *The iterates $\{x^{n,m} : m = 1, \ldots, n\}$ converge, for $n = 1, 2, \ldots$, to $\{x^m : m = 1, 2, \ldots\}$ in the following sense:*

$$\lim_{n \to \infty} x^{n,m} = x^m \qquad m = 1, 2, \ldots.$$

Proof. First, we define q_m to be

$$\max \left\{ m+1, \max_{m'=m+2,m+3,\ldots} \{m' : \exists\, i \in L_m,\ j \in L_{m'} \text{ connected through basic arcs in } x^0\} \right\}.$$

The value q_m provides an upper bound on the indices of the layers that can be affected in the first m iterations of the simplex method applied to the Phase II problem starting from x^0. The reason we examine nodes in layer $m+1$ rather than in layer m is that we allow pivot operations in the first m major iterations of the simplex algorithm to be performed on arcs in $A_{m,m+1}$. Note that $q_m < \infty$ for all m since x^0 satisfies the finite partition property. By Corollary 10.3.4 we know that, after performing k_{q_m} iterations of Phase I, the

flows in the first q_m layers are locked in, i.e.,

$$y^{k_{q_m}}(\bar{A}_{q_m}) = y^n(\bar{A}_{q_m}) = x^{n,0}(\bar{A}_{q_m}) \text{ for } n = k_{q_m}+1, k_{q_m}+2, \ldots.$$

For $n = k_{q_m}+1, k_{q_m}+2, \ldots$, pivots performed in the first m iterations of Phase II starting from $x^{n,0}$ can only affect flow up to layer q_m and thus the pivots performed in obtaining $x^{n,m}$ from $x^{n,0}$ for $n = k_{q_m}+1, k_{q_m}+2, \ldots$ are the same as the pivots performed in obtaining x^m from x^0, i.e.,

$$x^{k_{q_m},m}(\bar{A}_{q_m}) = x^{n,m}(\bar{A}_{q_m}) = x^m(\bar{A}_{q_m}) \text{ for } n = k_{q_m}+1, k_{q_m}+2, \ldots.$$

The desired result now follows since $\lim_{m\to\infty} q_m = \infty$. □

We next provide a convergence result on the costs associated with the matrix of iterates $\{x^{n,m} : m = 1, \ldots, n;\ n = 1, 2, \ldots\}$.

Theorem 10.3.6. *We have*

$$\lim_{m\to\infty} \lim_{n\to\infty} C(x^{n,m}) = C^*$$

where C^* is the optimal cost to the Phase II problem.

Proof. First, note that any solution $x^{n,m}$ satisfies the bound constraints in (P′). Since the function C is continuous on the set of all flows satisfying these constraints, Lemma 10.3.5, Theorem 10.2.3, and the fact that $\lim_{m\to\infty} q_m = \infty$ imply that

$$\lim_{m\to\infty} \lim_{n\to\infty} C\left(x^{n,m}(\bar{A}_{q_m})\right) = \lim_{m\to\infty} C\left(x^m(\bar{A}_{q_m})\right) = \lim_{m\to\infty} C(x^m) = C^*.$$

□

In the naive approach discussed so far we assume that we re-apply the simplex algorithm to the Phase II problem after each iteration of Phase I. However, we will next show that it is not necessary to always restart the simplex algorithm applied to the Phase II problem from $x^{n,0}$. In particular note that, for a fixed n, if the sequence of simplex iterations leading to $x^{n,m}$ *does not* affect flow in layers beyond, say, $f_{n,m}$ and, moreover,

the $(n+1)^{\text{st}}$ iteration of Phase I happens to *only* affect flows on arcs beyond layer $f_{n,m}$ then any pivots performed in Phase II to obtain $x^{n+1,m}$ from y^{n+1} are the same as the ones that were performed in Phase II to obtain $x^{n,m}$ from y^n. Therefore, in iteration $n+1$ of the hybrid algorithm we may essentially skip the first m' simplex iterations of Phase II, where m' is the largest index of a layer satisfying the above property, by an appropriate choice of starting point. More formally, let

$$p_n = \min\left\{0, \min_{m=1,\ldots,n}\{m : y^{n-1}(\bar{A}_m) \neq y^n(\bar{A}_m)\}\right\}$$

be the smallest index of a layer whose flows are affected during iteration n of Phase I. Furthermore, let

$$f'_{n,m} = \max\left\{m, \max_{j=m+1,\ldots,n}\{j : x_n^m\left(A_j \cup A_{j,j+1}\right) \neq x_n^{m-1}\left(A_j \cup A_{j,j+1}\right)\}\right\}$$

be the largest index of a layer that was affected by pivots performed in obtaining $x^{n,m}$ from $x^{n,m-1}$. Finally, let

$$f_{n,m} = \max_{j=1,\ldots,m} f'_{n,j}$$

be the largest index of a layer that was affected by pivots performed in obtaining $x^{n,m}$ from $x^{n,0}$.

We are now ready to present our hybrid algorithm:

Hybrid Algorithm

Step 0. Find y^0 and set $n = 1$.

Step 1. Perform the n^{th} iteration of Phase I starting from y^{n-1} and record p_n.

Step 2. Let

$$m' = \max\left\{0, \max_{m=1,\ldots n-1}\{m : p_n > f_{n-1,m}\}\right\}.$$

Then, for $m = 0, 1, \ldots m'$, set

$$\begin{aligned} x^{n,m}(\bar{A}_{f_{n-1,m'-1}}) &= x^{n-1,m}\left(\bar{A}_{f_{n-1,m'-1}}\right) \\ x^{n,m}\left(\bar{A}_n \backslash \bar{A}_{f_{n-1,m'-1}}\right) &= y^n\left(\bar{A}_n \backslash \bar{A}_{f_{n-1,m'-1}}\right) \end{aligned}$$

$$f'_{n,m} \;=\; f'_{n-1,m}.$$

Apply Phase II starting from $x^{n,m'}$, yielding $x^{n,m}$ for $m = m'+1, \ldots n$, recording the values of $f'_{n,m}$, and determine $f_{n,m}$.

Step 3. Increment n and return to Step 1.

By construction, the hybrid algorithm generates the same matrix of iterates $\{x^{n,m} : m = 1, \ldots, n;\ n = 1, 2, \ldots\}$ and therefore it shares the convergence properties with the naive approach. Our final result now strengthens the cost convergence result in Theorem 10.3.6 by showing that for all n there exists some a_n with the property that value convergence holds for the sequence of solutions $\{x^{n,a_n} : n = 1, 2, \ldots\}$. For convenience, let $\bar{A}_0 = \emptyset$.

Theorem 10.3.7. *There exists a nondecreasing sequence* $\{a_n : n = 1, 2, \ldots\}$ *such that*

$$\lim_{n\to\infty} C\left(x^{n,a_n}(\bar{A}_{a_n})\right) = C^*.$$

Proof. Consider the sequence $\{q_m : m = 1, 2, \ldots\}$ defined in the proof of Theorem 10.3.5. Then, let

$$a_n = \max\left\{0, \max_{m=1,\ldots n}\{m : k_{q_m} \le n\}\right\} \text{ for } n = 1, 2, \ldots.$$

By the definition of q_m we have that the sequence $\{a_n : n = 1, 2, \ldots\}$ is nondecreasing. Also, since $\lim_{m\to\infty} q_m = \infty$ and $\lim_{m\to\infty} k_{q_m} = \infty$ (by their respective definitions), it follows that $\lim_{n\to\infty} a_n = \infty$. Further, since $x^{n,0}(\bar{A}_{q_{a_n}}) = x^0(\bar{A}_{q_{a_n}})$ and pivots performed obtaining x^{n,a_n} only affect flow on arcs up to q_{a_n} we have

$$x^{n,a_n}(\bar{A}_{a_n}) = x^{a_n}(\bar{A}_{a_n}).$$

Since $\{x^{a_n} : n = 1, 2, \ldots\}$ is a subsequence of $\{x^m : m = 1, 2, \ldots\}$, this subsequence contains a convergent subsequence which we will denote by $\{\beta_n : n = 1, 2, \ldots\}$. That is, for each m we have

$$\lim_{n\to\infty} x^{\beta_n}(A_m \cup A_{m,m+1}) = x^*(A_m \cup A_{m,m+1})$$

where x^* is an optimal solution to (P′). Since $\lim_{n\to\infty} \beta_n = \infty$, this implies that

$$\lim_{n\to\infty} x^{\beta_n}(\bar{A}_{\beta_n}) = x^*.$$

By the continuity of $C(x)$, this in turn implies that

$$\lim_{n\to\infty} C(x^{\beta_n}(\bar{A}_{\beta_n})) = C(x^*) = C^*$$

and thus

$$\lim_{n\to\infty} C(x^{a_n}(\bar{A}_{a_n})) = C^*$$

as well. □

In fact, this proof immediately implies the following solution convergence result:

Corollary 10.3.8. *Consider the sequence* $\{x^{a_n}(\bar{A}_{a_n}) : n = 1, 2, \ldots\}$. *Then*

(i) any convergent subsequence converges to an optimal solution;

(ii) if the optimal solution to (P′) is unique, say x^*, *then*

$$\lim_{n\to\infty} x^{a_n}(\bar{A}_{a_n}) = x^*.$$

10.4 Chapter Summary and Future Research Directions

In this chapter, we have generalized the well-known network-simplex method for finite-dimensional networks to a large class of infinite networks. However, it would be interesting to study whether these results can be generalized or extended to a larger class of problems in which some of the simplifying assumptions that we have imposed are relaxed. We have assumed that each node in the network has finite degree. If the network represents a decision process over time with finite action space this assumption is often naturally satisfied. However, relaxing this assumption provides significant challenges since without it the flow balance constraints as we formulated them are not necessarily well defined. Another interesting research direction concerns the convergence properties of the simplex algorithm. We showed our simplex algorithm converges in value and, under the

additional assumption that the optimal solution is unique, solution convergence holds as well. We also provided an example with two distinct optimal solutions; in this case the simplex algorithm generates a sequence of solutions with two convergent subsequences – one corresponding to each optimal solution. It would be interesting and important to study whether a simplex algorithm can be constructed that converges even in case of alternative optima. Finally, we studied a class of network-flow problems that can be implemented in such a way that the pivots take only a finite amount of time. The identification of other classes of network-flow problems with this property would increase the practical applicability of the direct approach to solving minimum-cost network-flow problems in infinite networks.

CHAPTER 11
CONCLUSION

In this dissertation, we have examined several new classes of network design and network flow problems that arise in supply chain management. The network design problems that were examined are concerned with the assignment of customers to facilities where there is a facility cost function that is dependent on the set of customers assigned to the facility. We have studied approximate, heuristic, and exact methods to solve these network design problems. An interesting class of supply chain planning problems with customer selection arise as a subproblem in algorithms to solve the network design problems. In this class of problems, the supplier is interested in selecting a subset of the customers that maximizes its profits, i.e., the revenues received from the subset of customers minus the cost of serving them.

In Chapter 3 of this dissertation, we examined a nonlinear extension of the generalized assignment problem (the NL-GAP) where there is a nonlinear cost function for each facility whose argument is a linear function of the assignments to the facility. In the supply chain management context, we can interpret this function as the production cost of the amount of demand assigned to the facility. Often times, production cost functions in supply chain management are ill-structured (such as non-differentiable or discontinuous) so that standard optimization techniques may not be applicable. We overcome this difficulty in Chapter 3 by using theory from linear programming in developing new structural properties about some optimal solution to a class of nonlinear continuous optimization problems. This approach requires no assumptions about the nonlinear functions other than the existence of an optimal solution to the problem and leads to a novel set of dual multipliers for the problem. We use this approach in examining the continuous relaxation of the NL-GAP in Chapter 4 and developing a greedy procedure for it. We show that for facility-independent parameters, this greedy procedure is asymptotically optimal and feasible for any set of continuous facility cost functions. For facility-dependent parameters,

the greedy procedure is asymptotically optimal and feasible for convex functions. It will be an interesting area of future work to identify other classes of functions for which the procedure is asymptotically optimal and feasible. In Chapter 5, we use a geometrical interpretation of the results in Chapter 3 to develop methods to solve a class of convex programming problems. This class of problems includes the relaxation of the NL-GAP where the production cost functions are convex. The supply chain planning problem with customer selection that arises from the NL-GAP is an interesting nonlinear knapsack problem. In Chapter 6, we use the structural results from Chapter 3 to develop an efficient algorithm to solve the continuous relaxation of this nonlinear knapsack problem. Our algorithm uses complementary slackness conditions from linear programming in a novel way that requires no assumptions on the nonlinear portion of the problem. We extend the algorithm to similar classes of nonlinear multiple knapsack problems and nonlinear multiple-choice knapsack problems. These algorithms are shown to be extremely effective when compared to a commercial global optimization software package. In the future, it will be interesting to examine if the approach discussed in Chapter 3 can be successfully applied to other classes of nonlinear optimization problems.

In Chapter 7 of this dissertation, we have examined integrating facility location and production planning problems. In these problems, we must assign the set of customers to open facilities and determine the production and inventory levels of each open facility to ensure that we meet the demand assigned to the facility in each time period. In Chapter 8, we show that it is unlikely that, in general, these classes of integrated location and production planning problems can be approximated within a constant factor. Therefore, it is appropriate to focus on approximation algorithms for special cases of the problem. We have shown that several special cases of the problem can be approximated within a constant factor. One of these special cases gives rise to a metric uncapacitated facility location problem where each facility cost function is a concave function of the amount of demand assigned to the facility. For this new class of facility location problems,

we developed an approximation algorithm with a guarantee of 1.52. We are currently investigating whether constant factor approximation algorithms can be developed for other new classes of facility location problems that arise from this class of integrated facility location and production planning problems. In Chapter 9, we set the theoretical foundations of a branch and price algorithm to solve these integrated problems. The supply chain planning problem with customer selection that arises from this class of problems is an integrated production planning and customer selection problem. Although this class of problems was recently shown to be NP-hard, we discuss several practically relevant special cases of the problem that are polynomially solvable. It will be interesting to see if the theoretical foundations of the algorithm lead to a computationally efficient exact algorithm for these integrated location and production planning problems.

In Chapter 10 of this dissertation, we examined network flow problems in infinite networks. This class of problems encompasses many supply chain planning problems where a sequence of decisions needs to be made over an infinite horizon. Despite the typical mathematical pathologies associated with linear programming in infinite-dimensional spaces, we are able to extend the well-known network simplex method to infinite-dimensional spaces. We did so in a nonstandard but intuitively appealing way that, to the greatest extent possible, employs knowledge from the finite-dimensional network simplex method. Further, for a certain class of network flow problems in infinite networks, we showed that our network simplex method can be implemented in such a way that each pivot operation only requires a finite amount of time. In the future, the identification of other classes of problems with this property will increase the practical applicability of our infinite-dimensional network simplex method.

REFERENCES

[1] A. Aggarwal and J.K. Park. Improved algorithms for economic lot size problems. *Operations Research*, 41(3):549–571, 1993.

[2] H.-S. Ahn, M. Gumus, and P. Kaminsky. Pricing and manufacturing decisions when demand is a function of prices in multiple periods. *Operations Research*, 55(6):1039–1057, 2007.

[3] R.K. Ahuja, W. Huang, H.E. Romeijn, and D. Romero Morales. A heuristic approach to the multi-period single-sourcing problem with production and inventory capacities and perishability constraints. *INFORMS Journal on Computing*, 19:14–26, 2007.

[4] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin. *Network flows: Theory, algorithms, and applications*. Prentice-Hall, Englewood Cliffs, New Jersey, 1993.

[5] K.S. Alexander. Probability inequalities for empirical processes and a law of the iterated logarithm. *The Annals of Probability*, 12:1041–1067, 1984.

[6] E.J. Anderson and P. Nash. *Linear programming in infinite dimensional spaces*. Wiley, New York, New York, 1987.

[7] E.J. Anderson, P. Nash, and A.B. Philpott. A class of continuous network flow problems. *Mathematics of Operations Research*, 7:501–514, 1982.

[8] E.J. Anderson and A.B. Philpott. A continuous-time network simplex algorithm. *Networks*, 19:394–425, 1989.

[9] E. Arkin, D. Joneja, and R. Roundy. Computational complexity of uncapacitated multi-echelon production planning problems. *Operations Research Letters*, 8:61–66, 1989.

[10] I.S. Bakal, J. Geunes, and H.E. Romeijn. Market selection and inventory decisions under price-sensitive demand. Forthcoming in *Journal of Global Optimization*.

[11] V. Balachandran. An integer generalized transportation model for optimal job assignment in computer networks. *Operations Research*, 24:742–759, 1976.

[12] E. Balas and E. Zemel. An algorithm for large zero-one knapsack problems. *Operations Research*, 28:1132–1154, 1980.

[13] M.S. Bazaraa, H.D. Sherali, and C.M. Shetty. *Nonlinear programming: Theory and algorithms* (2nd ed.). John Wiley & Sons, New York, New York, 1993.

[14] J.C. Bean, J.R. Lohmann, and R.L. Smith. A dynamic infinite horizon replacement economy decision model. *The Engineering Economist*, 30:99–120, 1985.

[15] J.C. Bean and R.L. Smith. Conditions for the existence of planning horizons. *Mathematics of Operations Research*, 9:391–401, 1984.

[16] J.C. Bean and R.L. Smith. Conditions for the discovery of solution horizons. *Mathematical Programming*, 59:215–229, 1993.

[17] J.F. Benders, J.A. Keulemans, J.A.E.E. van Nunen, and G. Stolk. A decision support program for planning locations and allocations with the aid of linear programming. In C.B. Tilanus, O.B. de Gaus, and J.K. Lenstra, editors, *Quantitative methods in management: Cases studies of failures and successes*, chapter 4, pages 29–34. John Wiley & Sons, Chichester, 1986.

[18] D.P. Bertsekas. *Nonlinear programming* (2nd ed.). Athena Scientific, Belmont, Massachusetts, 1999.

[19] D.P. Bertsekas. *Convex analysis and optimization*. Athena Scientific, Nashua, New Hampshire, 2003.

[20] D. Bertsimas and J.N. Tsitsiklis. *Introduction to linear optimization*. Athena Scientific, Belmont, Massachusetts, 1997.

[21] C. Bès and S.P. Sethi. Concepts of forecast and decision horizons: Application to dynamic stochastic optimization problems. *Mathematics of Operations Research*, 13:295–310, 1988.

[22] G.R. Bitran and A.C. Hax. Dissagregation and resource allocation using convex knapsack problems with bounded variables. *Management Science*, 27:431–441, 1981.

[23] K.H. Borgwardt. *The simplex method: A probabilistic analysis*. Springer-Verlag, New York, New York, 1987.

[24] J.M. Borwein. Semi-infinite programming: How special is it? In A.V. Fiacco and K.O. Kortanek, editors, *Semi-infinite programming and applications*, pages 10–36. Springer Verlag, Berlin, Germany, 1983.

[25] K.M. Bretthauer and B. Shetty. The nonlinear resource allocation problem. *Operations Research*, 43(4):670–683, 1995.

[26] K.M. Bretthauer and B. Shetty. The nonlinear knapsack problem - algorithms and applications. *European Journal of Operational Research*, 138:459–472, 2002.

[27] K.M. Bretthauer and B. Shetty. A pegging algorithm for the nonlinear resource allocation problem. *Computers and Operations Research*, 29(5):505–527, 2002.

[28] P. Brucker. An $O(n)$ algorithm for quadratic knapsack problems. *Operations Research Letters*, 3(3):163–166, 1984.

[29] J. Byrka. An optimal bifactor approximation algorithm for the metric uncapacitated facility location problem. In *Proceedings of the 10th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, 2007.

[30] A. Caprara, D. Pisinger, and P. Toth. Exact solution of the quadratic knapsack problem. *INFORMS Journal on Computing*, 11(2):125–137, 1999.

[31] D.G. Cattrysse and L.N. Van Wassenhove. A survey of algorithms for the generalized assignment problem. *European Journal of Operational Research*, 60:260–272, 1992.

[32] A. Ceselli and G. Righini. An optimization algorithm for a penalized knapsack problem. *Operations Research Letters*, 34:394–404, 2006.

[33] M. Charikar and S. Guha. Improved combinatorial algorithms for facility location problems. *SIAM Journal on Computing*, 34(4):803–824, 2005.

[34] A. Charnes, W.W. Cooper, and K.O. Kortanek. Duality in semi-infinite programming and some works of Haar and Caratheodory. *Management Science*, 9:209–228, 1963.

[35] X. Chen and D. Simchi-Levi. Coordinating inventory control and pricing strategies with random demand and fixed ordering cost: The finite horizon case. *Operations Research*, 52:887–896, 2004.

[36] X. Chen and D. Simchi-Levi. Coordinating inventory control and pricing strategies with random demand and fixed ordering cost: The infinite horizon case. *Mathematics of Operations Research*, 29:698–723, 2004.

[37] F.A. Chudak and D.B. Shmoys. Improved approximation algorithms for the uncapacitated facility location problem. *SIAM Journal on Computing*, 33(1):1–25, 2003.

[38] S.A. Clark. An infinite-dimensional LP duality theorem. *Mathematics of Operations Research*, 28(2):233–245, 2003.

[39] M. Daskin, C. Coullard, and Z.-J. Shen. An inventory-location model: Formulation, solution algorithm, and computational results. *Annals of Operations Research*, 10:83–106, 2001.

[40] A. De Maio and C. Roveda. An all zero-one algorithm for a certain class of transportation problems. *Operations Research*, 19(6):1406–1418, 1971.

[41] S. Deng and C.A. Yano. Joint production and pricing decisions with setup costs and capacity constraints. *Management Science*, 52:741–756, 2006.

[42] R.J. Duffin, R.G. Jeroslow, and L.A. Karlovitz. Duality in semi-infinite linear programming. In A.V. Fiacco and K.O. Kortanek, editors, *Semi-infinite programming and spplications*, pages 50–62. Springer Verlag, Berlin, Germany, 1983.

[43] F. Duran. A large mixed integer production and distribution program. *European Journal of Operational Research*, 28:207–217, 1987.

[44] J.P. Dussault, J.A. Ferland, and B. Lemair. Convex quadratic programming with one constraint and bounded variables. *Mathematical Programming*, 36:90–104, 1986.

[45] M. Dyer. An $O(n)$ algorithm for the multiple-choice knapsack problem. *Mathematical Programming*, 29:57–63, 1984.

[46] M. Dyer and A. Frieze. Probabilistic analysis of the generalised assignment problem. *Mathematical Programming*, 55:169–181, 1992.

[47] A. Federgruen and M. Tzur. A simple forward algorithm to solve general dynamic lot sizing models with n periods in $O(n \log n)$ or $O(n)$ time. *Management Science*, 37(8):909–925, 1991.

[48] A. Federgruen and M. Tzur. The dynamic lot-sizing model with backlogging: A simple $O(n \log n)$ algorithm and minimal forecast horizon procedure. *Naval Research Logistics*, 40:459–478, 1993.

[49] A. Federgruen and M. Tzur. Minimal forecast horizons and a new planning procedure for the general dynamic lot-sizing model: Nervousness revisited. *Operations Research*, 42:456–468, 1994.

[50] A. Federgruen and M. Tzur. Fast solution and detection of minimal forecast horizons in dynamic programs with a single indicator of the future: Applications to dynamic lot-sizing models. *Management Science*, 41:874–893, 1995.

[51] U. Feige. A threshold of ln n for approximating set cover. *Journal of the ACM*, 45:634–652, 1998.

[52] M.L. Fisher and R. Jaikumar. A generalized assignment heuristic for vehicle routing. *Networks*, 11:109–124, 1981.

[53] L. Fleischer and E. Tardos. Efficient continuous-time dynamic network flow algorithms. *Operations Research Letters*, 23:71–80, 1998.

[54] B. Fleischmann. Designing distribution systems with transport economies of scale. *European Journal of Operational Research*, 70:31–42, 1993.

[55] M. Florian and M. Klein. Deterministic production planning with concave costs and capacity constraints. *Management Science*, 18:12–20, 1971.

[56] A. Forsgren, P.E. Gill, and M.H. Wright. Interior point methods for nonlinear optimization. *SIAM Review*, 44(4):525–597, 2002.

[57] R. Freling, H.E. Romeijn, D. Romero Morales, and A.P.M. Wagelmans. A branch-and-price algorithm for the multi-period single-sourcing problem. *Operations Research*, 51(6):922–939, 2003.

[58] A.F. Gabor and J.C.W van Ommeren. An approximation algorithm for a facility location problem with stochastic demands and inventories. *Operations Research Letters*, 34(3):257–263, 2006.

[59] A.F. Gabor and J.C.W van Ommeren. Approximation algorithms for facility location problems with a special class of subadditive cost functions. *Theoretical Computer Science*, 363(3):289–300, 2006.

[60] G. Gallego and G.J. van Ryzin. Optimal dynamic pricing of inventories with stochastic demand over finite horizons. *Management Science*, 40:999–1020, 1994.

[61] G. Gallo, P.L. Hammer, and B. Simeone. Quadratic knapsack problems. *Mathematical Programming*, 12:132–149, 1980.

[62] M.R. Garey and D.S. Johnson. *Computers and intractability.* W.H. Freeman and Company, New York, New York, 1979.

[63] S. van de Geer and L. Stougie. On rates of convergence and asymptotic normality in the multiknapsack problem. *Mathematical Programming*, 51:349–358, 1991.

[64] A.M. Geoffrion and G.W. Graves. Multicommodity distribution system design by Benders decomposition. *Management Science*, 20(5):822–844, January 1974.

[65] J. Geunes, Y. Merzifonluoğlu, and H.E. Romeijn. Capacitated procurement planning with price-sensitive demand and general concave revenue functions. Forthcoming in *European Journal of Operational Research.*

[66] J. Geunes, H.E. Romeijn, and K. Taaffe. Requirements planning with dynamic pricing and order selection flexibility. *Operations Research*, 54(2):394–401, 2006.

[67] J. Geunes, Z.-J. Shen, and H.E. Romeijn. Economic ordering decisions with market selection flexibility. *Naval Research Logistics*, 51(1):117–136, 2004.

[68] A. Ghate and R.L. Smith. Infinite linear programs: Characterizing extreme points through positive variables. Technical Report TR05-13, Department of Industrial and Operations Engineering, The University of Michigan, Ann Arbor, Michigan, 2005.

[69] A. Ghate and R.L. Smith. Duality theory for countably infinite linear programs. Technical report, Department of Industrial and Operations Engineering, The University of Michigan, Ann Arbor, Michigan, 2006.

[70] A. Ghate and R.L. Smith. A shadow simplex method for infinite linear programs. Technical report, Department of Industrial and Operations Engineering, The University of Michigan, Ann Arbor, Michigan, 2006.

[71] P.C. Gilmore and R.E. Gomory. A linear programming approach to the cutting-stock problem. *Operations Research*, 9:849–859, 1961.

[72] R. Grinold. Infinite horizon programs. *Management Science*, 18:157–170, 1971.

[73] R. Grinold. Finite horizon approximations of infinite horizon linear programs. *Mathematical Programming*, 12:1–17, 1977.

[74] R.C. Grinold and D.S.P. Hopkins. Duality overlap in infinite linear programs. *Journal of Mathematical Analysis and Applications*, 41:333–335, 1973.

[75] M.T. Hajiaghayi, M. Mahdian, and V.S. Mirrokni. The facility location problem with general cost functions. *Networks*, 42(1):42–47, 2003.

[76] F.W. Harris. How many parts to make at once. *Factory, The Magazine of Management*, 10:135–136, 152, 1913.

[77] W. van den Heuvel, O.E. Kundakcioglu, J. Geunes, H.E. Romeijn, T.C. Sharkey, and A.P.M. Wagelmans. Integrated market selection and production planning: Complexity and solution approaches. Technical report, Department of Industrial and Systems Engineering, University of Florida, Gainesville, Florida, 2007.

[78] W. van den Heuvel and A.P.M. Wagelmans. A polynomial time algorithm for a deterministic joint pricing and inventory model. *European Journal of Operational Research*, 170(2):463–480, 2006.

[79] J.B. Hiriart-Urruty. On optimality conditions in nondifferentiable programming. *Mathematical Programming*, 14:73–86, 1978.

[80] D.S. Hochbaum. *Approximation algorithms for NP-hard problems.* PWS Publishing Company, Boston, Massachusetts, 1997.

[81] D.S. Hochbaum and S.P. Hong. About strongly polynomial time algorithms for quadratic optimization over submodular constraints. *Mathematical Programming*, 69:269–309, 1995.

[82] B. Hoppe and E. Tardos. The quickest transshipment problem. *Mathematics of Operations Research*, 25(1):36–62, 2000.

[83] E. Horowitz and S. Sahni. Computing partitions with applications to the knapsack problem. *Journal of the ACM*, 21:277–292, 1974.

[84] R. Horst, P.M. Pardalos, and N.V. Thoai. *Introduction to global optimization* (2nd ed.). Kluwer Academic Publishers, Dordrecht, The Netherlands, 2000.

[85] W. Huang, H.E. Romeijn, and J. Geunes. The continuous-time single-sourcing problem with capacity expansion opportunities. *Naval Research Logistics*, 52(3):193–211, 2005.

[86] T. Ibaraki and N. Katoh, editors. *Resource allocation problems.* MIT Press, Cambridge, Massachusetts, 1988.

[87] K. Jain, M. Mahdian, E. Markakis, A. Saberi, and V. Vazirani. Greedy facility location algorithms analyzed using dual fitting with factor-revealing LP. *Journal of the ACM*, 50(6):795–824, 2003.

[88] P. Jones, J. Zydiak, and W. Hopp. Stationary dual prices and depreciation. *Mathematical Programming*, 41:357–366, 1988.

[89] A.H.G Rinnooy Kan, L. Stougie, and C. Vercellis. A class of generalized greedy algorithms for the multi-knapsack problem. *Discrete Applied Mathematics*, 42:279–290, 1993.

[90] K.C. Kiwiel. Breakpoint searching algorithms for the continuous quadratic knapsack problem. *Mathematical Programming*, 112:473–491, 2007.

[91] T.D. Klastorin. On a discrete nonlinear and nonseparable knapsack problem. *Operations Research Letters*, 9:233–237, 1990.

[92] J. Krarup and O. Bilde. Plant location, set covering, and economic lot size: An $O(mn)$-algorithm for structured problems. In *Numerische Methoden Bei Optimierungsaufgaben, Band 3: Optimierung Bei Graph-Theoretischen Und Ganzahligen Problemen*, pages 155–180. 1977.

[93] H. Kunreuther and L. Schrage. Optimal pricing and inventory decisions for non-seasonal items. *Econometrica*, 2:193–205, 1971.

[94] H. Kunreuther and L. Schrage. Joint pricing and inventory decisions for constant priced items. *Management Science*, 19:732–738, 1973.

[95] R. Levi, R.O. Roundy, and D.B. Shmoys. Primal-dual algorithms for deterministic inventory problems. *Mathematics of Operations Research*, 31(2):267–284, 2006.

[96] D.G. Luenberger. *Optimization by vector space methods.* Wiley, New York, New York, 1969.

[97] T.L. Magnanti and D. Stratila. Strongly polynomial primal-dual algorithms for concave cost combinatorial optimization problems. Technical report, MIT, Cambridge, Massachusetts, 2007.

[98] M. Mahdian, Y. Ye, and J. Zhang. Approximation algorithms for metric facility location problems. *SIAM Journal on Computing*, 36(2):411–432, 2006.

[99] S. Martello and P. Toth. An upper bound for the zero-one knapsack problem and a branch and bound algorithm. *European Journal of Operational Research*, 1:169–175, 1977.

[100] S. Martello and P. Toth. An algorithm for the generalized assignment problem. In J.P. Brans, editor, *Operational research*, pages 589–603. North-Holland, Amsterdam, The Netherlands, 1981.

[101] S. Martello and P. Toth. *Knapsack problems, algorithms and computer implementations*. John Wiley & Sons, New York, New York, 1990.

[102] M. Meanti, A.H.G Rinnooy Kan, L. Stougie, and C. Vercellis. A probabilistic analysis of the multi-knapsack value function. *Mathematical Programming*, 46:237–247, 1990.

[103] Y. Merzifonluoğlu, J. Geunes, and H.E. Romeijn. Integrated capacity, demand, and production planning with subcontracting and overtime options. *Naval Research Logistics*, 54(4):433–447, 2007.

[104] J.J. Moré and S.A. Vavasis. On the solution of concave knapsack problems. *Mathematical Programming*, 49:397–411, 1991.

[105] J.R. Munkres. *Topology*. Prentice-Hall, Englewood Cliffs, New Jersey, 1975.

[106] R.M. Nauss. An efficient algorithm for the 0-1 knapsack problem. *Management Science*, 23:27–31, 1976.

[107] G.L. Nemhauser and L.A. Wolsey. *Integer and combinatorial optimization*. John Wiley & Sons, New York, New York, 1988.

[108] A. Neumaier, O. Shcherbina, W. Huyer, and T. Vinko. A comparison of complete global optimization solvers. *Mathematical Programming, Series B*, 103:335–356, 2005.

[109] I.H. Osman. Heuristics for the generalized assignment problem: Simulated annealing and tabu search heuristics. *OR Spektrum*, 17:211–225, 1995.

[110] J. Pang. A new and efficient algorithm for a class of portfolio optimization problems. *Operations Research*, 28(3):754–767, 1980.

[111] P.M. Pardalos and N. Kovoor. An algorithm for a singly constrained class of quadratic programs subject to upper and lower bound constraints. *Mathematical Programming*, 46:321–328, 1990.

[112] P.M. Pardalos, Y. Ye, and C. Han. Algorithms for the solution of quadratic knapsack problems. *Linear Algebra and its Applications*, 152:69–91, 1991.

[113] N.C. Petruzzi and M. Dada. Pricing and the newsvendor problem: A review with extensions. *Operations Research*, 47(2):183–194, 1999.

[114] N. Piersma. *Combinatorial optimization and empirical processes*. Thesis Publishers, Amsterdam, The Netherlands, 1993.

[115] N. Piersma and H.E. Romeijn. Parallel machine scheduling: A probabilistic analysis. *Naval Research Logistics*, 43(6):897–916, 1996.

[116] M.C. Pullan. An algorithm for a class of continuous linear programs. *SIAM Journal on Control and Optimization*, 31(6):1558–1577, 1993.

[117] M.C. Pullan. A duality theory for separated continuous linear programs. *SIAM Journal on Control and Optimization*, 34(3):931–965, 1996.

[118] C. Rainwater, J. Geunes, and H.E. Romeijn. The generalized assignment problem with flexible jobs. Technical report, Department of Industrial and Systems Engineering, University of Florida, Gainesville, Florida, 2007.

[119] W.T. Rhee and M. Talagrand. A concentration inequality for the k-median problem. *Mathematics of Operations Research*, 14:189–202, 1989.

[120] A.G. Robinson, N. Jiang, and C.S. Lerme. On the continuous quadratic knapsack problem. *Mathematical Programming*, 55:99–108, 1992.

[121] G. Rodolakis, S. Siachalou, and L. Georgiadis. Replicated server placement with QoS constraints. In *Quality of Service in Multiservice IP Networks: Third International Workshop, QoS-IP 2005, LNCS 3375*, pages 207–220, 2005.

[122] H.E. Romeijn, J. Geunes, and K. Taaffe. On a nonseparable convex maximization problem with continuous knapsack constraints. *Operations Research Letters*, 35(2):172–180, 2007.

[123] H.E. Romeijn and N. Piersma. A probabilistic feasibility and value analysis of the generalized assignment problem. *Journal of Combinatorial Optimization*, 3:325–355, 2000.

[124] H.E. Romeijn and D. Romero Morales. A class of greedy algorithms for the generalized assignment problem. *Discrete Applied Mathematics*, 103:209–235, 2000.

[125] H.E. Romeijn and D. Romero Morales. A probabilistic analysis of the multi-period single-sourcing problem. *Discrete Applied Mathematics*, 112:301–328, 2001.

[126] H.E. Romeijn and D. Romero Morales. An asymptotically optimal greedy heuristic for the multi-period single-sourcing problem: The cyclic case. *Naval Research Logistics*, 50(5):412–437, 2003.

[127] H.E. Romeijn and D. Romero Morales. Asymptotic analysis of a greedy heuristic for the multi-period single-sourcing problem: The acyclic case. *Journal of Heuristics*, 10:5–35, 2004.

[128] H.E. Romeijn, D. Sharma, and R.L. Smith. Extreme point characterizations for infinite network flow problems. *Networks*, 48(4):209–222, 2006.

[129] H.E. Romeijn, J. Shu, and C.P. Teo. Designing two-echelon supply networks. *European Journal of Operational Research*, 178(2):449–462, 2007.

[130] H.E. Romeijn and R.L. Smith. Shadow prices in infinite dimensional linear programming. *Mathematics of Operations Research*, 23(1):239–256, 1998.

[131] H.E. Romeijn, R.L. Smith, and J.C. Bean. Duality in infinite dimensional linear programming. *Mathematical Programming*, 53:79–97, 1992.

[132] G.T. Ross and R.M. Soland. A branch and bound algorithm for the generalized assignment problem. *Mathematical Programming*, 9:91–103, 1975.

[133] G.T. Ross and R.M. Soland. Modeling facility location problems as generalized assignment problems. *Management Science*, 24(3):345–357, 1977.

[134] R. Roundy. Efficient, effective lot-sizing for multi-product, multi-stage production systems. *Operations Research*, 41:371–386, 1993.

[135] S. Ryan and J. Bean. Degeneracy in infinite horizon optimization. *Mathematical Programming*, 43:305–316, 1989.

[136] M.P.W. Savelsbergh. A branch–and–price algorithm for the generalized assignment problem. *Operations Research*, 45(6):831–841, 1997.

[137] I.E. Schochetman and R.L. Smith. Infinite horizon optimization. *Mathematics of Operations Research*, 14:559–574, 1989.

[138] I.E. Schochetman and R.L. Smith. Convergence of best approximations from unbounded sets. *Journal of Mathematical Analysis and Applications*, 166:112–128, 1992.

[139] I.E. Schochetman and R.L. Smith. Finite dimensional approximation in infinite dimensional mathematical programming. *Mathematical Programming*, 54:307–333, 1992.

[140] T.C. Sharkey, H.E. Romeijn, and J. Geunes. A class of nonlinear nonseparable continuous knapsack and multiple-choice knapsack problems. Technical report, Department of Industrial and Systems Engineering, University of Florida, Gainesville, Florida, 2007.

[141] Z.J. Shen, C. Coullard, and M.S. Daskin. A joint location-inventory model. *Transportation Science*, 37(1):40–55, 2003.

[142] D.B. Shmoys, C. Swamy, and R. Levi. Facility location with service installation costs. In *Proceedings of the 15th ACM Symposium on Discrete Algorithms*, pages 1088–1097, 2004.

[143] D.B. Shmoys, E. Tardos, and K. Aardal. Approximation algorithms for facility location problems. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 265–274, 1997.

[144] J. Shu, C.P. Teo, and Z.J. Shen. Stochastic transportation-inventory network design problem. *Operations Research*, 53:48–60, 2005.

[145] P. Sinha and A.A. Zoltners. The multiple-choice knapsack problem. *Operations Research*, 27(3):503–515, 1979.

[146] S. Smale. On the average number of steps of the simplex method of linear programming. *Mathematical Programming*, 27:241–262, 1983.

[147] K. Sourirajan, L. Ozsen, and R. Uzsoy. A single-product network design model with lead time and safety stock considerations. *IIE Transactions*, 39:411–424, 2007.

[148] M. Sviridenko. A 1.582-approximation algorithm for the metric uncapacitated facility location problem. In *Proceedings of the 9th Conference on Integer Programming and Combinatorial Optimization*, 2002.

[149] K. Taaffe, J. Geunes, and H.E. Romeijn. Target market selection with demand uncertainty: the selective newsvendor problem. *European Journal of Operational Research*, 189(3):987–1003, 2008.

[150] M. Talagrand. Sharper bounds for Gaussian and empirical processes. *The Annals of Probability*, 22:28–76, 1994.

[151] C.-P. Teo and J. Shu. Warehouse-retailer network design problem. *Operations Research*, 52(3):396–408, 2004.

[152] J. Thomas. Price-production decisions with deterministic demand. *Management Science*, 16(11):747–750, 1970.

[153] J. Thomas. Price-production decisions with random demand. *Operations Research*, 22(3):513–518, 1974.

[154] A.F. Veinott. Minimum concave cost solutions of Leontief substituion models of multi-facility inventory systems. *Operations Research*, 17:262–291, 1969.

[155] A. Wagelmans, S. van Hoesel, and A. Kolen. Economic lot sizing: An $O(n \log n)$ algorithm that runs in linear time in the Wagner-Whitin case. *Operations Research*, 40-S1:S145–S156, 1992.

[156] H.M. Wagner. A postscript to dynamic problems of the theory of the firm. *Naval Research Logistics Quarterly*, 7:7–12, 1960.

[157] H.M. Wagner and T.M. Whitin. Dynamic version of the economic lot size model. *Management Science*, 5:89–96, 1958.

[158] G. Weiss. A simplex based algorithm to solve separated continuous linear programs. Technical report, Department of Statistics, University of Haifa, Haifa, Israel, 2004.

[159] W.I. Zangwill. A deterministic multi-product, multi-facility production and inventory model. *Operations Research*, 17:486–507, 1966.

[160] W.I. Zangwill. The convex simplex method. *Management Science*, 14:221–238, 1967.

[161] E. Zemel. An $O(n)$ algorithm for linear multiple choice knapsack problems and related problems. *Information Processing Letters*, 18:123–128, 1984.

BIOGRAPHICAL SKETCH

Thomas C. Sharkey was born in Livingston, NJ, on April 19, 1982. He lived in New Jersey until graduating from Warren Hills Regional High School in June 2000. He then attended Johns Hopkins University in Baltimore, MD, where he received B.S. and M.S.E degrees in Mathematical Sciences in May 2004. He had the great fortune of marrying his wife, Melissa, on August 7, 2004 and beginning his doctoral studies in the Industrial and Systems Engineering department of the University of Florida later that month. He will receive his Doctor of Philosophy in industrial and systems engineering in August 2008 and then join the Department of Decision Sciences and Engineering Systems at Rensselaer Polytechnic Institute as an assistant professor.