MIXED INTEGER PROGRAMMING APPROACHES TO LOT-SIZING AND ASSET REPLACEMENT PROBLEMS

By

İ. ESRA BÜYÜKTAHTAKIN

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

2009

To my parents and my brothers

ACKNOWLEDGMENTS

I would like to express my sincere appreciation to my advisors Dr. Joseph C. Hartman and Dr. J. Cole Smith for their assistance in developing the ideas in this dissertation, their constant support and enormous patience throughout my doctoral studies. It has been a privilege to work with them. I am grateful to Dr. Joseph C. Hartman for the financial support needed to complete my studies, and for giving me the freedom to work on the problems that I am interested in. I owe a lot to his encouragement, understanding and guidance. I am indebted to Dr. J. Cole Smith for his attention to detail, dedication to perfection, insightful comments and invaluable counseling, which have significantly contributed to my development as a researcher. My Florida adventure would have not been so great without him.

I am also grateful to Dr. Joseph Geunes, Dr. Arunava Banerjee, and Dr. Fazil T. Najafi for their willingness to be on my dissertation committee as well as their valuable suggestions. I also would like to take this opportunity to thank to Dr. Fazil T. Najafi for his constant moral support during my graduate studies.

I also would like to recognize the professors at UF and Lehigh University from whom I have learned a lot during my graduate study. Special thanks to Dr. Ted Ralphs and Dr. Jeff Linderoth, who served as references during my job search. I would like to express my appreciation to Dr. Jean-Philippe Richard, whose lectures inspired me to develop some of the ideas in this thesis, and to Dr. Alper Atamtürk for his precious feedbacks regarding my research. I am also thankful to Dr. Tuba Yavuz-Kahveci and Dr. Tamer Kahveci for their constant support and mentoring.

I would like to thank to everyone who helped me to realize this dissertation. In particular, I would like to thank to my office mates Semra Ağralı and Caner Taşkın not only for their friendship and support but also for making Gainesville a more enjoyable place for me. I am indebted to Caner for insisting me to use a software package, which helped me to save substantial amount of time while coding my algorithms. I also would

like to thank to Chase Rainwater, Suat Boğ, Dinçer Konur, İbrahim Karakayalı, Joon-Hui Yoon and Shuang Chen for being great office mates to me. Thanks to Gonca Yıldırım for her friendship and enthusiastic technical support. I would like to thank Gudbjort Gylfadottir for all fun activities she organized for me and my fellows during graduate years. I am also thankful to ISE Department staff, in particular Marilyn Marlow and Mike Seufert for their help and technical support during my graduate study.

Thanks to my dear friend Hayriye Çiçekçi for her sincere friendship and endless support from overseas. I also would like to thank to my friends from Lehigh University, particularly to Mustafa Rasim Kılınç and Berrin Aytaç for their enormous help and support during my graduate study at Lehigh. Thanks to Özlem Demir, who has been a good friend and a good listener to me whenever I need, and Ayşegül Özkan for being such a wonderful roommate in my last year of graduate study.

Finally, I would like to thank to my parents Ayşe and Adem Büyüktahtakın, and my brothers İbrahim and Ömer for their unconditional love, support and encouragement. They are the source of my happiness, motivation and success.

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

MIXED INTEGER PROGRAMMING APPROACHES TO LOT-SIZING AND ASSET REPLACEMENT PROBLEMS

By

İ. Esra Büyüktahtakın

August 2009

Chair: Joseph C. Hartman
Cochair: J. Cole Smith
Major: Industrial and Systems Engineering

In this dissertation, we develop mixed integer programming approaches for solving capacitated lot-sizing and parallel asset replacement problems. For capacitated lot-sizing, we analyze the use of dynamic programming in mixed integer programming frameworks. Specifically, this research aims to make contributions to the polyhedral characterization of the capacitated lot-sizing problem by defining a new set of valid inequalities derived from the end-of stage solutions of a dynamic programming algorithm. The end-of-stage solutions of the dynamic program provide valid bounds on the partial objective function values of the problem. We then define the stage value function according to the state values for a given level of inventory in a given stage and approximate it by its convex envelope. These inequalities can then be lifted by investigating potential state information at future stages. We test several possible implementations of these inequalities on randomly generated instances and demonstrate that our approach is more efficient than other integer programming based algorithms.

We also consider a generalization of the capacitated lot-sizing problem called the multi-item capacitated lot-sizing problem (MCLSP). We study a mixed integer programming model for solving the MCLSP, which incorporates shared capacity on the production of items for each period throughout a planning horizon. We derive valid bounds on the partial objective function of the MCLSP formulation by solving the first

t periods of the problem over a subset of all items, using dynamic programming and integer programming techniques. We then develop algorithms for strengthening these valid inequalities by employing lifting and back-lifting procedures. These inequalities can be utilized in a cutting-plane strategy, in which we perturb the partial objective function coefficients to identify violated inequalities for the MCLSP polytope. We test the effectiveness of the proposed valid inequalities on randomly generated instances, and demonstrate that they are promising for solving MCLSP instances.

Our study of the parallel replacement problem is motivated by similar characteristics between the parallel replacement problem and lot-sizing problem. The parallel replacement problem under economies of scale (PRES) determines minimum cost replacement schedules for each individual asset in a group of assets that operate in parallel and are economically interdependent as a result of economies of scale. Economies of scale are due to the presence of the fixed charge in any period in which an asset is purchased. Both the parallel replacement and the lot-sizing problems have periodic demands that must be satisfied throughout the planning horizon. In the lot-sizing problem, production or purchases are made by trading off a fixed charge (set-up cost) against inventory holding and production/purchase costs. In the parallel replacement problem under economies of scale, additional fixed charges are incurred if assets are not replaced simultaneously. We prove that PRES is NP-hard. We then derive cutting plane approaches for the integer programming formulation of PRES. These cutting planes are motivated by the optimal replacement strategies implied by the no-splitting rule in the literature, which states that an optimal solution exists such that assets of the same age in the same time period are kept or replaced as a group. As a result of the no-splitting rule and constant demand, a purchase is enforced by a salvage in any optimal solution. We model PRES such that flow conservation constraints require a purchase whenever an asset is salvaged. We then use this property to generate inequalities for strengthening the PRES formulation. In addition, our inequalities have some similar characteristics with the flow cover inequalities

derived for capacitated fixed charge networks. We present a set of experiments to illustrate the computational efficiency of the inequalities with respect to solving the mixed integer programs in a cut-and-branch framework.

We also study the integer programming formulation of the PRES under technological change and deterioration. We provide optimal solution characteristics and insights about the economics of the problem. We propose cutting planes for strengthening the problem formulation and effective solution algorithms based on these cutting planes for the PRES under technological change. Finally, we present some computational results to illustrate the effectiveness of the proposed methods.

CHAPTER 1
INTRODUCTION

1.1 Background and Motivation

Given the increasing interest in operations, service and logistics costs, decisions regarding strategic planning of scarce resources and the efficient operations scheduling have become more important to the success of industrial corporations. The problem of satisfying service requirements at the lowest possible cost is complicated in many cases due to the combinatorial nature of the possible decisions and requires sophisticated solution methodologies for decision support.

Production planning is one important area, which requires strategic planning of the acquisition and allocation of resources such as parts, raw materials, machines and labor, as well as planning of production activities to transform resources or raw materials into finished goods in order to meet customer demand. The goal of production planning is to make optimal decisions with the typical objective of minimizing costs such as purchase, set-up (fixed) and inventory holding, or maximizing profit. To achieve this goal, industrial enterprises need to use quantitive tools in order to increase productivity while reducing costs under capacity restrictions. Restrictions on production arise due to limitations on the machine and/or labor or other resource capacities and, in general, make the problem harder to solve.

Parallel replacement is another important decision problem, which requires effective utilization and timely replacement of capital assets. Replacement analysis aims to provide decision support optimizing the trade-off between keeping assets longer, at higher operating costs, versus replacing with newer assets at higher capital costs. Similar to production planning, solving parallel replacement problems requires analytical tools to make sequential decisions for the management of the capital assets. The objective is to minimize costs by determining whether to keep or replace an existing asset among a group of assets, the amount of future assets that are going to be purchased, the timing of the

purchases, and how long each asset is kept in the system. The decisions regarding optimal asset replacement are further complicated when technological change and deterioration are incorporated.

Mixed integer programming (MIP) is a natural framework to model production planning and parallel replacement problems because of the problem characteristics (such as fixed costs) with regards to purchase decisions and capacity constraints on production or purchases. The main drawback of this approach is that MIP models may be difficult to solve for large instances that are usually encountered in production planning and parallel replacement systems. In particular, branch-and-bound algorithms, in which linear programming (LP) relaxations are used to prune nodes in the search tree, do not, generally, perform well for problems with fixed charge network flow characteristics due to the weak bounds provided by the LP relaxation. To overcome this limitation, sophisticated techniques can be used to tighten the LP relaxation bounds by tightening the mathematical formulations. One way to achieve tight formulations is to add valid inequalities or cutting planes to the LP formulation. The strengthened formulations may substantially reduce the computational time needed to solve them. Furthermore, for harder instances, these techniques may make it possible to increase the size of models solvable to optimality, or close to optimality.

Besides mixed integer programming approaches, designing other efficient optimization algorithms may be extremely useful for solving these hard problems. For instance, dynamic programming (DP) is a sequential optimization approach which works well for a class of production planning and parallel replacement problems. This motivates us to use the information from DP formulations to enhance the related MIP formulations.

This dissertation focuses on the development of methods to solve a class of production planning and parallel replacement problems more efficiently. We address these problems by analyzing their mixed integer programming formulations. We also study the techniques through which we can utilize DP to obtain stronger MIP formulations of these problems.

We present a number of new techniques for these problems and analyze the effectiveness of the proposed methodologies on randomly generated test problems.

In the remainder of this chapter, we first introduce a specific production planning problem called the lot-sizing problem, which is a special case fixed charge network flow problem. We also discuss the earlier polyhedral contributions and optimization approaches to lot-sizing problems. We then present another special case fixed charge network flow problem called the parallel equipment replacement problem. We also give a brief discussion of the earlier studies on this problem. Finally, we discuss our contributions and give a brief outline of the dissertation.

1.2 The Capacitated Lot-Sizing Problem

We consider a specific class of production planning problems referred to lot-sizing models since production planning in various industries is concerned with the determination of the size of the lot, or the amount to produce in each period over a finite time horizon. The objective is to satisfy the demand at minimum cost, subject to capacity constraints, which limits the production. This problem is NP-hard (Florian et al. (1980)).

Specifically, consider a set of products or items, for which periodic demand is known for a given horizon. Demand in each period can be satisfied either through production or inventory remaining from previous periods. Demands are not backlogged. Production is restricted due to machine, labor or resource capacity, and all items to be produced compete for this limited capacity. In all the lot-sizing models that we consider in this dissertation, production is capacitated, which is a realistic consideration that arises in many industrial settings.

The relevant costs are set-up (fixed) costs, production or purchase costs, and inventory holding costs. The fixed set-up cost must be paid before any production can occur, and total production costs depend on the quantity produced. There is also a unit inventory cost for each unit of item that is held in stock or inventory per unit of time.

1.2.1 Mixed Integer Programming Approaches to Lot-Sizing Problems

Lot-sizing is a fundamental problem in optimization whose importance stems from its applications in production/inventory planning and supply chain management. In particular, the lot-sizing structure (Atamtürk and Küçükyavuz (2008); Atamtürk and Muñoz (2004); Pochet and Wolsey (1991); Wolsey (1958)) is at the core of production planning problems involving multiple products and levels over a finite discrete time horizon.

Lot-sizing has been studied extensively in the mathematical programming literature, including defining valid inequalities and facets for the associated polyhedron. Barany et al. (1984b) first defined the convex hull of the uncapacitated lot-sizing problem (ULSP) and then used the facets for the ULSP to reformulate the multi-item capacitated lot-sizing problem (Barany et al. (1984a)). Küçükyavuz and Pochet (2009) give the convex hull of solutions for the ULSP with backlogging. For the constant-capacity case, Leung et al. (1989) and Pochet and Wolsey (1993) derive facets for the single-item capacitated lot-sizing problem. However, a complete linear description of the convex hull of solutions for this problem is unknown. Pochet (1988), Miller et al. (2000, 2003b), Loparic et al. (2003) and Atamtürk and Muñoz (2004) present facet-defining inequalities for the capacitated lot-sizing polytope where capacity restricting the production is general.

Atamtürk and Küçükyavuz (2005) analyze polyhedral characteristics of the ULSP in which there exist capacities and fixed charge costs on inventory in each period. Also, for the case in which periodic fixed charge costs exist for inventory as well as production, Atamtürk and Küçükyavuz (2008) provide an $O(T^2)$ optimal algorithm. Balas and Saxena (2008) provide rank-1 split cuts for general MIP problems and test them on the benchmark instances of Atamtürk and Muñoz (2004) for the general capacitated lot-sizing problem.

The single-item capacitated lot-sizing formulation forms a substructure for the multi-item and multi-level versions of the lot-sizing problem. Therefore the results found

for the single item lot-sizing problem can be useful in developing solution methods for multi-item problems. There have been various studies regarding multi-item capacitated lot-sizing and its variants in the literature. Most of the exact procedures proposed for the multi-item capacitated lot-sizing are based on integer programming algorithms and decomposition techniques. Polyhedral studies on the multi-item capacitated lot-sizing problems are devoted to strengthening the problem formulation by deriving valid inequalities via the analysis of knapsack, single node flow and single period relaxations of the problem (Miller (1999); Miller et al. (2000, 2003b); Padberg et al. (1985); Pochet and Wolsey (2006)).

Constantino (1996) provides cutting planes based on submodular inequalities for the single item capacitated lot-sizing problem with start-up costs, and uses these cuts for the multi-item version of the problem. Constantino (2000) extends the (K, l, S, I) inequalities of Pochet and Wolsey (1993) for the multi-item lot-sizing problem and derives several classes of valid inequalities for the multi-item capacitated lot-sizing problem, including special cases with backlogging and start-up costs. Pochet and Wolsey (1991) and Belvaux and Wolsey (2000, 2001) provide formulations and valid inequalities for the multi-item and multi-stage lot-sizing problems. Miller et al. (2003a) give a tight formulation for the multi-item lot-sizing problem with constant demands and set-up times. A 0-1 mixed integer programming formulation of a practical case of multi-item lot-sizing and scheduling is given in Smith-Daniels and Smith-Daniels (1986). Jans and Degraeve (2004) propose a decomposition algorithm to solve a multi-item, multi-resource capacitated lot-sizing problem with backlogging. Lagrangean-based heuristics and decomposition algorithms for multi-item capacitated lot-sizing problem are studied in Brahimi et al. (2006), Chen and Thizy (1991), Diaby et al. (1992), Tempelmeier and Derstroff (1996) and Trigeiro et al. (1989).

1.2.2 Dynamic Programming Approaches to Lot-Sizing Problems

Variants of the single-item lot-sizing problem have also been studied in the dynamic programming literature. An $O(T^2)$ algorithm for the ULSP was developed by Wagner and Whitin (1958). Wagelmans et al. (1992), Aggarwal and Park (1993) and Federgruen and Tzur (1991) provide $O(T \log T)$ algorithms for the same problem. Love (1973) gives an $O(T^3)$ algorithm for the ULSP with piecewise concave costs and bounded inventory. For the constant capacity case, Florian and Klein (1971) provide an $O(T^4)$ algorithm while Van Hoesel and Wagelmans (1996) present an $O(T^3)$ algorithm. Florian et al. (1980) present a dynamic programming algorithm with complexity $O(D_T C_T)$ for the single-item capacitated lot-sizing problem, where $D_T = \sum_{t=1}^{T} d_t$ is the cumulative demand and $C_T = \sum_{t=1}^{T} c_t$ is the cumulative capacity over all periods. For the capacitated lot-sizing problem where there are no restrictions on the cost functions, Chen et al. (1994) develop a continuous dynamic programming algorithm that is exponential in complexity but demonstrated to be computationally efficient in practice. Atamtürk and Hochbaum (2001) provide polynomial-time algorithms for the constant-capacity lot-sizing problem with capacity acquisition and subcontracting. Van Hoesel and Wagelmans (2001) give fully polynomial approximation schemes for the single-item capacitated lot-sizing problem. For the same problem, Baker et al. (1978) provide a branch-and-bound algorithm while Chung et al. (1994) present a hybrid branch-and-bound and dynamic programming algorithm.

There are exact approaches that tie dynamic programming to integer programming. Eppen and Martin (1987) provide tighter MIP formulations for the single and multi-item lot-sizing problems using a variable redefinition approach. They first drop the capacity constraints from the traditional lot-sizing formulation and represent the subproblem with the dynamic programming network structure. This shortest path network can be written as an integer program (IP), with the arcs corresponding to binary variables and the nodes corresponding to flow balance constraints. They then relate the variables of the traditional model to the new set of variables through a linear transformation. By

using this transformation, they insert the complicating constraints in terms of the new variables into the new formulation. Although this new reformulation has more variables and constraints, it has a tighter LP relaxation lower bound leading to reduced solution times. Martin et al. (1990) formulate polynomially solvable optimization problems as shortest path problems by using dynamic programming. They then represent the dynamic program as an LP having a polynomial number of variables and constraints. The extreme points of this LP are represented by the solution vectors of the DP, and the dual of the LP provides the DP formulation. They also show that with an appropriate change of variables, the LP formulation obtained from the DP provides a polyhedral description of the model considered.

1.3 Parallel Equipment Replacement Problem

The parallel equipment replacement problem under economies of scale (PRES) is concerned with the schedule of replacements for a group of assets in each time period so that the fixed and variable costs associated with replacing existing assets, operating and maintenance costs associated with the utilization and inventory of assets less the revenue obtained from the salvage of assets, are minimized. Assets operate in parallel and are economically interdependent. Economic interdependence is a result of economies of scale including a fixed charge in any period in which an asset is purchased.

One of the most important motivations behind replacing assets is advances in technology. Assets in use may become obsolete, as technological improvements make it possible for newer assets to operate more efficiently. Therefore, as technology evolves, managers might tend to keep less inventory to take advantage of new technology and increase the performance of the system. Another motivation for replacement is the deterioration of the assets as they are used over time. Deterioration results in increased operating and maintenance costs and reduced capacity due to machine breakdowns necessitating the replacement of the aged asset with a newer one.

Applications of this problem arise in fleet sizing, manufacturing line replacement, and replacement of the medical equipment. Additionally, the solution methodologies for PRES can be extended to provide further insights into decisions regarding capital equipment and technology selection problems.

1.3.1 Serial Replacement Analysis

The serial equipment replacement problem determines an optimal replacement schedule for an individual asset over a number of consecutive periods. In the serial replacement problem, unlike parallel replacement problems, replacement decisions are made independently since no economic interdependence among assets is assumed.

A number of papers have examined the serial equipment replacement problem. The first dynamic programming formulation for this problem is provided by Bellman (1955). In this formulation the state space represents the age of the asset, and the decisions include whether to keep or replace an asset in each stage. Later, Wagner (1975) provides another dynamic programming formulation in which the states are represented by the time periods and the decisions are how many periods to keep the asset. The dynamic programming formulation of the problem under technological change and multiple assets is also studied in the literature (see, e.g. Bean et al. (1985), Bean et al. (1994), Oakford et al. (1984)). For other single asset replacement studies considering the impact of technological change on capital and operating costs, see Hopp and Nair (1991), Hritonenko and Yatsenko (2007) and Regnier et al. (2004), among others.

1.3.2 Parallel Replacement Analysis

A variety of parallel replacement problems have also been studied in the literature. The parallel replacement problem under economies of scale was first examined by Jones et al. (1991). They prove that in an optimal policy, groups of same aged assets in the same time period are either kept or replaced together, assuming constant demand and no capital budgeting constraints. Under mild assumptions, they also prove that it is never optimal to replace newer clusters before older clusters. These rules vastly reduce

the amount of computation required to identify an optimal replacement policy in their dynamic programming approach. Later, Tang and Tang (1993) prove a stronger result, which states that in any period, an optimal replacement policy either keeps or replaces all machines regardless of age. Childress and Durango-Cohen (2005) consider a stochastic version of the parallel replacement problem with increasing failure rates and analyze the structure of optimal policies under general classes of replacement cost functions.

Chen (1998) provides a 0-1 integer programming formulation of the problem and uses Benders' decomposition while McClurg and Chand (2002) provide a forward algorithm for solving the problem with discounted costs. Chand et al. (2000) integrate the parallel replacement problem with the capacity expansion problem and use enumerative and heuristic algorithms to solve the problem. Rajagopalan (1998) provides an integer programming formulation model with fluctuating demand and various technology choices, and gives a dual based solution procedure to solve this problem. Hartman and Lohmann (1997) present an integer programming model for solving the demand constrained finite horizon parallel replacement problem with homogeneous assets where purchases, leases and rebuilds are viable replacement options, and analyze real-sized problems from the railroad industry. Hartman (2000) provides a general integer programming model including demand and capital budgeting constraints and illustrates that the linear programming relaxation of the parallel replacement problem under economies of scale has integer extreme points if the binary variables are fixed.

1.4 Contributions and Overview

As described in Section 1.2.1, most of the integer programming based approaches proposed for lot-sizing focus on the lot-sizing polyhedron without consideration of the objective function. Our contribution in this study is providing an approach that uses the objective function to guide us in adding valid inequalities that tighten the feasible region in the part of the polyhedron where an optimal solution lies. We first describe new valid inequalities for the single-item capacitated lot-sizing problem that are derived

from end-of-stage optimal solutions of a dynamic programming algorithm. Motivated by this approach, we provide valid inequalities for the multi-item lot-sizing polytope by obtaining valid bounds on the partial objective functions. In the multi-item case we also use both dynamic programming and other exact approaches such as integer programming methods to solve relaxations of the problem that yield valid bounds on the partial objective function. We then study techniques by which we can improve these inequalities by using lifting techniques. Another important contribution of this research is to develop and implement effective solution algorithms that use these cutting planes to reduce the solution time of the capacitated lot-sizing problem.

Our research on the parallel replacement problems has the following four contributions. First, we prove that PRES is NP-hard, as this has not yet been shown in the literature. Second, we define new cutting planes for the problem to strengthen its formulation. The cutting planes exploit the optimal replacement strategies and the network structure of the problem. Computational results show that the incorporation of these cuts in the problem formulation significantly reduces the time required to solve parallel replacement problems. Third, we provide an integer programming model for the parallel replacement problem where we directly incorporate the effect of technological change and deterioration on the system capacity, and give some insights on optimal asset replacement decisions under technological advances and deterioration. Fourth, we analyze and provide optimal solution characteristics for PRES under technological change, and, using these properties, we extend the inequalities for PRES to the technological change case.

Although this research addresses solution algorithms for lot-sizing and parallel replacement problems, we believe that the results presented in this thesis may give insights on solving other problems. Since both of the studied problems have fixed charge network flow properties, the developed solution approaches have the potential to be used for other fixed charge network flow problems. It is also possible to generalize some of the results

for lot-sizing to more general mixed integer programming problems, which may also be a valuable future contribution.

The remainder of this thesis is outlined as follows.

In Chapter 2, we consider the single item capacitated lot-sizing problem (CLSP). We use iterative solutions of forward and backward dynamic programming formulations for the CLSP to generate valid inequalities for an equivalent integer programming formulation. The inequalities essentially capture convex and concave envelopes of intermediate-stage value functions, and can be lifted by examining potential state information at future stages. We test several possible implementations that employ these inequalities, and demonstrate that our approach is more efficient than alternative integer programming based algorithms. For certain data sets, our algorithm also outperforms a pure dynamic programming algorithm for the problem.

In Chapter 3, we study a mixed integer programming model for solving the multi-item capacitated lot-sizing problem (MCLSP), which assumes shared capacity on the production of items in each period throughout a planning horizon. We derive valid bounds on the partial objective function of the MCLSP formulation by solving the first t-period relaxations of the problem using dynamic programming and integer programming techniques. We then lift these valid inequalities by strengthening the cut coefficients via back-lifting techniques. Using these techniques, we develop effective solution algorithms, in which we change the orientation of the partial objective function inequalities such that they cut off the fractional optimal solutions. We test the effectiveness of the proposed valid inequalities on randomly generated MCLSP instances.

In Chapter 4, we study PRES. We show that PRES is NP-hard, and derive cutting planes for the integer programming formulation of the parallel replacement problem considering constant demand. The motivation behind the cuts is the "no-splitting rule" in the literature. Experimental results illustrate the effectiveness of the cuts with respect to solving the integer programs in a cut-and-branch framework.

In Chapter 5, we consider the parallel replacement problem under technological change and deterioration and provide its integer programming formulation. Our model incorporates technological change as a gain in capacity, while deterioration is considered in terms of loss in capacity, increased operating and maintenance costs (O&M) costs and decreased salvage values. We illustrate how technology and deterioration affect the optimal replacement policy and give some insights into the problem and optimal solution characteristics. We also extend the inequalities developed for PRES to the technological change case and demonstrate their effectiveness in the computational experiments. We conclude with Chapter 6, and discuss promising future directions.

CHAPTER 2
SINGLE-ITEM CAPACITATED LOT-SIZING PROBLEM

2.1 Introduction

The capacitated lot-sizing problem (CLSP) is defined as follows: determine production levels in each period $t = 1, \ldots, T$ over a finite horizon in order to meet periodic demands (without backlogging) under periodic capacity constraints. The objective is to minimize periodic set-up, production, and inventory costs. The problem, which is NP-hard, is central to production planning and inventory control problems and has been studied widely in the literature, especially through integer and dynamic programming algorithms.

For each period $t = 1, \ldots, T$, let p_t, s_t and h_t denote the per-unit production, set-up and inventory costs for period t, respectively. Also, define variable x_t as the amount produced in period t, y_t as a binary set-up variable for period t (which equals 1 if production occurs in period t and 0 otherwise), and i_t as the amount of inventory held at the end of period t. The classical integer programming formulation for the CLSP is given as:

$$\min \sum_{t=1}^{T} (p_t x_t + s_t y_t + h_t i_t) \tag{2–1}$$

subject to:

$$\begin{aligned} i_{t-1} + x_t - d_t &= i_t && t = 1, \ldots, T && (2–2)\\ x_t &\leq c_t y_t && t = 1, \ldots, T && (2–3)\\ i_t, x_t &\geq 0 && t = 1, \ldots, T && (2–4)\\ y_t &\in \{0, 1\} && t = 1, \ldots, T. && (2–5) \end{aligned}$$

The objective function (2–1) minimizes the costs associated with set-up, production and inventory decision variables. Constraints (2–2) enforce flow balance conditions that require inventory remaining at the end of time t to equal previously held inventory plus new

production minus demand. Constraints (2–3) restrict production capacity to not exceed c_t if $y_t = 1$, and prohibit production if $y_t = 0$, in each period t. Constraints (2–4) define nonnegativity restrictions on the i- and x-variables, and (2–5) define binary restrictions on the y-variables. Note that while the initial inventory i_0 is free to take any value, we assume that it equals zero in this dissertation. Also note that the i_t and x_t variables are integer if y_t is integer.

This study makes contributions to the polyhedral characterization of lot-sizing problems by defining a new set of valid inequalities for the CLSP that are derived from the end-of-stage solutions of a dynamic programming algorithm. We present several implementations that incorporate inequalities into the root node of the branch-and-bound tree. Computational tests indicate that our algorithm is more efficient than the solution of the traditional MIP formulation and the approach of Atamtürk and Muñoz (2004) on a randomly generated data set. As in much of the cited related literature, we advocate the use of our approach only when a straightforward dynamic programming approach is intractable due to the size of the input data, or when the lot-sizing constraints are a subset of a larger problem.

The rest of this chapter is outlined as follows. In Section 2.2, we review the dynamic programming formulation for the CLSP and then derive valid inequalities based on forward and backward approaches in Section 2.3. Section 2.4 discusses implementing the valid inequalities and follows with computational results that illustrate the efficiency of our approach.

2.2 Dynamic Programming Approach

The dynamic programming approach of Florian et al. (1980) defines the state of the system in period t as the cumulative level of production through time t. Our approach is quite similar, although we define the state of the system in period t as the inventory level at time t.

Define $F_t(i)$ as the minimum cost of making optimal production decisions through period t, ending with inventory i. Periodic production decisions must meet demand d_t, and abide by production capacity limits c_t, in each period. Note that the minimum inventory level at period $t = 1, \ldots, T$ in any feasible solution is given by:

$$L_t = \max \left\{ 0, \max_{\tau=t+1,\ldots,T} \sum_{j=t+1}^{\tau} (d_j - c_j) \right\}, \tag{2–6}$$

since inventory must always be nonnegative and sufficient to cover future demands if capacity in future periods is not sufficient to cover these demands.

Similarly, defining $C_t = \sum_{j=1}^{t} c_j$ and $D_t = \sum_{j=1}^{t} d_j$, $\forall t = 1, \ldots, T$, the maximum inventory level at period $t = 1, \ldots, T$ in any optimal solution is given by:

$$U_t = \min \left\{ C_t - D_t, \sum_{j=t+1}^{T} d_j \right\}, \tag{2–7}$$

where the first term of (2–7) gives the maximum inventory that could accumulate after t periods and the second term gives the cumulative demand in future periods. As this is a finite horizon problem, no inventory remains after the final period of production in an optimal solution.

Note that in period $t = 1, \ldots, T$, given inventory level $L_t \leq i_t \leq U_t$, production in any optimal solution at period t lies in the set $X_{t,i_t} = \{\max\{0, i_t + d_t - U_{t-1}\}, \ldots, \min\{c_t, i_t + d_t - L_t\}\}$. Setting $F_0(0) = 0$ (since we have assumed $i_0 = 0$), the forward dynamic programming recursion can now be written as:

$$F_t(i_t) = \min_{x_t \in X_{t,i_t}} \{p_t x_t + s_t y_t + h_t i_t + F_{t-1}(i_t + d_t - x_t)\}, \ \forall t = 1, \ldots, T, \ L_t \leq i_t \leq U_t, \tag{2–8}$$

where $y_t = 1$ if $x_t > 0$ and $y_t = 0$ otherwise. The optimal objective function value is defined as $F_T(0)$.

This dynamic programming formulation can be represented as an acyclic graph, as shown in Figure 2-1 (depicted in Section 2.3). The nodes represent feasible states in each period, while the arcs represent feasible decisions for each state. The goal is to find

a shortest path connecting node $(0,0)$, representing the initial state of the system (no inventory at time zero), to node $(T,0)$, representing the final period of the problem when no inventory is needed as the problem terminates. (The nodes in Figure 2-1 are merely labeled with the inventory at the end of the period.) Here, the arc lengths are defined by the costs associated with each decision arc.

The number of nodes generated for a given instance is $\sum_{t=1}^{T-1}(U_t - L_t) + 2$, and the maximum number of arcs is $\sum_{t=1}^{T-2}(U_t - L_t)(U_{t+1} - L_{t+1}) + (U_1 - L_1) + (U_{T-1} - L_{T-1})$. The number of arcs determines the complexity of solving this problem by dynamic programming, and since $U_t - L_t \leq D_t$ for each $t = 1, \ldots, T$, the complexity of the algorithm is given by $O(T + TD_T^2)$, as all L_t and D_t can be defined in $O(T)$ time. Since $TD_T^2 >> T$, the complexity of the algorithm is essentially summarized by $O(TD_T^2)$.

The backward recursion, which traverses the network in Figure 2-1 from node $(T,0)$ to $(0,0)$, is defined similarly as:

$$G_{t-1}(i_{t-1}) = \min_{x_t \in X_{t,i_t}} \{p_t x_t + s_t y_t + h_t(i_t + x_t - d_t) + G_t(i_t + x_t - d_t)\}, \quad \forall t = 1, \ldots, T,\ L_{t-1} \leq i_t \leq U_{t-1}, \tag{2–9}$$

where, $G_t(i_t)$ represents the minimum cost of making optimal production decisions through period t, ending with inventory i_t in a backward recursion, and as before, $y_t = 1$ if $x_t > 0$ and $y_t = 0$ otherwise. In this case, the initial condition is defined as $G_T(0) = 0$ and the optimal objective function is $G_0(0)$.

Note that the state space and feasible periodic decisions are identical for each dynamic programming approach. Thus, the complexity of the backward recursion is the same as the forward recursion. However, the benefit of using both approaches can be readily illustrated by examining the representative networks, which are also the same.

The number of nodes (states) in the "middle" of the network is generally greater than the number of nodes at each end of the network. This is because the network initiates and ends with a single node (no inventory). From each of these end nodes, the number

of feasible states grows in each stage according to U_t and L_t. While U_t grows from stage 0 in the forward recursion, as it is defined by cumulative demand, it is also limited in the final stages (initial stages for the backward recursion) as it is defined by remaining cumulative demand. Thus, both the forward and backward recursions are easier to solve in their early stages due to the limited number of feasible states when compared to later stages. Furthermore, as the recursions are traversing different parts of the network, they each provide different information to produce valid inequalities, as discussed in below.

2.3 Valid Inequalities

Since the dynamic programming algorithm embodied by (2–8) usually approaches its worst-case pseudopolynomial running time, that algorithm is rarely implemented in full when D_T is large. However, a truncated version of this algorithm yields information that translates to valid inequalities for the CLSP mixed-integer programming formulation.

We begin by noting that for any $t = 1, \ldots, T$, the following inequality is valid:

$$\sum_{j=1}^{t} (p_j x_j + s_j y_j + h_j i_j) \geq F_t(L_t), \tag{2–10}$$

since $F_t(i_1) < F_t(i_2)$ for $L_t \leq i_1 < i_2 \leq U_t$, and hence $F_t(L_t)$ represents the minimum cost of all feasible decisions through period t. Our goal is to strengthen these inequalities by considering all of the state information available at each stage.

As a first step towards this goal, we can write the following conditional inequality for any combination of $t = 1, \ldots, T$ and $i = L_t, \ldots, U_t$:

$$\sum_{j=1}^{t-1} (p_j x_j + s_j y_j + h_j i_j) + s_t y_t + p_t x_t \geq F_t(i) - h_t i \quad \text{if } i = i_t. \tag{2–11}$$

Constraints (2–11) generate a lower bound on the objective function through period t, with the exception of period t inventory costs, for each state in a given stage. Lemma 1 states a valid inequality for the CLSP based on (2–11).

Lemma 1. *The following inequality is valid for CLSP:*

$$\sum_{j=1}^{t-1}(p_j x_j + s_j y_j + h_j i_j) + s_t y_t + p_t x_t \geq F_t(L_t) - h_t L_t, \tag{2–12}$$

and is at least as tight as (2–10) *(i.e., any solution* $(\hat{x}, \hat{y}, \hat{i})$ *that satisfies* (2–12) *must also satisfy* (2–10)*).*

Proof. Since (2–11) must be valid for at least one value of i, it is sufficient to show that $F_t(L_t) - h_t L_t \leq F_t(i) - h_t i$ for each $i = L_t + 1, \ldots, U_t$. Suppose by contradiction that $F_t(L_t) - h_t L_t > F_t(i) - h_t i$ for some $L_t + 1 \leq i \leq U_t$. Modify the solution associated with $F_t(i)$ by producing $i - L_t$ fewer units of inventory at the most recent period(s) of production. Then a modified feasible solution would exist having L_t units of inventory. Letting $\hat{F}_t(L_t)$ represent the objective function of the modified solution, we have

$$\widehat{F}_t(L_t) + h_t(i - L_t) < F_t(i),$$

where the inequality is due to the fact that production and inventory costs before time t have been ignored. However, our initial assumption, combined with the above inequality, yields

$$\begin{aligned} & F_t(L_t) - h_t L_t > F_t(i) - h_t i > [\widehat{F}_t(L_t) + h_t(i - L_t)] - h_t i \\ \Rightarrow\ & F_t(L_t) > \widehat{F}_t(L_t), \end{aligned}$$

which contradicts the optimal value of $F_t(L_t)$ at stage t, state L_t. Thus, (2–12) must be valid.

Next, note that any feasible solution $(\hat{x}, \hat{y}, \hat{i})$ that satisfies (2–12) must also satisfy $i_t \geq L_t$. Since $h_t i_t \geq h_t L_t$ aggregated with (2–12) yields (2–10), $(\hat{x}, \hat{y}, \hat{i})$ also satisfies (2–10). (The reverse is not necessarily true.) Hence, (2–12) is at least as strong as (2–10). □

Alternatively, we can state a lower bound on the costs accrued through time t as a function of the inventory at stage t. While $F_t(i_t)$ is not necessarily a convex function in terms of i_t, we can derive inequalities that define the convex envelope of $F_t(i_t)$ in terms of i_t. Such inequalities would have the form:

$$\sum_{j=1}^{t} (p_j x_j + s_j y_j + h_j i_j) \geq m_{tq} i_t + b_{tq} \tag{2–13}$$

for parameters m_{tq} and b_{tq}, $q = 1, \ldots, Q_t$, where Q_t is the number of segments defining the convex envelope. (Such inequalities must be valid by the assumption that inequalities (2–13) form the convex envelope of $F_t(i_t)$.) Our procedure for deriving these inequalities is given in Algorithm 1. In this algorithm, at stage t, given computed values $F_t(L_t), \ldots, F_t(U_t)$, we ensure that $m_{t1} < \cdots < m_{tQ_t}$, and that (2–13) for t, q is binding for two values $i_{t,q-1}$ and i_{tq}, where $i_{t0} = L_t$ and $i_{tQ_t} = U_t$. This procedure has complexity $O((U_t - L_t)^2)$. An alternative way of computing inequalities of the form (2–13) is using the algorithm of Chan (1996) with a complexity of $O((U_t - L_t)log(U_t - L_t))$ for stage t.

Algorithm 1: Computation of Convex Envelope Inequalities.

Input: Period t, inventory range $L_t, \ldots, U_t$, and values $F_t(i_t)$, $\forall i_t = L_t, \ldots, U_t$
Set segment count $Q_t \leftarrow 0$ and left segment point left $\leftarrow L_t$
while left $< U_t$ **do**
  slope $\leftarrow \infty$
  for point $\leftarrow$ (left + 1) to U_t **do**
    if $(F_t(\text{point}) - F_t(\text{left}))/(\text{point} - \text{left}) \leq$ slope **then**
      slope $\leftarrow (F_t(\text{point}) - F_t(\text{left}))/(\text{point} - \text{left})$
      right $\leftarrow$ point
    end
  end
  $Q_t \leftarrow Q_t + 1$
  Record the segment joining inventory levels left and right
  left $\leftarrow$ right
end

We now explore the tightness of (2–13) and seek to strengthen it where possible. For convenience, we define $z_t = \sum_{j=1}^{t}(p_j x_j + s_j y_j + h_j i_j)$ to simplify the expression of the left-hand-side of (2–13). For each stage $t = 1, \ldots, T$, we consider the convex hull of the

set:

$$P^t = \{(\nu_t, y_{t+1}, \ldots, y_T) : \text{ there exists a feasible solution to the CLSP having values } (y_{t+1}, \ldots, y_T) \text{ and } \nu_t \geq z_t\}, \quad (2\text{–}14)$$

i.e., $conv(P^t)$ is the epigraph of the stage t value function, projected onto the dimensions $y_{t+1}, \ldots, y_T$.

Suppose that we begin by fixing $y_t = 1$ for any period t such that $L_t > 0$. For simplicity in the following discussion, however, assume that no such variables can be fixed to 1, except for y_1, which must equal 1 if $d_1 > 0$. (The following results can readily be modified for the case in which some y_t-variables can be fixed to 1 for $t > 1$.) After executing Algorithm 1 above, define $i_{t,q-1}$ and i_{tq} to be the lower and upper inventory levels, respectively, that define segment q of the value function, $\forall q = 1, \ldots, Q_t$.

Lemma 2. *For $t \in \{1, \ldots, T\}$, $dim(conv(P^t)) = T - t + 1$.*

Proof. We show that there exist $T - t + 2$ affinely independent points corresponding to $(z_t, y_{t+1}, \ldots, y_T)$ in $conv(P^t)$. Define e to be a row vector of $T - t$ ones, and e_i to be the ith identity row for a $(T - t) \times (T - t)$ matrix. Consider points $w^i = (F_t(U_t), (e - e_i))$ for each $i = 1, \ldots, T - t$, $w^{T-t+1} = (F_t(U_t), e)$, and $w^{T-t+2} = (F_t(U_t) + 1, e)$. (The first element of these points corresponds to ν_t, while the remaining elements correspond to $y_{t+1}, \ldots, y_T$.) Note that points w^i, $i = 1, \ldots, T - t$, belong to $conv(P^t)$ since ν_t has been set large enough to procure the maximum possible inventory after period t (given by $F_t(U_t)$), and we have assumed that given this maximum inventory, there does not exist a period $u \in \{t + 1, \ldots, T\}$ such that production must occur in period u. Since all of these points are feasible, permitting production in all periods must also be feasible, and thus $w^{T-t+1} \in conv(P^t)$. This in turn implies by the inequality $\nu_t \geq z_t$ that $w^{T-t+2} \in conv(P^t)$.

Define W to be the $(T - t + 1) \times (T - t + 1)$ matrix where row 1 of W is given by $w^{T-t+2} - w^{T-t+1}$, and where row i of W is given by $w^{T-t+2} - w^{i-1}$, for $i = 2, \ldots, T - t + 1$. W is a binary-valued matrix with 1's in its first column and on its diagonal, and 0's

elsewhere, and is therefore nonsingular. Hence, $w^i,\ i = 1, \ldots, T-t+2$, are affinely independent and $\dim(conv(P^t)) = T - t + 1$. □

Proposition 1. *For $t \in \{1, \ldots, T\}$, suppose that given an inventory level of i_{tq} for some $q \in \{1, \ldots, Q_t\}$, there does not exist a $u \in \{t+1, \ldots, T\}$ such that $y_u = 1$ in every feasible solution. Then (2–13) for stage t, segment q, defines a facet to $conv(P^t)$.*

Proof. Given that $\dim(conv(P^t)) = T - t + 1$, we must identify $T - t + 1$ affinely independent points in $conv(P^t)$ that are binding on (2–13). Defining e and e_i as in the proof of Lemma 2, consider points $w^i = (F_t(i_{tq}), (e - e_i))$ for each $i = 1, \ldots, T-t$, and $w^{T-t+1} = (F_t(i_{t,q-1}), e)$. The first $T - t$ points must belong to $conv(P^t)$ by the assumption that no variable y_u, for $u = t+1, \ldots, T$, must be fixed to 1 in any feasible solution, while the last point belongs to $conv(P^t)$ since opening up production in all future periods must be feasible given an inventory level of $i_{t,q-1} \geq L_t$. Furthermore, these points are binding on (2–13) by definition of $F_t(i_{tq})$ and $F_t(i_{t,q-1})$. Consider the $(T-t) \times (T-t+1)$ matrix W in which row i is given by $w^{T-t+1} - w^i$, for each $i = 1, \ldots, T-t$. Columns $2, \ldots, T-t+1$ form an identity matrix, which verifies that W has full row rank, and hence points $w^i,\ \forall i = 1, \ldots, T-t+1$, are affinely independent. □

However, suppose now that given some segment $q \in \{1, \ldots, Q_t\}$ for period t, we have that all feasible solutions to the CLSP must set $y_u = 1$, for some $t+1 \leq u \leq T$, given that the inventory level after period t is i_{tq}. In this case, our derivation of points w^i in the proof of Proposition 1 is invalid, and in fact, (2–13) does not necessarily induce a facet to $conv(P^t)$. Instead, we now investigate how to lift (2–13) into a facet-defining inequality to $conv(P^t)$ in this case.

At the smallest feasible inventory level for period t, define $\gamma_{t1} = L_t$, and define $S_{t1} \subseteq \{t+1, \ldots, T\}$ as the set of time periods u for which $y_u = 1$ in every feasible solution for which $i_t = i_{tq}$ in stage t. Increment this inventory level until there exists a $u \in S_{t1}$ such that $y_u = 0$ in some feasible solution. Let γ_{t2} be this inventory level, and define $S_{t2} \subset S_{t1}$

as the set of time periods u for which $y_u = 1$ in every feasible solution, given inventory γ_{t2}. Repeat this process until γ_{tR_t} is reached, for which $S_{tR_t} = \emptyset$. (We assume that such an inventory level exists, or else our assumption that for any $u \in \{1, \ldots, T\}$, $y_u = 0$ in some feasible solution to the given CLSP instance, is false.) Also, define $\gamma_{t,R_t+1} = \infty$.

Given a constraint (2–13) corresponding to stage t, segment q, define r such that $\gamma_{tr} \leq i_{tq} < \gamma_{t,r+1}$. If $S_{tr} = \emptyset$, then (2–13) already defines a facet to $conv(P^t)$. Else, choose any $u_1 \in S_{tr}$, and find the largest value of α_{t1} for which:

$$z_t \geq m_{tq} i_t + b_{tq} + \alpha_{t1}(1 - y_{u_1}) \tag{2–15}$$

is valid. Since we are only concerned about feasible solutions for which $y_{u_1} = 0$ in the derivation of α_{t1}, we wish to maximize α_{t1}, subject to:

$$\alpha_{t1} \leq z_t - m_{tq} i_t - b_{tq}, \quad \forall (z_t, i_t) : \text{there exists a feasible solution having partial cost } z_t, \text{ period } t \text{ inventory } i_t, \text{ and } y_{u_1} = 0. \tag{2–16}$$

Define r' as the smallest index for which $u_1 \notin S_{tr'}$. For the first such value of α_{t1}, this lifting can be done in pseudopolynomial time by enumerating i_t-values greater than or equal to $\gamma_{tr'}$, and z_t-values corresponding to those inventory levels as already computed within the dynamic programming phase.

However, as the number of previously lifted terms grows, this lifting process becomes more difficult. Suppose that we have lifted $y_{u_1}, \ldots, y_{u_w}$, obtaining coefficients $\alpha^\star_{t1}, \ldots, \alpha^\star_{tw}$, and are now attempting to lift $y_{u_{w+1}}$. That is, we seek the largest possible value of $\alpha_{u_{w+1}}$ such that

$$\alpha_{u_{w+1}} \leq z_t - m_{tq} i_t - b_{tq} - \sum_{k=1}^{w} \alpha^\star_{tk}(1 - y_{u_k}), \tag{2–17}$$

for any feasible $z_t, i_t, y_{u_1}, \ldots, y_{u_w}$, and where $y_{u_{w+1}} = 0$. At this point, maximizing $\alpha_{tu_{w+1}}$ cannot practically be done by enumerating every relevant i_t-value. This task becomes virtually isomorphic with solving the CLSP itself (as can be expected with the derivation of strong inequalities for NP-hard problems). Hence, in our computational study, we

investigate the effectiveness of lifting in a single dimension only, when necessary, by identifying the maximum right-hand-side value of (2–16).

Upper-bounding inequalities. Similar to (2–10) and (2–12), we can also generate inequalities considering the maximum state value in each period. Specifically,

$$\sum_{j=1}^{t} (p_j x_j + s_j y_j + h_j i_j) \leq F_t(U_t), \tag{2–18}$$

and the strengthened inequality:

$$\sum_{j=1}^{t-1} (p_j x_j + s_j y_j + h_j i_j) + p_t x_t + s_t y_t \leq F_t(U_t) - h_t U_t, \tag{2–19}$$

can both be generated without cutting off any optimal solutions, although they may cut off feasible non-optimal solutions. Hence, while we can use (2–19) in optimizing CLSP, these inequalities are not technically valid.

Proposition 2. *All optimal CLSP solutions satisfy* (2–19).

Proof. By contradiction, suppose that $F_t(i_t) - h_t i_t > F_t(U_t) - h_t U_t$ for some optimal solution $\tilde{x}$ having i_t units of inventory at stage t, where $\sum_{j=1}^{t}(p_j x_j + s_j y_j + h_j i_j) = F_t(i_t)$. Now, consider a solution $\overline{x}$ that yields a partial cost of $F_t(U_t)$ through stage t, and has U_t units of inventory at stage t. Create a feasible solution $\hat{x}$ as a modification of $\overline{x}$ in which $U_t - i_t$ fewer units are produced at the most recent periods to t in which production occurred. The objective value of $\hat{x}$ at stage t, state i_t, satisfies:

$$\widehat{F}_t(i_t) < F_t(U_t) - h_t(U_t - i_t), \tag{2–20}$$

where the inequality is due to saving costs from producing $(U_t - i_t)$ fewer units, as well as potential savings in inventory and set-up costs. But assuming $F_t(i_t) - h_t i_t > F_t(U_t) - h_t U_t$, then by noting (2–20), we also have:

$$F_t(i_t) - h_t i_t > F_t(U_t) - h_t U_t > \left[\widehat{F}_t(i_t) + h_t(U_t - i_t)\right] - h_t U_t$$
$$\Rightarrow F_t(i_t) > \widehat{F}_t(i_t).$$

This contradicts the assumption that $\tilde{x}$ would have been an optimal solution in state t, stage i_t, and completes the proof. □

As with the convex envelope defined by the lower bounds on the value function $F_t(i_t)$, we can define a concave envelope from the upper bounds on $F_t(i_t)$ as:

$$\sum_{j=1}^{t} (p_j x_j + s_j y_j + h_j i_j) \leq m'_{tq} i_t + b'_{tq} \tag{2–21}$$

for parameters m'_{tq} and b'_{tq}, $q = 1, \ldots, Q_t$, where Q_t is the number of segments defining the convex envelope. These inequalities can be derived similarly to those for the convex envelope as in Algorithm 1. Note that these upper-bounding inequalities will not cut off optimal linear programming relaxation solutions if implemented in isolation because optimality ensures that the partial objective function values on the left-hand-side of (2–21) (and of (2–18) and (2–19)) are minimized. However, given the presence of lower-bounding inequalities, the upper-bounding inequalities serve to distribute costs among stages. For instance, suppose that a lower-bounding inequality of the form (2–10) states that the partial objective function through stage $2 \leq t \leq T$ is at least ν_1, and that an upper-bounding inequality of the form (2–18) states that the objective through stage $1 \leq t' < t$ should be no more than ν_2. The inclusion of the upper-bounding inequality thus implies that the costs incurred between stages $t'+1$ through t should be at least $\nu_1 - \nu_2$.

We illustrate the derivation of the foregoing inequalities in the following example.

Example 1. Inequality generation for an instance with $T = 4$. To illustrate our techniques, consider an instance of CLSP with $T = 4$, $c_t = (5, 3, 4, 3)$, $d_t = (2, 3, 3, 3)$, $p_t = (1, 2, 1, 3)$, $s_t = (8, 7, 6, 7)$ and $h_t = (2, 2, 1, 1)$. This data yields $L_t = 0$ and $U_t = 3$ for $t = 1, 2, 3$. The problem can be visualized by the acyclic graph depicted in Figure 2-1, with each node representing a feasible state (inventory level) in each period. The F-values are provided as node (state) labels in the figure for both the forward and backward dynamic programming approaches.

The optimal solution is represented by the bold path in Figure 2-1. Production occurs in periods 1, 3 and 4 as $x_t = (5, 0, 4, 2)$ at a total cost of 43 ($F_4(0)$ for the forward recursion and $F_0(0)$ for the backward recursion).

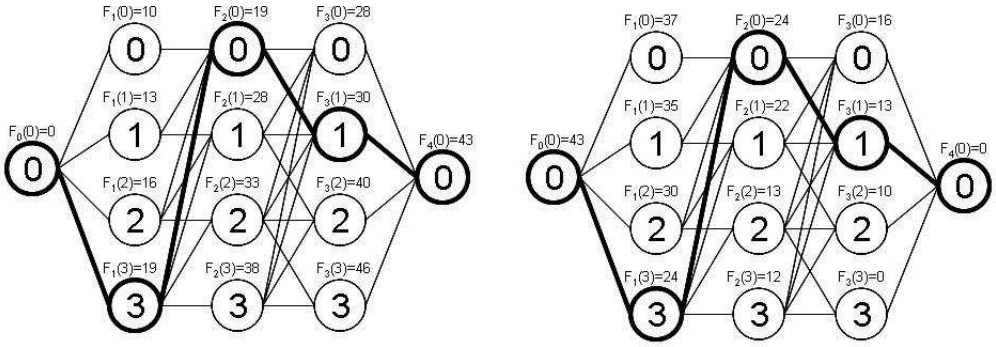

Figure 2-1. Graphs for forward (left) and backward (right) dynamic programming recursions.

The following discussion follows according to the forward dynamic programming approach. For $t = 1$, the inequality corresponding to (2–10) is:

$$x_1 + 8y_1 + 2i_1 \geq 10 \tag{2–22}$$

and reduces to:

$$x_1 + 8y_1 \geq 10 \tag{2–23}$$

according to (2–12). The single inequality that defines the entire convex envelope for $F_1(i_1)$, from (2–13), is:

$$x_1 + 8y_1 + 2i_1 \geq 3i_1 + 10 \tag{2–24}$$

In general, the period 1 constraint $z_1 \geq (p_1 + h_1)i_1 + (s_1 + p_1 d_1)$ will always define exactly the function $F_1(i_1)$ if $d_1 > 0$, since cost $(s_1 + p_1 d_1)$ must be paid in any solution in order to accommodate period 1 demand (recall that $i_0 = 0$), and each extra unit produced is placed into inventory at a cost $p_1 + h_1$. Note that (2–24) is a stronger inequality than (2–23), which is stronger than (3–13).

While the definition of $F_1(i_1)$ is convex for all values of i_1, Figure 2-2 illustrates that $F_2(i_2)$ and $F_3(i_3)$ are not convex over i_3. Thus, we define convex envelope inequalities, as given in the figure.

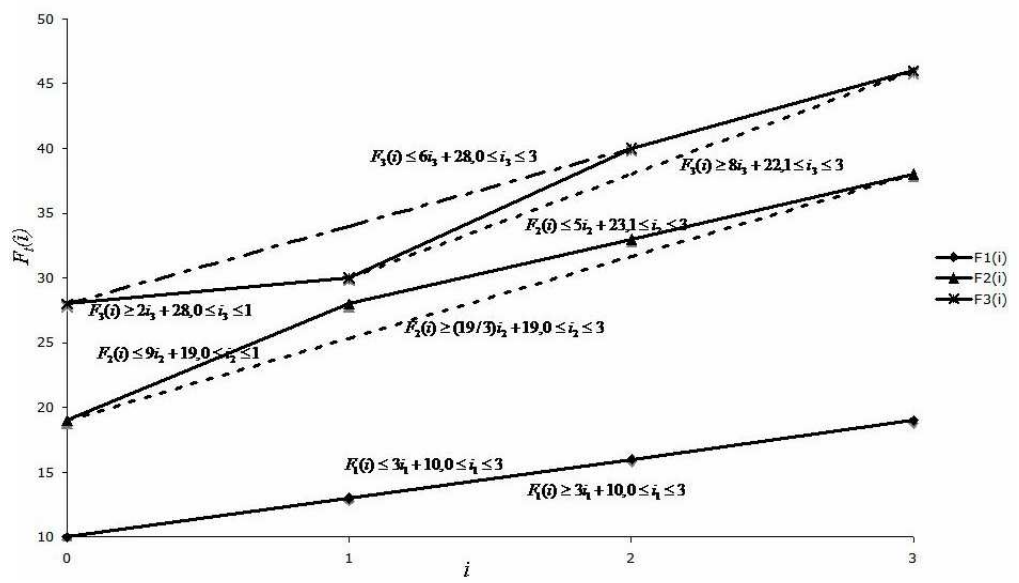

Figure 2-2. Graph representation of $F_t(i_t)$ values and associated convex envelope inequalities.

For $t = 2$, connecting inventory levels 0 and 3 defines:

$$\sum_{j=1}^{2} (p_j x_j + s_j y_j + h_j i_j) \geq (19/3) i_2 + 19. \tag{2–25}$$

The convex envelope is defined by two inequalities for $t = 3$. Connecting inventory levels 0 and 1 defines:

$$\sum_{j=1}^{3} (p_j x_j + s_j y_j + h_j i_j) \geq 2 i_3 + 28, \tag{2–26}$$

while connecting inventory levels 1 and 3 defines:

$$\sum_{j=1}^{3} (p_j x_j + s_j y_j + h_j i_j) \geq 8 i_3 + 22. \tag{2–27}$$

Proposition 1 indicates that (2–25) and (2–27) define facets to P^2 and P^3, respectively, since the upper inventory limit of both inequalities is 3, and thus no y-variables at future time periods may be fixed to 1. However, the upper inventory limit for (2–26) is 1; if only

1 unit is left in inventory after period 3, then we must set $y_4 = 1$. With this information, setting $u_1 = 4$, we can lift (2–26) by adding the term $\alpha_{t1}(1 - y_4)$ to the right-hand-side of (2–26), and then by computing $\alpha_{t1} = 46 - 2(3) - 28$ as prescribed by (2–16). The resulting inequality is given by

$$z_3 \geq 2i_3 + 28 + 12(1 - y_4). \tag{2–28}$$

Thus, the first segment of the convex envelope for $F_3(i_3)$ is essentially shifted up by 12 units if $y_4 = 0$.

The graphical representations of $F_t(i_t)$ in Figure 2-2 also illustrate the inequalities used to define the concave envelope of $F_t(i_t)$. For $t = 1$, as noted earlier, the convex and concave envelopes are the same since production must occur in the first period. For $t = 2$, inventory levels between 0 and 1 define:

$$\sum_{j=1}^{2} (p_j x_j + s_j y_j + h_j i_j) \leq 9i_2 + 19 \tag{2–29}$$

while inventory levels between 1 and 3 define:

$$\sum_{j=1}^{2} (p_j x_j + s_j y_j + h_j i_j) \leq 5i_2 + 23 \tag{2–30}$$

For $t = 3$, the single inequality:

$$\sum_{j=1}^{3} (p_j x_j + s_j y_j + h_j i_j) \leq 6i_3 + 28 \tag{2–31}$$

summarizes the concave envelope over all inventory levels. As with the convex envelope, these inequalities may be strengthened through a similar lifting procedure.

Finally, if desired, we can repeat all of the above procedures using the backward dynamic programming information. For example, from Figure 2-1 and $t = 2$, the recursion defines:

$$x_3 + 6y_3 + i_3 + 3x_4 + 7y_4 + i_4 \geq 12 \tag{2–32}$$

which can be modified and lifted as previously. The benefit of using both approaches may often be seen in large-scale implementations.

2.4 Computational Experiments

In this section we present experiments on the computational effectiveness of the inequalities derived in the previous section on randomly generated CLSP instances. All computations were performed on a Linux workstation with two 1.8Ghz processors and 1 GB memory, using the callable libraries of CPLEX[1] Version 11.0. A 1800 CPU-second time limit was imposed for all test instances.

2.4.1 Instance Generation

Problems are generated according to the data generation technique of Atamtürk and Muñoz (2004) who noted that (a) the tightness of the capacities with respect to demand and (b) the ratio between the set-up cost and the inventory holding cost play major roles in defining the difficulty of a CLSP instance. Specifically, instances are generated for capacity-to-demand ratios $c \in \{2, 3, 4, 5\}$, set-up-to-holding-cost ratios $f \in \{100, 200, 500, 1000\}$, and number of stages $T \in \{90, 120, 150\}$. Five random instances are generated for each combination of these parameters, for a total of 240 instances. The unit production costs p_t, demands d_t, capacities c_t and set-up costs s_t are randomly generated from integer uniform distributions with ranges as follows: $p_t \in [81, 119], d_t \in [1, 19], c_t \in \left[0.75c\bar{d}, 1.25c\bar{d}\right], s_t \in \left[0.90f\bar{h}, 1.10f\bar{h}\right]$ where $\bar{d}$ and $\bar{h}$ are averages for demand and holding costs. The holding cost h_t is fixed at 10 for each period.

2.4.2 Implementation and Experimental Design

Our implementation executes the forward dynamic programming algorithm for a limited number of stages to obtain all necessary information to generate our proposed inequalities. We append these inequalities to strengthen the CLSP formulation (2–1)–(2–5), and solve the resulting model using CPLEX with its default settings.

[1] CPLEX is a trademark of ILOG, Inc.

Additionally, although we can also generate inequalities based on a partial backward recursion, our preliminary computational results indicated that these inequalities are not computationally effective.

We compare the efficiency of solving the following models in our computational experiments.

- **base**: Formulation (2–1)–(2–5), without adding any user inequalities.
- **weakl**: Inequalities (2–12) with base.
- **weaklu**: Inequalities (2–12) and (2–19) with base.
- **envl**: Convex envelope inequalities (2–13) that violate the initial LP relaxation of (2–1)–(2–5), with base.
- **envlu**: Concave envelope inequalities (2–21) with envl.
- **liftenvl**: Lifted convex envelope inequalities (2–13) that violate the initial LP relaxation, with base.
- **weakl+env**: weakl + liftenvl, with base. Note that this essentially states a single lower-bounding constraint (2–12) for each stage considered, and provides tightening with the lifted convex envelope inequalities where they cut off the current LP relaxation solution.
- **bc**: The bottleneck formulation given by Atamtürk and Muñoz (2004), without adding any user cuts.
- **bclift**: Lifted bottleneck inequalities given by Atamtürk and Muñoz (2004), with bc. We add these inequalities based on the heuristic separation approach of Atamtürk and Muñoz (2004) with one round to the initial LP relaxation of the bottleneck model. Then we solve the strengthened model with CPLEX.

Atamtürk and Muñoz (2004) show that bclift is equivalent to the (l, S) inequality of Barany et al. (1984b) for the uncapacitated case, and at least as strong as the flow cover inequality of Padberg et al. (1984) and the surrogate flow inequality of Pochet (1988). Since bclift was also shown to be the most effective cutting plane implementation, we restrict the computational comparison of our procedures to bc and bclift. Note that there are several further implementations that combine the use of our inequalities, but

for the sake of brevity, we present only those that we found to be most efficient in our computational experiments.

2.4.3 Summary of Experimental Results

In Tables 2-1–2-6 we report the following data by column:

- **exp**: Solution approach.
- **stage**: The number of forward-recursion stages up to which DP inequalities are generated; if inequalities are added in combination, we report the number of stages as a pair of numbers:
  - **weaklu**: Stages used in generating ((2–12), (2–19)).
  - **envlu**: Stages used in generating (convex envelope inequalities (2–13) that violate the initial LP relaxation, (2–21)).
  - **weakl+env**: Stages used in generating ((2–12), lifted inequalities (2–13) that violate the initial LP relaxation).
- **initgap**: The initial optimality gap given an incumbent objective function value (initub) found by CPLEX, and lower bound after user inequalities are added, i.e., gap = 100(bestub−initlb)/bestub.
- **gapimp**: The improvement in the initial gap over base after adding the DP based inequalities, that is, gapimp = 100(1−gap/basegap), where basegap is the optimality gap in the base model.
- **DPineq**: The number of DP based inequalities added to the original formulation.
- **CPXineq**: The number of CPLEX cuts added in the search tree
- **nodes**: The number of nodes that CPLEX explores in the branch-and-bound tree.
- **ineqtime**: The total CPU seconds required to generate the DP based inequalities.
- **time**: The total CPU seconds required to solve the problem including inequality generation time.(Note that for bc and bclift, the foregoing columns refer to the impact of bottleneck inequalities rather than DP based inequalities.)

Tables 2-1 and 2-2 present results for $T = 90$, where each table entry corresponds to the average performance of an algorithm over 20 instances (five each for $f = 100$, 200, 500 and 1000). We observe that the presented DP based strategies all improve upon the direct solution of the CLSP by CPLEX, either with the base or bottleneck formulation. The

liftenv strategy, which lifts the convex envelope inequalities (2–13) that violate the initial LP relaxation, appears to provide the most benefit, with a factor of 2 improvement in CPU run time over base, averaging over all instances. The envl strategy, which is similar to liftenv but does not perform the lifting operation, also performs well in most instances. The bclift implementation improves upon bc as expected for these instances, but the liftenv strategy performs better than bclift implementation. Hence, we omit the bc and bclift computational results from this point forward.

Tables 2-3 and 2-4 present the results when repeating the experiments for $T = 120$. As expected, these instances are significantly more challenging than the $T = 90$ instances. As with the $T = 90$ case, the envl and liftenvl strategies performed well, with nearly a factor of 6 improvement over base when averaging over all instances. The weakl+env(60,60) strategy provided the best computational results (nearly 8 times improvement over base), although it required a significant portion of the DP to be solved in generating the inequalities.

Tables 2-5 and 2-6 provide computational results when $T = 150$. For these experiments, some algorithms failed to solve instances within the allotted 1800 CPU seconds. The number of such instances is denoted with a superscript in the final column. If an instance fails to solve within the time limit, a time of 1800 seconds is recorded as the computational time for the instance, thus underestimating the true average time required to optimize these instances.

Of the 80 instances for $T = 150$, 34 could not be solved to optimality by base within the time limit, while 77 could be solved by liftenv(100) and 71 could be solved by weakl+env(50). The liftenv(75) strategy reduced the average base solution time by a factor of 4, while liftenv(50) improved solution time by nearly a factor of 2 when averaged over all of the instances.

Table 2-1. Summary of experiments for $T = 90$, $c = 2, 3$.

		c=2						
exp	**stage**	**initgap**	**gapimp**	**Dpineq**	**CPXineq**	**nodes**	**ineqtime**	**time**
base	-	2.98	-	0	158	59098	0.00	29.44
weakl	30	2.31	21.58	30	143	38030	0.03	24.53
weakl	45	1.96	32.44	45	124	25774	0.08	19.24
weakl	60	1.51	47.19	60	104	31969	0.14	31.01
weaklu	(30,30)	2.33	21.09	60	138	56141	0.03	33.59
weaklu	(45,45)	1.98	31.82	90	120	32210	0.08	25.04
weaklu	(60,60)	1.54	46.47	120	95	21527	0.13	25.93
envl	30	1.90	36.75	132	125	13034	0.03	12.12
envl	45	1.48	49.88	227	122	8639	0.08	23.80
envl	60	1.03	65.67	329	131	7114	0.13	38.63
envlu	(30,30)	1.90	36.75	133	122	12246	0.03	11.65
envlu	(45,45)	1.48	49.88	228	123	8460	0.08	24.04
envlu	(60,60)	1.03	65.65	331	116	8225	0.13	40.69
liftenvl	30	1.89	36.96	135	110	15448	0.03	16.72
liftenvl	45	1.46	50.52	230	108	4046	0.08	12.37
liftenvl	60	1.00	66.39	333	115	6318	0.13	36.75
weakl+env	(30,30)	1.89	36.96	165	114	14359	0.03	14.85
weakl+env	(45,45)	1.46	50.52	275	100	7655	0.08	19.60
weakl+env	(60,60)	1.00	66.39	393	109	2725	0.13	20.83
bc	-	2.98	-	0	332	158736	0.00	127.82
bclift	-	1.53	43.63	149*	170	19220	1.80	53.38
		c=3						
base	-	5.00	-	0	139	37210	0.00	18.15
weakl	30	3.83	23.41	30	162	19761	0.10	13.97
weakl	45	3.15	36.33	45	140	24674	0.22	17.98
weakl	60	2.49	48.98	60	128	17804	0.27	16.31
weaklu	(30,30)	3.86	22.80	60	149	21774	0.11	14.65
weaklu	(45,45)	3.19	35.60	90	134	25455	0.22	20.32
weaklu	(60,60)	2.53	48.19	120	119	9777	0.27	13.96
envl	30	3.16	37.47	142	117	21144	0.11	20.68
envl	45	2.46	51.11	243	138	2943	0.21	10.91
envl	60	1.77	64.43	354	131	1880	0.27	18.52
envlu	(30,30)	3.16	37.47	142	117	21144	0.11	20.81
envlu	(45,45)	2.46	51.11	243	134	2951	0.21	11.04
envlu	(60,60)	1.77	64.41	354	128	1954	0.27	18.79
liftenvl	30	3.13	37.90	144	140	7110	0.10	10.12
liftenvl	45	2.44	51.58	245	131	2735	0.22	11.76
liftenvl	60	1.76	64.79	356	133	1753	0.27	19.77
weakl+env	(30,30)	3.13	37.90	174	125	17235	0.10	17.45
weakl+env	(45,45)	2.44	51.58	290	121	3576	0.21	13.03
weakl+env	(60,60)	1.76	64.79	416	124	1919	0.27	19.31
bc	-	4.99	-	0	301	41633	0.00	80.84
bclift	-	2.25	51.65	173*	257	6967	1.62	27.54

*Number of lifted bottleneck cutting planes generated

Table 2-2. Summary of experiments for $T = 90$, $c = 4, 5$.

		c=4						
exp	**stage**	**initgap**	**gapimp**	**Dpineq**	**CPXineq**	**nodes**	**ineqtime**	**time**
base	-	7.21	-	0	116	21128	0.00	9.70
weakl	30	5.46	24.38	30	133	8554	0.20	6.47
weakl	45	4.51	37.16	45	119	6967	0.30	6.62
weakl	60	3.47	51.34	60	113	3838	0.37	5.97
weaklu	(30,30)	5.50	23.73	60	127	9442	0.19	7.22
weaklu	(45,45)	4.57	36.40	90	119	7752	0.30	7.56
weaklu	(60,60)	3.54	50.49	120	100	4715	0.36	8.58
envl	30	4.57	36.80	142	124	3544	0.19	5.82
envl	45	3.55	50.84	241	106	1694	0.30	7.44
envl	60	2.40	66.94	348	127	1205	0.36	13.89
envlu	(30,30)	4.57	36.80	142	125	3731	0.19	6.56
envlu	(45,45)	3.55	50.84	241	106	1938	0.30	7.94
envlu	(60,60)	2.40	66.94	348	125	1190	0.36	13.80
liftenvl	30	4.56	36.91	143	100	5143	0.19	6.66
liftenvl	45	3.54	50.95	242	102	1769	0.30	7.78
liftenvl	60	2.39	67.04	349	114	1086	0.36	13.93
weakl+env	(30,30)	4.56	36.91	173	119	4147	0.19	6.43
weakl+env	(45,45)	3.54	50.95	287	100	1960	0.30	8.03
weakl+env	(60,60)	2.39	67.04	409	118	1434	0.36	14.14
bc	-	7.21	-	0	176	31744	0.00	24.81
bclift	-	2.51	63.56	216⋆	167	3071	1.81	11.29
		c=5						
base	-	9.68	-	0	123	38780	0.00	16.30
weakl	30	7.43	23.10	30	127	14193	0.24	8.64
weakl	45	6.06	37.17	45	118	8628	0.36	7.26
weakl	60	4.60	52.08	60	94	5701	0.43	6.61
weaklu	(30,30)	7.50	22.43	60	99	15601	0.25	9.55
weaklu	(45,45)	6.14	36.35	90	102	9785	0.36	8.74
weaklu	(60,60)	4.69	51.16	120	96	4637	0.42	8.26
envl	30	6.23	35.67	132	101	4380	0.24	5.84
envl	45	4.77	50.76	232	115	1465	0.36	7.32
envl	60	3.23	66.58	344	133	1135	0.42	14.65
envlu	(30,30)	6.23	35.67	132	101	4378	0.25	5.94
envlu	(45,45)	4.77	50.76	232	121	1358	0.36	7.46
envlu	(60,60)	3.23	66.58	344	137	1151	0.43	15.57
liftenvl	30	6.21	35.86	133	134	4736	0.25	7.20
liftenvl	45	4.75	50.99	233	136	1641	0.38	8.42
liftenvl	60	3.22	66.73	344	140	1075	0.44	14.93
weakl+env	(30,30)	6.21	35.86	163	122	4758	0.26	6.84
weakl+env	(45,45)	4.75	50.99	278	113	1608	0.38	8.15
weakl+env	(60,60)	3.22	66.73	404	131	1162	0.44	15.80
bc	-	9.68	-	0	265	46631	0.00	53.76
bclift	-	3.04	67.74	251⋆	81	2754	1.85	7.41

⋆Number of lifted bottleneck cutting planes generated

Table 2-3. Summary of experiments for $T = 120$, $c = 2, 3$.

		c=2						
exp	**stage**	**initgap**	**gapimp**	**Dpineq**	**CPXineq**	**nodes**	**ineqtime**	**time**
base	-	2.89	-	0	188	751284	0.00	407.74
weakl	40	2.21	22.86	40	199	213389	0.06	188.11
weakl	60	1.82	35.91	60	165	112931	0.19	105.45
weakl	80	1.43	48.86	80	137	68199	0.32	79.10
weaklu	(40,40)	2.23	22.41	80	199	269207	0.07	246.76
weaklu	(60,60)	1.84	35.33	120	167	174521	0.19	177.74
weaklu	(80,80)	1.45	48.15	160	126	146484	0.32	204.85
envl	40	1.88	34.67	199	153	72952	0.07	104.55
envl	60	1.45	50.08	328	173	15655	0.20	79.81
envl	80	1.03	63.97	485	160	8390	0.31	90.81
envlu	(40,40)	1.88	34.67	200	147	67957	0.07	81.40
envlu	(60,60)	1.48	49.33	330	177	12251	0.19	63.42
envlu	(80,80)	1.07	62.57	486	170	11322	0.31	119.10
liftenvl	40	1.87	35.00	201	159	48506	0.06	80.31
liftenvl	60	1.45	50.20	332	130	43182	0.19	144.10
liftenvl	80	1.02	63.85	489	152	8419	0.32	80.97
weakl+env	(40,40)	1.87	35.00	241	148	78774	0.07	117.29
weakl+env	(60,60)	1.44	52.71	392	142	13075	0.19	55.46
weakl+env	(80,80)	1.07	64.58	569	140	28436	0.32	170.14
		c=3						
base	-	5.08	-	0	251	894623	0.00	516.09
weakl	40	3.93	22.15	40	240	409545	0.23	345.65
weakl	60	3.18	36.15	60	207	160543	0.46	160.68
weakl	80	2.39	51.30	80	173	33339	0.58	35.08
weaklu	(40,40)	3.95	21.65	80	243	442524	0.23	333.01
weaklu	(60,60)	3.21	35.54	120	212	253128	0.45	276.53
weaklu	(80,80)	2.43	50.60	160	183	73292	0.58	116.56
envl	40	3.38	33.39	196	206	61544	0.23	64.40
envl	60	2.58	48.61	345	226	51191	0.45	132.34
envl	80	1.75	64.98	506	238	5308	0.58	71.47
envlu	(40,40)	3.38	33.39	196	195	61571	0.22	64.88
envlu	(60,60)	2.58	48.61	345	217	51925	0.45	133.37
envlu	(80,80)	1.75	64.98	506	235	5242	0.58	66.78
liftenvl	40	3.36	33.79	198	213	73793	0.23	91.37
liftenvl	60	2.57	48.87	347	218	14417	0.45	51.66
liftenvl	80	1.73	65.30	508	232	6724	0.58	93.13
weakl+env	(40,40)	3.36	33.79	238	210	86022	0.23	86.32
weakl+env	(60,60)	2.57	48.87	407	230	20323	0.46	64.53
weakl+env	(80,80)	1.73	65.30	588	192	9223	0.57	87.82

Table 2-4. Summary of experiments for $T = 120$, $c = 4, 5$.

		c=4						
exp	**stage**	**initgap**	**gapimp**	**Dpineq**	**CPXineq**	**nodes**	**ineqtime**	**time**
base	-	7.29	-	0	188	666567	0.00	388.91
weakl	40	5.53	23.59	40	202	370266	0.42	285.68
weakl	60	4.45	38.22	60	168	188771	0.67	151.83
weakl	80	3.33	53.49	80	158	65223	0.78	60.87
weaklu	(40,40)	5.57	23.02	80	198	416699	0.41	294.18
weaklu	(60,60)	4.50	37.57	120	183	201402	0.66	224.51
weaklu	(80,80)	3.38	52.76	160	151	73192	0.78	99.25
envl	40	4.78	33.93	191	178	134469	0.41	144.22
envl	60	3.68	49.21	336	221	24060	0.65	88.54
envl	80	2.45	66.06	505	224	11677	0.78	163.49
envlu	(40,40)	4.78	33.93	192	166	135039	0.41	143.15
envlu	(60,60)	3.68	49.21	336	207	22382	0.65	77.20
envlu	(80,80)	2.45	66.06	505	226	11443	0.81	163.54
liftenvl	40	4.76	34.29	192	190	133103	0.42	152.21
liftenvl	60	3.65	49.61	337	166	17757	0.65	54.94
liftenvl	80	2.43	66.34	505	160	14904	0.78	158.61
weakl+env	(40,40)	4.76	34.28	232	175	149968	0.41	166.42
weakl+env	(60,60)	3.65	49.61	397	172	15978	0.66	52.17
weakl+env	(80,80)	2.43	66.34	585	216	5711	0.78	68.33
		c=5						
base	-	9.24	-	0	173	696683	0.00	321.65
weakl	40	6.96	24.54	40	189	349971	0.53	216.19
weakl	60	5.55	39.58	60	176	136393	0.78	114.15
weakl	80	4.13	54.98	80	153	121461	0.91	113.13
weaklu	(40,40)	7.01	23.94	80	167	360449	0.52	263.32
weaklu	(60,60)	5.61	38.87	120	162	229883	0.77	184.16
weaklu	(80,80)	4.20	54.25	160	138	112402	0.90	127.53
envl	40	6.03	34.82	195	171	94289	0.52	91.14
envl	60	4.51	51.07	342	242	7571	0.77	35.62
envl	80	3.13	66.21	502	225	4000	0.90	47.37
envlu	(40,40)	6.03	34.82	195	171	94289	0.52	91.20
envlu	(60,60)	4.51	51.07	342	242	7571	0.77	35.92
envlu	(80,80)	3.13	66.21	502	225	4000	0.90	47.75
liftenvl	40	6.01	35.02	196	206	76220	0.53	81.22
liftenvl	60	4.50	51.22	343	204	7079	0.77	35.67
liftenvl	80	3.11	66.42	502	238	3604	0.90	49.11
weakl+env	(40,40)	6.01	35.02	236	185	77399	0.52	78.71
weakl+env	(60,60)	4.50	51.22	403	242	7717	0.77	38.47
weakl+env	(80,80)	3.11	66.41	582	215	4987	0.90	59.59

Table 2-5. Summary of experiments for $T = 150$, $c = 2, 3$.

		c=2						
exp	**stage**	**initgap**	**gapimp**	**Dpineq**	**CPXineq**	**nodes**	**ineqtime**	**time**
base	-	2.73	-	0	239	1579484	0.00	1064.41^{9}
weakl	50	2.05	23.89	80	259	885970	0.12	892.82^{7}
weakl	75	1.67	37.01	125	223	567813	0.38	701.65^{4}
weakl	100	1.24	51.95	172	183	305753	0.61	434.14^{2}
weaklu	(50,50)	2.09	22.49	100	254	965459	0.12	925.25^{7}
weaklu	(75,75)	1.73	35.29	150	204	591190	0.38	679.43^{4}
weaklu	(100,100)	1.30	49.91	200	171	273983	0.62	451.04^{2}
envl	50	1.82	32.90	256	268	316056	0.12	463.54^{2}
envl	75	1.41	48.09	431	208	125114	0.37	495.38^{3}
envl	100	0.96	63.65	636	226	35399	0.62	368.54^{1}
envlu	(50,50)	1.82	32.88	257	258	421850	0.12	554.18^{2}
envlu	(75,75)	1.41	48.09	432	189	103054	0.37	360.71^{1}
envlu	(100,100)	0.96	63.61	637	187	38830	0.61	383.71^{2}
liftenvl	50	1.81	33.35	259	234	328544	0.12	451.23^{1}
liftenvl	75	1.38	48.95	435	215	64663	0.37	298.55^{1}
liftenvl	100	0.95	64.17	640	194	18453	0.60	251.67
weakl+env	(50,50)	1.80	33.41	309	185	381673	0.12	595.95^{2}
weakl+env	(75,75)	1.38	48.97	510	203	85180	0.37	328.28^{1}
weakl+env	(100,100)	0.95	64.06	740	247	30820	0.60	408.36^{1}
		c=3						
base	-	5.06	-	0	283	1447386	0.00	1028.15^{10}
weakl	50	3.74	25.35	65	296	884541	0.45	882.69^{7}
weakl	75	2.90	41.21	101	230	486848	0.91	596.34^{4}
weakl	100	2.15	55.62	139	204	135059	1.15	206.25
weaklu	(50,50)	3.77	24.58	100	268	1002471	0.44	982.90^{9}
weaklu	(75,75)	2.98	39.96	150	241	612371	0.93	760.89^{7}
weaklu	(100,100)	2.21	54.53	200	191	307076	1.15	621.78^{4}
envl	50	3.29	34.84	264	306	307212	0.44	514.67^{3}
envl	75	2.42	51.57	463	314	81871	0.91	321.18^{2}
envl	100	1.70	65.60	691	324	33107	1.15	523.21^{3}
envlu	(50,50)	3.28	34.88	264	319	292334	0.45	512.97^{3}
envlu	(75,75)	2.41	51.62	463	337	48985	0.91	301.19^{1}
envlu	(100,100)	1.69	65.68	692	303	28517	1.16	400.37^{1}
liftenvl	50	3.23	35.80	265	294	338422	0.45	540.73^{4}
liftenvl	75	2.38	52.29	464	326	42549	0.92	237.11^{1}
liftenvl	100	1.69	65.75	693	291	23607	1.16	392.94
weakl+env	(50,50)	3.23	35.79	315	318	229984	0.45	465.70^{3}
weakl+env	(75,75)	2.38	52.28	539	278	99341	0.92	405.49^{2}
weakl+env	(100,100)	1.71	65.53	793	269	31334	1.15	431.94^{2}

Superscript in **time** column refers to number of instances not solved to optimality.

Table 2-6. Summary of experiments for $T = 150$, $c = 4, 5$.

		c=4						
exp	**stage**	**initgap**	**gapimp**	**Dpineq**	**CPXineq**	**nodes**	**ineqtime**	**time**
base	-	6.95	-	0	289	984151	0.00	736.19^{6}
weakl	50	5.13	25.90	50	277	458784	0.87	474.74^{3}
weakl	75	4.05	41.13	75	227	276947	1.49	342.51^{1}
weakl	100	2.98	56.31	100	217	105854	1.76	161.96^{1}
weaklu	(50,50)	5.16	25.47	100	289	513579	0.79	520.91^{3}
weaklu	(75,75)	4.10	40.51	150	234	208360	1.29	314.40^{1}
weaklu	(100,100)	3.02	55.77	200	195	87028	1.58	197.97^{1}
envl	50	4.54	34.69	275	259	217313	0.79	448.19^{3}
envl	75	3.39	51.24	472	285	33888	1.28	192.86^{1}
envl	100	2.32	66.41	703	279	26463	1.55	367.80^{2}
envlu	(50,50)	4.54	34.69	275	267	224367	0.79	445.14^{3}
envlu	(75,75)	3.39	51.24	472	285	36664	1.28	205.27^{1}
envlu	(100,100)	2.32	66.41	703	278	26947	1.55	367.54^{2}
liftenvl	50	4.52	34.93	276	221	196258	0.80	355.01^{2}
liftenvl	75	3.38	51.39	473	321	39920	1.28	202.73^{1}
liftenvl	100	2.32	66.26	705	315	24424	1.72	294.89^{1}
weakl+env	(50,50)	4.53	34.88	326	226	194000	0.80	384.58^{2}
weakl+env	(75,75)	3.38	51.38	548	291	38633	1.32	203.24^{1}
weakl+env	(100,100)	2.27	67.22	805	282	11896	1.57	252.59
		c=5						
base	-	9.11	-	0	252	1206288	0.00	855.33^{9}
weakl	50	6.64	26.81	65	220	867700	1.02	706.58^{5}
weakl	75	5.29	41.58	99	184	455875	1.51	443.15^{3}
weakl	100	3.89	56.82	137	190	199708	1.78	257.56^{2}
weaklu	(50,50)	6.74	25.87	100	214	900826	1.02	796.77^{6}
weaklu	(75,75)	5.39	40.62	150	172	459680	1.51	549.01^{4}
weaklu	(100,100)	3.98	55.88	200	165	183244	1.77	343.75^{1}
envl	50	5.87	35.50	265	304	265372	1.02	479.01^{2}
envl	75	4.50	50.45	460	338	47930	1.51	246.40
envl	100	3.08	66.21	689	368	32008	1.77	409.10^{2}
envlu	(50,50)	5.87	35.50	265	304	276529	1.03	481.48^{2}
envlu	(75,75)	4.50	50.45	460	327	47929	1.50	246.19
envlu	(100,100)	3.08	66.21	689	365	32102	1.78	413.15^{2}
liftenvl	50	5.87	35.60	265	271	302203	1.02	491.66^{3}
liftenvl	75	4.50	50.50	460	301	62235	1.51	247.34^{1}
liftenvl	100	3.07	66.32	689	275	29753	1.77	369.24^{2}
weakl+env	(50,50)	5.87	35.58	315	275	250109	1.01	394.90^{2}
weakl+env	(75,75)	4.49	50.58	535	263	54553	1.52	227.33^{1}
weakl+env	(100,100)	3.12	65.92	789	330	27755	1.77	409.42^{2}

Superscript in **time** column refers to number of instances not solved to optimality.

Further note that at least one strategy was able to solve all instances for a given value of c. Specifically, liftenv(100) solved all instances for $c = 2$ and 3; weakl+env(100,100) for $c = 4$; and envlu(75,75) for $c = 5$.

In all experiments described thus far, the generation of the inequalities via DP takes a few CPU seconds in the worst case and indeed, the problems can be solved to optimality most effectively by a full application of DP. Noting that the CLSP structure is often included in much more difficult problems (such as multi-item lot sizing problems), our intent has been to illustrate that these inequalities have the potential to be useful in a broader domain of applications that cannot effectively be solved by DP. However, an interesting question that we address in a final experiment is whether or not our proposed methods are preferable to DP on a different set of randomly generated CLSP instances.

We generated ten instances with $T = 90$, a capacity-to-demand ratio $c = 3$ and a setup-to-holding-cost ratio $f = 10000$. The unit production costs p_t, demands d_t, capacities c_t and setup costs s_t were randomly generated with uniform distributions with ranges as follows: $p_t \in [1, 5]$, $d_t \in [0, 600]$, $c_t \in \left[0.7c\bar{d}, 1.1c\bar{d}\right]$, $s_t \in \left[0.90f\bar{h}, 1.10f\bar{h}\right]$. Note that the possible spread in demand makes executing the DP recursion challenging. The holding cost h_t was fixed at 1 for each period.

The results of this experiment are given in Table 2-7. Here, we compare the weakl+env method with DP inequalities generated at various numbers of stages with the base strategy, and with the complete solution by DP. As before, our weakl+env outperformed base when solving 1/9, 1/6, and 1/3 of the possible DP stages. However, these instances display the potential trade-off in computational benefit resulting from tightening the base model with DP based inequalities and the computational expense associated with generating these inequalities. As the number of stages increases, the time spent by CPLEX *after* the generation of the DP based inequalities decreases. However, factoring in the computational effort required to generate these inequalities, the best overall implementation tested uses 15 stages, which is roughly 34 percent faster on average than the DP approach (given in column 1).

Table 2-7. Experiments for $T = 90$ and $f = 10000$.

runno	exp	stage	initgap	gapimp	Dpineq	CPXineq	ineqtime	time
	base	-	6.82	0.00	0	54	0.00	851.14
Instance 1	weakl+env	(10+10)	6.14	9.97	25	54	3.28	456.22
DP time = 205	weakl+env	(15+15)	5.94	12.90	34	215	8.80	163.54
	weakl+env	(30+30)	5.08	25.51	63	53	59.55	134.62
	base	-	7.68	0.00	0	61	0.00	172.47
Instance 2	weakl+env	(10+10)	6.38	16.93	36	219	2.13	88.92
DP time = 155	weakl+env	(15+15)	6.32	17.71	45	244	6.69	94.39
	weakl+env	(30+30)	5.96	22.40	75	53	53.80	156.14
	base	-	7.36	0.00	0	212	0.00	24.73
Instance 3	weakl+env	(10+10)	5.8	21.20	31	201	3.86	30.58
DP time = 219	weakl+env	(15+15)	5.7	22.55	41	208	10.73	48.56
	weakl+env	(30+30)	4.75	35.46	70	193	67.28	92.56
	base	-	6.53	0.00	0	59	0.00	175.19
Instance 4	weakl+env	(10+10)	5.86	10.26	31	53	4.11	252.22
DP time = 229	weakl+env	(15+15)	5.78	11.49	41	56	10.14	363.08
	weakl+env	(30+30)	5.02	23.12	68	47	60.44	350.48
	base	-	9.02	0.00	0	258	0.00	219.89
Instance 5	weakl+env	(10+10)	7.84	13.08	25	56	6.44	356.42
DP time = 225	weakl+env	(15+15)	7.57	16.08	35	220	16.63	174.37
	weakl+env	(30+30)	6.29	30.27	65	179	93.31	204.73
	base	-	7.22	0.00	0	233	0.00	491.48
Instance 6	weakl+env	(10+10)	5.8	19.67	28	52	3.55	667.84
DP time = 220	weakl+env	(15+15)	5.68	21.33	38	227	10.11	249.03
	weakl+env	(30+30)	4.94	31.58	68	183	63.44	219.92
	base	-	6.73	0.00	0	234	0.00	172.83
Instance 7	weakl+env	(10+10)	5.61	16.64	33	53	2.72	389.92
DP time = 231	weakl+env	(15+15)	5.38	20.06	40	54	8.25	204.95
	weakl+env	(30+30)	4.75	29.42	69	208	61.92	217.70
	base	-	6.78	0.00	0	223	0.00	117.22
Instance 8	weakl+env	(10+10)	5.41	20.21	35	214	2.80	91.56
DP time = 256	weakl+env	(15+15)	5.34	21.24	44	208	7.99	85.64
	weakl+env	(30+30)	4.61	32.01	74	176	60.14	108.26
	base	-	7.32	0.00	0	50	0.00	26.56
Instance 9	weakl+env	(10+10)	5.96	18.58	35	43	4.02	49.06
DP time = 241	weakl+env	(15+15)	5.75	21.45	43	50	10.38	29.59
	weakl+env	(30+30)	5.56	24.04	72	48	64.05	94.16
	base	-	5.95	0.00	0	64	0.00	47.41
Instance 10	weakl+env	(10+10)	5.12	13.95	24	202	2.80	29.33
DP time = 235	weakl+env	(15+15)	5.04	15.29	33	56	8.06	52.95
	weakl+env	(30+30)	4.66	21.68	62	193	60.89	116.53
	base	-	7.14	0.00	0	145	0.00	229.89
Average	weakl+env	(10+10)	5.99	16.05	30	115	3.57	241.21
DP time = 222	weakl+env	(15+15)	5.85	18.01	39	154	9.78	146.61
	weakl+env	(30+30)	5.16	27.55	69	133	64.48	169.51

2.5 Summary

In this chapter, we identify a set of DP based inequalities to strengthen the CLSP MIP formulation by using iterative solutions of forward and backward DP formulations of the problem. The convex envelope of the stage value function provides efficient inequalities, which can then be lifted by investigating potential state information at future stages. Our best implementation appends an initial set of inequalities (2–12) to the traditional MIP formulation based on stage information from the partial execution of a forward DP recursion and then adds lifted cutting-plane inequalities in a cut-and-branch fashion at the root node.

For future research, we will explore the use of the inequalities developed in this paper within problem domains that contain the CLSP constraints as a substructure. One natural extension of this work would investigate more complex variations of the CLSP, such as the multi-item CLSP, which we consider in Chapter 3. Given the success of our approach on the CLSP, we believe that a similar approach may prove effective for other combinatorial problems as well.

CHAPTER 3
MULTI-ITEM CAPACITATED LOT-SIZING PROBLEM

3.1 Introduction

We consider the multi-item capacitated lot-sizing problem (MCLSP), in which we determine production and inventory levels for each item in each time period over a finite horizon in order to meet periodic demands. In the MCLSP, capacity limits restrict the total production of all items during each time period. The objective is to minimize the sum of periodic set-up, production, and inventory costs of all items. The MCLSP is NP-hard in the strong sense (Florian et al. (1980)), and forms the basis of many production planning and inventory problems.

The data for MCLSP is summarized below.

M: Number of items;

T: Number of periods;

p_{it}: Per-unit production cost for item $i = 1, \ldots, M$ in period $t = 1, \ldots, T$;

h_{it}: Per-unit inventory holding cost for item $i = 1, \ldots, M$ in period $t = 1, \ldots, T$;

f_{it}: Set-up cost that must be incurred for any production of item $i = 1, \ldots, M$ in period $t = 1, \ldots, T$;

c_t: Total capacity in period $t = 1, \ldots, T$.

The decision variables are given as follows:

x_{it}: Amount of item $i = 1, \ldots, M$ produced in period $t = 1, \ldots, T$;

s_{it}: Amount of item $i = 1, \ldots, M$ placed in inventory in period $t = 1, \ldots, T$;

y_{it}: Binary variable that equals 1 if production for item $i = 1, \ldots, M$ occurs in period $t = 1, \ldots, T$, else 0.

The MCLSP can be formulated as the following mixed integer program:

$$\min \quad \sum_{i=1}^{M} \sum_{t=1}^{T} (p_{it} x_{it} + f_{it} y_{it} + h_{it} s_{it}) \tag{3–1a}$$

$$\text{s.t.} \quad s_{i,t-1} + x_{it} - d_{it} = s_{it} \qquad t = 1, \ldots, T,\ i = 1, \ldots, M \tag{3–1b}$$

$$x_{it} \leq \min\{c_t, \sum_{j=t}^{T} d_{ij}\} y_{it} \qquad t = 1, \ldots, T,\ i = 1, \ldots, M \tag{3–1c}$$

$$\sum_{i=1}^{M} x_{it} \leq c_t \qquad t = 1, \ldots, T \tag{3–1d}$$

$$s_{it}, x_{it} \geq 0 \qquad t = 1, \ldots, T,\ i = 1, \ldots, M \tag{3–1e}$$

$$y_{it} \in \{0, 1\} \qquad t = 1, \ldots, T,\ i = 1, \ldots, M. \tag{3–1f}$$

The objective function (3–1a) minimizes the sum of production, set-up, and inventory costs. Constraints (3–1b) define the remaining inventory for each product after each time period. Constraints (3–1c) force the binary set-up variable y_{it} to equal 1 whenever x_{it} is positive, and ensure that production for an item in a period cannot exceed the minimum of the capacity in that period and the corresponding remaining demand through the end of horizon. Constraints (3–1d) represent the shared capacity constraint linking the production of different items. Finally, constraints (3–1e) and (3–1f) state bounds and integer restrictions on the variables. MCLSP differs from CLSP by incorporating multiple items and the shared capacity restriction on them.

Lot-sizing is a fundamental problem in optimization with important applications in production/inventory planning and supply chain management. In particular, the MCLSP structure (Atamtürk and Küçükyavuz (2008); Atamtürk and Muñoz (2004); Pochet and Wolsey (1991); Wolsey (1958)) is at the core of the production planning problems involving multiple products and levels over a finite discrete time horizon.

Polyhedral studies on the MCLSP are devoted to strengthening the problem formulation by deriving valid inequalities via analysis of knapsack, single node flow and single period relaxations of the problem (Miller (1999); Miller et al. (2000, 2003b); Padberg et al. (1985); Pochet and Wolsey (2006)). These approaches focus on the MCLSP polyhedron without consideration of the objective function. Our approach is to use the objective function to guide us in adding valid inequalities that serve to tighten the feasible region in the part of the polyhedron where an optimal solution lies. In Chapter 2 we generate valid inequalities for the single-item capacitated lot-sizing problem that are derived from the end-of stage optimal solutions of a dynamic programming (DP)

algorithm. Motivated by this approach, we aim to provide new valid inequalities for the lot-sizing polytope by obtaining valid bounds on the partial objective functions. We then study techniques by which we can improve these inequalities via the use of lifting techniques. We also implement effective solution algorithms using the cutting planes developed for the MCLSP.

The rest of Chapter 3 is outlined as follows. In Section 3.2 we propose partial objective function inequalities for the MCLSP. In Section 3.3 we discuss lifting and separation problems associated with the partial objective inequalities. In Section 3.4 we present computational results to demonstrate the effectiveness of the proposed inequalities and the lifting procedure with a separation algorithm.

3.2 Valid Inequalities

In this section, we decompose formulation (3–1) into M single-item problems, where each individual item can use the remaining capacity after allotting enough capacity to each other item in order to meet demand in each period. We extend the valid inequalities in Chapter 2 for the single-item CLSPs to the k-item case, where we decompose the problem into two-item problems and utilize dynamic programming formulations to derive valid inequalities for the MCLSP. In addition, we consider integer programming based approaches to solve k-item, t-period relaxations of the problem ($k \leq M$, $t \leq T$) to obtain valid bounds on the partial MCLSP objective function.

3.2.1 Single-Item Partial Objective Inequalities

We begin by summarizing the DP based inequalities given in Chapter 2 for the single-item capacitated lot-sizing problem. Suppose that we have employed this algorithm and have obtained the optimal cost $F_j(s_j)$ to feasibly accumulate s_j units of inventory, for all stages $j = \{1, \ldots, T\}$ and all possible state values for s_j.

For any $t = 1, \ldots, T$, the following inequality is valid:

$$\sum_{j=1}^{t} (p_j x_j + f_j y_j + h_j s_j) \geq F_t(L_t), \tag{3–2}$$

where L_t denotes the minimum inventory level at period t and so $F_t(L_t)$ represents the minimum cost of all feasible decisions through period t. Here, we omit the item index "i" since we consider a single item.

We can compute stronger inequalities than (3–2) by examining all costs accrued through time t as a function of the inventory s at stage t. While the stage t value function $F_t(s_t)$ is not necessarily a convex function of s_t, the inequalities that define the convex envelope of $F_t(s_t)$ are valid for the CLSP. Such inequalities have the form:

$$\sum_{j=1}^{t} (p_j x_j + f_j y_j + h_j s_j) \geq m s_t + b, \tag{3–3}$$

for the slope and the intercept parameters m and b, respectively.

Now suppose that we decompose an MCLSP instance into M single-item CLSPs, where each CLSP allows its item to consume all of the allowable production capacity at each time stage. The inequalities given in (3–2) and (3–3) that we derive from any CLSP are valid for the MCLSP formulation (3–1), since they are based on single-item relaxations of (3–1).

In the context of the MCLSP, we can strengthen these inequalities by incorporating necessary conditions for feasibility based on shared capacity constraints. First, given a single item i and the partial objective function $\sum_{j=1}^{t} (p_{ij} x_{ij} + f_{ij} y_{ij} + h_{ij} s_{ij})$ corresponding to the first $t = 1, \ldots, T$ periods, we can generate valid lower bounds on this function by minimizing this partial objective function subject to the constraints (3–1b)–(3–1f). In this problem, we only restrict $y_{i1}, \ldots, y_{it}$ to be binary, because an optimal solution will exist in which all other y-variables are set to 1. In addition, we can either optimize this problem or use the lower bound obtained after a predetermined solution time limit. Letting L be the best lower bound within the allotted time limit over the first t stages, the following inequality is valid:

$$\sum_{j=1}^{t} (p_{ij} x_{ij} + f_{ij} y_{ij} + h_{ij} s_{ij}) \geq L. \tag{3–4}$$

Next, we note that inequalities (3–2) and (3–3) generated from intermediate stage solutions to the single-item CLSPs may be weakened due to the fact that we allot all production capacity to the item under examination. We thus tighten (3–2) and (3–3) by reducing the possible inventory levels allowed for item i at each stage due to necessary capacity utilization of the other items.

The minimum inventory level at period $t = 1, \ldots, T$ for item i in any optimal solution is given by:

$$LB_{it} = \max \left\{ 0, \max_{\tau=t+1,\ldots,T} \sum_{j=t+1}^{\tau} (d_{ij} - c_j) \right\}, \tag{3–5}$$

since inventory of item i must always be nonnegative and large enough to cover future demands for item i if capacity in future periods is not sufficient to cover these demands (see, e.g., Constantino (1996), Chapter 2).

To compute the maximum possible inventory level for item i at period $t = 1, \ldots, T$, let $C_t = \sum_{j=1}^{t} c_j$ and $D_{it} = \sum_{j=1}^{t} d_{ij}$. Then the maximum inventory level for item i at period t in any optimal solution is given by:

$$UB_{it} = \min \left\{ C_t - \sum_{i=1}^{M} D_{it} - \sum_{k \in M, k \neq i} LB_{kt}, \sum_{j=t+1}^{T} d_{ij} \right\}. \tag{3–6}$$

The first term of (3–6) gives the maximum inventory of item i that could accumulate after t periods. This term ensures that we satisfy the demand for all the items and allot enough capacity for the future uncovered demand corresponding to the other items. The second term of (3–6) gives the cumulative demand for item i that must be satisfied in future periods.

3.2.2 Multi-Item Partial Objective Inequalities

Consider some subset $K \subseteq \{1, \ldots, M\}$. Then for any $t = 1, \ldots, T$, the following inequality is valid for MCLSP:

$$\sum_{i \in K} \sum_{j=1}^{t} (p_{ij} x_{ij} + f_{ij} y_{ij} + h_{ij} s_{ij}) \geq F_{kt}, \tag{3–7}$$

where F_{kt} represents the minimum cost to satisfy production requirements of items in K through period t.

We calculate minimum and maximum inventory levels for multi-item problems at period $t = 1, \ldots, T$ in any optimal solution as follows. Given $K = \{i_1, \ldots, i_k\}$, the minimum inventory level at period $t = 1, \ldots, T$ in any optimal solution is given by:

$$LB_{i_1,\ldots,i_k,t} = \max\left\{0, \max_{\tau=t+1,\ldots,T} \sum_{j=t+1}^{\tau} \left(\sum_{i\in K} d_{ij} - c_j\right)\right\}, \tag{3–8}$$

and the maximum inventory level at period $t = 1, \ldots, T$ in any optimal solution is given by:

$$UB_{i_1,\ldots,i_k,t} = \min\left\{C_t - \sum_{i=1}^{M} D_{it} - \sum_{k\in M, k\neq i_1,\ldots,i_k} LB_{kt}, \sum_{j=t+1}^{T}\sum_{i\in K} d_{ij}\right\}. \tag{3–9}$$

Note that we also compute the lower and upper inventory bounds for each individual item using (3–5) and (3–6), since individual bounds may also help to reduce the state space. For instance, suppose that the maximum inventory level given by (3–9) for two items in a stage is 3 units, while maximum individual inventory levels for these items given by (3–6) are 1 and 2, respectively. Let the states for a two-item DP algorithm be given by a pair of inventory values for the items. Then the individual inventory bound (3–6) eliminates the states $(0, 3)$, $(3, 0)$ and $(2, 1)$ from the state space. Similarly, if the minimum inventory level given by (3–8) for the two items is 2, and the individual minimum inventory levels are 2 and 0, respectively, then we can also eliminate states $(1, 1)$ and $(0, 2)$.

The DP recursion for the multi-item problem is similar to the DP recursion for single-item problem (2–8) given in Chapter 2. For period $t = 1, \ldots, T$, given inventory level $LB_{it} \leq s_{it} \leq UB_{it}$ for each item i, the production in any optimal solution at period t lies in the set $X_{t,s_{it}} = \{\max\{0, s_{it} + d_{it} - \min\{UB_{i,t-1}, UB_{i_1,\ldots,i_k,t}\}\}, \ldots, \min\{c_t, s_{it} + d_{it} - LB_{it}\}\}$. Setting $F_0(0, \ldots, 0) = 0$ (since we have assumed $s_{i0} = 0$), the forward dynamic programming recursion for the multi-item problem can be written as:

$$F_t(s_{1t}, \ldots, s_{kt}) =$$

$$\min_{x_{it} \in X_{t,s_{it}}, \forall i \in K} \left\{ \sum_{i \in K} (p_{it} x_{it} + f_{it} y_{it} + h_{it} s_{it}) + F_{t-1}(s_{1t} + d_{1t} - x_{1t}, \ldots, s_{kt} + d_{kt} - x_{kt}) \right\},$$

$$\forall 1 \leq t \leq T,\ LB_{it} \leq s_{it} \leq UB'_{it}, \quad (3\text{–}10)$$

where $UB'_{it} = \min\{UB_{i,t}, UB_{i_1,\ldots,i_k,t}\}$, and $y_{it} = 1$ if $x_{it} > 0$ and $y_{it} = 0$ otherwise. The optimal objective function is defined as $F_T(0, \ldots, 0)$.

This dynamic programming formulation can be represented as an acyclic graph, as shown in Figure 3-1. The nodes represent feasible states corresponding to the possible inventory levels of the items in each period, while the arcs represent feasible decisions for each state. The nodes in Figure 3-1 are labeled with the inventory at the end of each period for each item $i \in K$. Here, the arc lengths are defined by the costs associated with each decision arc. Note that all the arcs in Figure 3-1 are not presented for clarity but if there is an arc from a state s_t to another state s_{t+1}, then there must also be an arc from s_t to each of all possible inventory values lower than s_{t+1}. The goal is to find a shortest path connecting node $(0, \ldots, 0)$ in period 0, representing the initial state of the system (no inventory at time zero), to node $(0, \ldots, 0)$ in period T, representing the final period of the problem when no inventory is needed as the problem terminates.

The number of nodes generated for a given instance is $\sum_{t=1}^{T-1} \prod_{i \in K} (UB'_{it} - LB_{it}) + 2$, and the maximum number of arcs is $\sum_{t=1}^{T-1} \prod_{i \in K} (UB'_{it} - LB_{it}) \prod_{i \in K} (UB'_{i,t+1} - LB_{i,t+1}) + \prod_{i \in K} (UB'_{i1} - LB_{i,1}) + \prod_{i \in K} (UB'_{i,T-1} - LB_{i,T-1})$. The number of arcs determines the complexity of solving this problem by dynamic programming, and letting $UB - LB = UB'_{it} - LB_{it}$ for each $t = 1, \ldots, T$, the complexity of the algorithm is given by $O(KT + T(UB - LB)^{2K})$, as all LB_{it} and UB_{it} can be defined in $O(KT)$ time. The complexity of the dynamic programming algorithm is exponential due to the term $(UB - LB)^{2K}$.

Since, as $|K|$ grows, the state space in a dynamic programming approach to solve the problem grows exponentially, we consider computing F_{kt}-values for the case in which

$|K| \leq 2$ using DP. We first compute all possible states $(s_{i_1 j}, s_{i_2 j})$ for each time period $j = 1, \ldots, t$, and then use the DP recursion (3–10) to compute the optimal state function values $F_j(s_{i_1 j}, s_{i_2 j})$, for each possible time period and state. Note that here, $F_j(s_{i_1 j}, s_{i_2 j})$ denotes the minimum cost to feasibly accumulate $s_{i_1 j}$ units of inventory for item 1 and $s_{i_2 j}$ units of inventory for item 2 through state j.

After computing the state space and corresponding functional values with DP for each t, we derive inequalities that define the entire convex hull of the points defined by $(s_{i_1 t}, s_{i_2 t}, F_t(s_{i_1 t}, s_{i_2 t}))$ by using the Gift Wrapping algorithm based on Jarvis's march algorithm (Jarvis (1973)). This approach projects all points onto the two-dimensional space by replacing $(s_{i_1 t}, s_{i_2 t}, F_t(s_{i_1 t}, s_{i_2 t}))$ with $(s_{i_1 t}, s_{i_2 t}, 0)$. We begin by determining a one-dimensional edge of the three-dimensional convex hull, which passes through points r_1 and r_2. This edge can be found by projecting the given set of points onto any two dimensions, finding a point q_1 having the smallest value in one dimension of the projection, and then finding a projected point q_2 such that no projected points lie strictly on both sides of the line spanning q_1 and q_2. We take r_1 and r_2 to be the three-dimensional counterparts of projected points q_1 and q_2, respectively. Next, note that the three-dimensional convex hull contains at most two facets that pass through r_1 and r_2. We find a third point r_3 such that the plane passing through r_1, r_2 and r_3 induces a facet to the convex hull of points, again by examining each of the other points to ensure that they lie only on one side of the plane (for now, assuming that there are no points that are affine combinations of r_1, r_2 and r_3). Then, we put edges (r_1, r_2), (r_1, r_3) and (r_2, r_3) in a queue. From this point, the algorithm selects and removes an edge from the queue and finds the other facet passing through the edge. If a new facet is identified, the two new edges defining this facet are added to the queue, unless they are already present. If an edge is present, it is removed from the queue, since the second facet including this edge has been found. At any point, if there exists more than three points that lie on a generated plane of the convex hull, we find the two-dimensional convex hull of those

points, obtain the edges of this plane, and add the edges to the queue (or subtract them if they are already present). The process stops when the queue is empty. The Gift Wrapping algorithm constructs the convex hull in $O(nl)$ time, where l is the number of vertices of the convex hull and n is the number of points.

For the $|K| = 2$ item lot-sizing problem, the inequalities defining the lower convex hull of the stage t value function have the form:

$$\sum_{i=1}^{2}\sum_{j=1}^{t}(p_{ij}x_{ij} + f_{ij}y_{ij} + h_{ij}s_{ij}) \geq a_{tq}s_{i_1t} + b_{tq}s_{i_2t} + c_{tq} \tag{3–11}$$

for parameters a_{tq}, b_{tq} and c_{tq}, $q = 1, \ldots, Q_t$, where Q_t is the number of facets defining the lower convex hull, and the inequalities defining the upper hull have the form:

$$\sum_{i=1}^{2}\sum_{j=1}^{T}(p_{ij}x_{ij} + f_{ij}y_{ij} + h_{ij}s_{ij}) \leq \tilde{a}_{t\tilde{q}}s_{i_1t} + \tilde{b}_{t\tilde{q}}s_{i_2t} + \tilde{c}_{t\tilde{q}} \tag{3–12}$$

for parameters $\tilde{a}_{tq}$, $\tilde{b}_{tq}$ and $\tilde{c}_{tq}$, $\tilde{q} = 1, \ldots, \tilde{Q}_t$, where $\tilde{Q}_t$ is the number of facets defining the upper convex hull.

Example 2. To illustrate the valid inequalities, consider an instance of MCLSP with $T = 4,\ M = 2,\ c_t = (4, 5, 3, 5),\ d_{1t} = (1, 2, 1, 2),\ d_{2t} = (2, 2, 1, 2),\ p_{1t} = (8, 9, 8, 8),\ p_{2t} = (11, 5, 9, 9),\ f_{1t} = (7, 8, 5, 4),\ f_{2t} = (6, 5, 6, 5)$ and $h_{1t} = h_{2t} = 1$ for $t = 1, 2, 3, 4$. Figure 3-1 gives the DP representation of this instance, with each node representing a feasible state (feasible inventory levels for item 1 and item 2, respectively) in each period. The F-values are provided above each node in the figure.

The optimal solution is represented by the bold path in Figure 3-1. Production for item 1 and production for item 2 is given by $x_{1t} = (2, 1, 3, 0)$ and $x_{2t} = (2, 4, 0, 1)$, with a total cost of 142.

Now, let $K = \{1, 2\}$. For $t = 1$, inequality (3–7) is:

$$8x_{11} + 7y_{11} + s_{11} + 11x_{21} + 6y_{21} + s_{21} \geq 43, \tag{3–13}$$

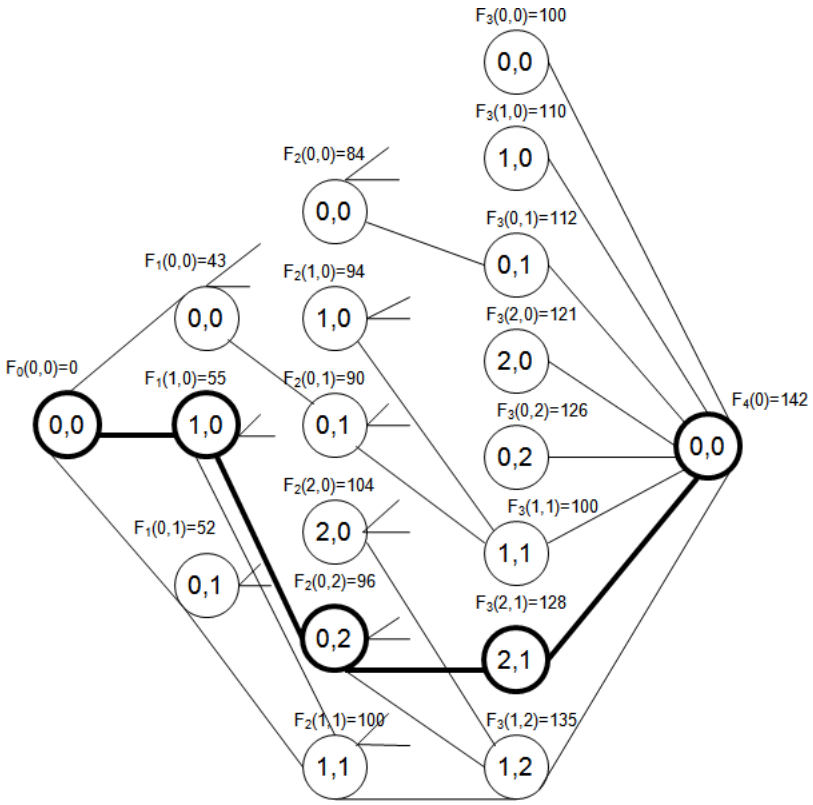

Figure 3-1. Network representation of the DP formulation of MCLSP for $T = 4$ and $M = 2$.

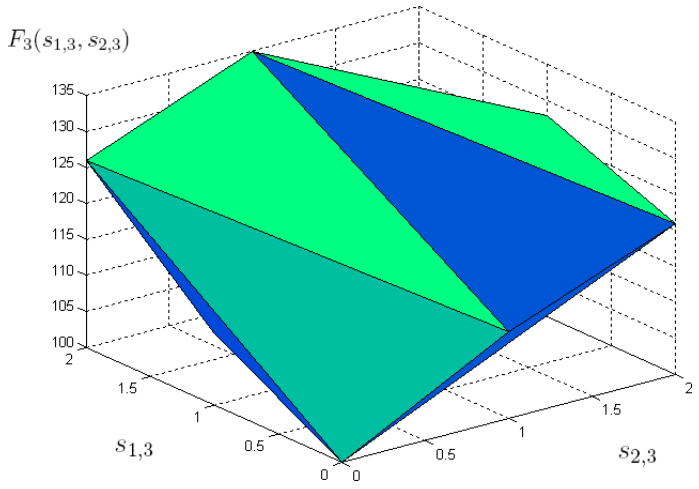

Figure 3-2. Convex hull defining the functional values for $t = 3$.

and for $t = 2$, inequality (3–7) is:

$$8x_{11} + 7y_{11} + s_{11} + 11x_{21} + 6y_{21} + s_{21} + 9x_{12} + 8y_{12} + s_{12} + 5x_{22} + 5y_{22} + s_{22} \geq 84. \quad (3\text{–}14)$$

The single inequality that defines the entire convex hull for $F_2(s_{1,2}, s_{2,2})$, from (3–11), is:

$$\begin{aligned} &8x_{11} + 7y_{11} + s_{11} + 11x_{21} + 6y_{21} + s_{21} + \\ &9x_{12} + 8y_{12} + s_{12} + 5x_{22} + 5y_{22} + s_{22} \geq 84 + 10s_{12} + 6s_{22}. \end{aligned} \quad (3\text{–}15)$$

The convex hull of the points formed by $s_{1,2}$, $s_{2,2}$ and $F_2(s_{1,2}, s_{2,2})$ is a plane, but the convex hull of the points formed by $s_{1,3}$, $s_{2,3}$ and $F_3(s_{1,3}, s_{2,3})$ is three-dimensional as illustrated in Figure 3-2. Let $(s_{1,3}, s_{2,3}, F_2(s_{1,2}, s_{2,2}))$ represent the points of this convex hull. Then the facet formed by points $(1, 2, 135)$,$(2, 0, 121)$ and $(2, 1, 128)$ as part of the lower convex hull for $t = 3$ (3–11) is:

$$\sum_{i=1}^{2}\sum_{t=1}^{3}(p_{it}x_{it} + f_{it}y_{it} + h_{it}s_{it}) \geq -s_{13} + 6s_{23} + 121, \quad (3\text{–}16)$$

and the facet formed by points $(1, 2, 135)$,$(2, 0, 121)$ and $(1, 0, 112)$ as part of the lower convex hull for $t = 3$ (3–11) is:

$$\sum_{i=1}^{2}\sum_{t=1}^{3}(p_{it}x_{it} + f_{it}y_{it} + h_{it}s_{it}) \geq 16s_{13} + 21s_{23} + 206, \quad (3\text{–}17)$$

The facet formed by nodes $(2, 1, 128)$,$(2, 0, 121)$ and $(0, 0, 100)$ as part of the lower concave hull for $t = 3$ (3–12) is:

$$\sum_{i=1}^{2}\sum_{t=1}^{3}(p_{it}x_{it} + f_{it}y_{it} + h_{it}s_{it}) \leq 19s_{13} + 12s_{23} + 200, \quad (3\text{–}18)$$

The Gift Wrapping algorithm can be applied to points lying in any dimensional space, so we can use this algorithm to generate convex hull inequalities for multi-item instances with three or more items. However, a dynamic programming algorithm typically consumes prohibitively large computational resources for $|K| \geq 3$. Therefore, we obtain inequalities of the form (3–11) only when $|K| \leq 2$. For $|K| \geq 3$, we do not attempt to explore the

DP state space, but instead generate inequalities of the form (3–7) by utilizing integer programming techniques.

3.3 Lifting and Separation

In this section we analyze techniques for tightening and generating the single-item inequalities introduced in the previous section. In Section 3.3.1, we provide lifting and back-lifting algorithms for these inequalities. We then describe a separation algorithm in Section 3.3.2 in which we orient the objective function so that the new partial objective inequality cuts off the fractional optimal solution obtained from a relaxation of MCLSP.

3.3.1 Lifting

In this section we study several techniques for strengthening the partial objective inequalities. In particular, we investigate back-lifting techniques (Easton et al. (2003)) to improve the coefficients of the inequalities (3–2), (3–3) and (3–4). Lifting on binary variables has proven to be useful for solving 0-1 integer programs by branch-and-cut algorithms (see Balas (1975); Crowder et al. (1983); Wolsey (1975) among others). In order to extend the ideas for 0-1 integer programming to the mixed integer programming case, it is essential to study the lifting of continuous variables. While lifting of the binary variables has been widely studied, there are fewer studies dealing with lifting of the continuous variables (see de Farias et al. (2000, 2002); de Farias and Nemhauser (2001); Easton et al. (2003); Richard et al. (2002) among others). In this study we analyze exact and approximate approaches for lifting both binary and continuous variables in the partial objective inequality (3–4) and the convex envelope inequalities (3–3) and (3–11).

3.3.1.1 Back-lifting binary variables

For a given item i and period t' we back-lift $y_{it'}$ in the partial objective inequality (3–4) to obtain a valid inequality of the form:

$$a_{y_{it'}} y_{it'} + \sum_{j \in \{1,\ldots,t\}-t'} f_{ij} y_{ij} + \sum_{j \in \{1,\ldots,t\}} (p_{ij} x_{ij} + h_{ij} s_{ij}) \geq L. \tag{3–19}$$

To provide the strongest possible inequality of the form (3–19), we minimize $a_{y_{it'}}$ such that:

$$a_{y_{it'}} y_{it'} \geq L - \left(\sum_{j \in \{1,\ldots,t\}-t'} f_{ij} y_{ij} + \sum_{j \in \{1,\ldots,t\}} (p_{ij} x_{ij} + h_{ij} s_{ij}) \right) \tag{3–20}$$

for all feasible solutions to the MCLSP. If $y_{it'} = 0$, then $a_{y_{it'}}$ can be set to any arbitrary value. Else, if $y_{it'} = 1$, then $a_{y_{it'}}$ must be at least as large as the right-hand-side of (3–20) for any (s, x, y) solution with $y_{it'} = 1$. This leads us to solving an integer program with an objective function through period t excluding item i subject to the original set of constraints including all items and all periods and the constraint forcing $y_{it'}$ to be 1. This integer program is given as follows:

$$z = \min \quad \sum_{j \in \{1,\ldots,t\}-t'} f_{ij} y_{ij} + \sum_{j \in \{1,\ldots,t\}} (p_{ij} x_{ij} + h_{ij} s_{ij}) \tag{3–21a}$$

$$\text{s.t.} \quad s_{l,j-1} + x_{lj} - d_{lj} = s_{lj} \qquad j = 1, \ldots, T,\ l = 1, \ldots, M \tag{3–21b}$$

$$x_{lj} \leq \min\{c_j, \sum_{r=j}^{T} d_{lr}\} y_{lj} \qquad j = 1, \ldots, T,\ l = 1, \ldots, M \tag{3–21c}$$

$$\sum_{l=1}^{M} x_{lj} \leq c_j \qquad j = 1, \ldots, T \tag{3–21d}$$

$$s_{lj}, x_{lj} \geq 0 \qquad j = 1, \ldots, T,\ l = 1, \ldots, M \tag{3–21e}$$

$$y_{lj} \in \{0, 1\} \qquad j = 1, \ldots, T,\ l = 1, \ldots, M. \tag{3–21f}$$

$$y_{it'} = 1. \tag{3–21g}$$

After solving (3–21), we set $a_{it'} = L - z$ if $a_{it'} < f_{it'}$.

The lifting problem (3–21) is NP-hard. If t is sufficiently small, we can solve (3–21) quickly with commercial MIP solvers. For relatively larger values of t, we provide an alternative technique for solving (3–21) via a cutting-plane approach that we describe below.

We begin by setting $k = 1$ and initializing a lower bound 0 and an upper bound to ∞. We then solve a mixed integer relaxation of (3–21), in which we ignore the shared capacity

constraint (3–21d). We call this the master problem and formulate it as:

$$\min \quad z \tag{3–22a}$$

$$z \geq \sum_{j\in\{1,\ldots,t\}-t'} f_{ij}y_{ij} + \sum_{j\in\{1,\ldots,t\}} (p_{ij}x_{ij} + h_{ij}s_{ij}) \tag{3–22b}$$

$$\text{Constraints (3–21b), (3–21c), (3–21e) – (3–21g).} \tag{3–22c}$$

Let $\left(s^k, x^k, y^k, z^k\right)$ represent an optimal solution to (3–22). We set the lower bound equal to z^k. Next, we construct a subproblem by fixing the y-variables in the original problem (3–1) to y^k. We next solve this subproblem, which is of the form:

$$\min \quad \sum_{j=1}^{t} (p_{ij}x_{ij} + h_{ij}s_{ij}) \tag{3–23a}$$

$$\text{s.t.} \quad s_{l,j-1} + x_{lj} - d_{lj} = s_{lj} \qquad j = 1,\ldots,T,\ l = 1,\ldots,M \tag{3–23b}$$

$$x_{lj} \leq \min\{c_j, \sum_{r=j}^{T} d_{lr}\} y_{lj}^k \qquad j = 1,\ldots,T,\ l = 1,\ldots,M \tag{3–23c}$$

$$\sum_{l=1}^{M} x_{lj} \leq c_j \qquad j = 1,\ldots,T \tag{3–23d}$$

$$s_{lj}, x_{lj} \geq 0 \qquad j = 1,\ldots,T,\ l = 1,\ldots,M. \tag{3–23e}$$

Associate dual variables α_{lj} with (3–23b), $-\beta_{lj}$ with (3–23c) and $-\gamma_j$ with (3–23d). Define $\delta_{lj} = 1$ if $l = i$ and $j \leq t$, and 0 otherwise. Then we obtain the following dual formulation:

$$\max \quad \sum_{l=1}^{M}\sum_{j=1}^{T} \left(d_{lj}\alpha_{lj} - \min\{c_j, \sum_{r=j}^{T} d_{lr}\} y_{lj}^k \beta_{lj} \right) - \sum_{j=1}^{T} c_j\gamma_j \tag{3–24a}$$

$$\text{s.t.} \quad \alpha_{lj} - \beta_{lj} - \gamma_j \leq \delta_{lj}p_{lj} \quad j = 1,\ldots,T, \tag{3–24b}$$

$$\alpha_{l,j+1} - \alpha_{lj} \leq \delta_{lj}h_{lj} \quad j = 1,\ldots,T-1, \tag{3–24c}$$

$$-\alpha_{lT} \leq \delta_{lT}h_{lT} \tag{3–24d}$$

$$\alpha_{lj} \quad \text{free} \qquad j = 1,\ldots,T,\ l = 1,\ldots,M \tag{3–24e}$$

$$\beta_{lj} \geq 0,\ \gamma_{lj} \geq 0 \qquad j = 1,\ldots,T,\ l = 1,\ldots,M. \tag{3–24f}$$

Note that if (3–23) is infeasible, then (3–24) must be unbounded since the trivial all-zero solution guarantees the feasibility of the dual formulation (3–24). Letting $(\alpha^*, \beta^*, \gamma^*)$ be an unbounded dual ray to (3–24), we get the following feasibility cut:

$$\sum_{l=1}^{M}\sum_{j=1}^{T} d_{lj}\alpha^*_{lj} - \sum_{j=1}^{T} c_j\gamma^*_j - \sum_{l=1}^{M}\sum_{j=1}^{T} \min\{c_t, \sum_{r=j}^{T} d_{lr}\} y^k_{lj}\beta^*_{lj} y_{lj} \leq 0. \tag{3–25}$$

Let $A = \sum_{l=1}^{M}\sum_{r=1}^{T} d_{lr}\alpha^*_{lj} - \sum_{j=1}^{T} c_j\gamma^*_j$. Then we can further improve the inequality (3–25) by:

$$\min\left\{A, \sum_{l=1}^{M}\sum_{j=1}^{T} \min\{c_j, \sum_{r=j}^{T} d_{lr}\} y^k_{lj}\beta^*_{lj}\right\} y_{lj} \geq A, \tag{3–26}$$

since $\min\{c_j, \sum_{r=j}^{T} d_{lr}\} y^k_{lj}\beta^*_{lj} \geq 0$ for all l and j.

Otherwise, if (3–23) has an optimal solution, then so does (3–24). Let $(\alpha^*, \beta^*, \gamma^*)$ be an optimal solution with objective z^k_{SP}. Set the upper bound equal to z^k_{SP} if $z^k_{SP} <$ upper bound. If $z^k = z^k_{SP}$, then stop with an optimal solution $\left(s^k, x^k, y^k\right)$. Else $z^k < z^k_{SP}$, and we add the following optimality cut to the master problem:

$$\sum_{l=1}^{M}\sum_{j=1}^{T} d_{ij}\alpha^*_{ij} - \sum_{j=1}^{T} c_j\gamma^*_j - \sum_{l=1}^{M}\sum_{j=1}^{T} \min\{c_j, \sum_{r=j}^{T} d_{lr}\} y^k_{lj}\beta^*_{lj} y_{lj} \leq z^k_{SP}. \tag{3–27}$$

If either (3–26) or (3–27) was added to the master problem, we increment k by one and re-solve the master problem.

3.3.1.2 Back-lifting continuous variables

Back-lifting with continuous variables is more complicated than back-lifting with binary variables since continuous back-lifting requires the solution of nonlinear integer problems rather than linear integer problems. Since the back-lifting procedures we discuss are identical for both s- and x-variables, we present our back-lifting procedures for s-variables only in this discussion. Given item i and period t', consider lifting the inventory variable $s_{it'}$ in (3–4). For convenience, we define

$H_{t,t'}(s,x,y) = \left(\sum_{j\in\{1,\dots,t\}-t'} h_{ij}s_{ij} + \sum_{j\in\{1,\dots,t\}} (p_{ij}x_{ij} + f_{ij}y_{ij})\right)$. Then the optimal lifting coefficient corresponding to variable $s_{it'}$ can be computed by:

$$a^*_{s_{it'}} = \max\left\{\frac{L - H_{t,t'}(s,x,y)}{s_{it'}} : \text{ Constraints (3–21b)} - \text{(3–21f)}, \ s_{it'} \geq \max\{1, LB_{it}\}\right\}. \tag{3–28}$$

Since solving for $a^*_{s_{it'}}$ requires the solution of a nonlinear mixed integer problem given by (3–28), we consider the continuous relaxation of this problem to estimate the lifting coefficient while guaranteeing its validity. We discuss three different techniques to compute $a^*_{s_{it'}}$ in this manner.

First suppose that we substitute $\lambda = \frac{L - H_{t,t'}(s,x,y)}{s_{it'}}$ and relax the integrality constraints in (3–28). We obtain the following nonlinear program:

$$\begin{aligned}
\max \quad & \lambda && \text{(3–29a)}\\
\text{s.t.} \quad & L - H_{t,t'}(s,x,y) \geq \lambda s_{it'} && \text{(3–29b)}\\
& \text{Constraints (3–21b)} - \text{(3–21e)} && \text{(3–29c)}\\
& 0 \leq y_{lj} \leq 1 \qquad j = 1,\dots,t, \ l = 1,\dots,M && \text{(3–29d)}\\
& s_{it'} \geq \max\{1, LB_{it}\}. && \text{(3–29e)}
\end{aligned}$$

To obtain the approximate lifting coefficient of $s_{it'}$, we solve problem (3–29) by fixing λ as a parameter and use the bisection method to determine candidate values of λ and then solve the feasibility problem given by (3–29). If we find a feasible solution to (3–29), we determine the largest value of λ allowed by the solution obtained and let this largest value be our new lower bound on λ. If a feasible solution is not found, then we set the upper bound to λ, and repeat the bisection method until the difference between the upper and lower bound on λ is at most ϵ, where $\epsilon > 0$ is some prespecified parameter. Pseudocode for this algorithm is given in Algorithm 2.

Algorithm 2 has complexity $O(\log_2\left(\frac{u-l}{\epsilon}\right)Q)$, where Q is the complexity of solving the LP (3–29). Since Q is a polynomial function of the input for (3–29), Algorithm 2 is polynomial in complexity.

Algorithm 2: Binary Search for Computing Lifting Coefficient for s-Variables

Input: Interval $[l, u]$ that contains optimal λ

while $u - l \leq \epsilon$ **do**

Solve the feasibility problem for $\lambda = \frac{u+l}{2}$ subject to constraints (3–29)

if (3–29) is feasible **then**

set $l = \frac{L - H_{t,t'}(s^*, x^*, y^*)}{s^*_{it'}}$ where (s^*, x^*, y^*) represents a feasible solution to (3–29) for a given λ

else

set $u = \lambda$

end if

end

Set $a_{s_{it'}} = l$

As a second alternative, we consider the following problem:

$$G(\lambda) = \max\{L - H_{t,t'}(s, x, y) - \lambda s_{it'} : \ (3\text{–}29c) - (3\text{–}29e)\}$$

It is known (Dinkelbach (1967); Ibaraki (1983)) that $G(\lambda)$ is convex, continuous and decreasing monotonically (i.e., $G(\lambda)'' < G(\lambda)'$, if $\lambda' < \lambda''$) over $\lambda \in R$. Also, there exists a unique λ^* such that $G(\lambda^*) = 0$ and:

$$\lambda^* = \frac{L - H_{t,t'}(s, x, y)}{s^*_{it'}} = \max\left\{\frac{L - H_{t,t'}(s, x, y)}{s^*_{it'}} : \ (3\text{–}29c) - (3\text{–}29e)\right\}$$

if and only if:

$$G(\lambda^*) = G(\lambda^*, s^*, x^*, y^*) = \max\{L - H_{t,t'}(s, x, y) - \lambda^* s^*_{it'} : \ (3\text{–}29c) - (3\text{–}29e)\} = 0.$$

Based on these properties, we provide a modified version of Algorithm 2. Given an interval $[l, u]$ that contains the optimal λ, we search for a value λ^* such that $G(\lambda^*) = 0$.

We select l and u such that $G(l) > 0$ and $G(u) < 0$. Let (s^n, x^n, y^n) optimize $G(n)$ for $n = l$ and u. If we cannot select a u value with $G(u) < 0$, then the lifting coefficient $h_{it'}$ cannot be improved.

After determining the interval for λ, we utilize a binary search algorithm similar to Algorithm 2. We modify this algorithm by improving lower and upper bounds in each step using the convexity of $G(\lambda)$. Consider the points $(l, G(l))$ and $(u, G(u))$ on the λ and $G(\lambda)$ axes, and note that the line connecting these points crosses the λ-axis at $u_{new} = l + (u - l)G(l)/(G(u) - G(l))$. Since $G(l) > 0$, $G(u) < 0$ and $G(\lambda)$ is convex, λ^* cannot be greater than u_{new}. Also, letting $G'(\lambda)$ be the derivative of $G(\lambda)$ at λ, consider the tangent lines passing through l and u having slopes $G'(l)$ and $G'(u)$, respectively. The tangent line passing through $G(\lambda)$ at l intersects the λ-axis at $\hat{l} = (L - H_{t,t'}(s^l, x^l, y^l))/s^l_{it'}$ and the tangent line passing through $G(\lambda)$ at u intersects the λ-axis at $\hat{u} = (L - H_{t,t'}(s^u, x^u, y^u))/s^u_{it'}$. Since $G(\lambda)$ is convex, $\max\{\hat{l}, \hat{u}\}$ cannot be greater than λ^*. Thus we can improve the lower bound l to $\max\{\hat{l}, \hat{u}\}$. We then update l and u as described above, set $\lambda = (u + l)/2$ and check whether $G(\lambda) = 0$. If $G(\lambda) = 0$, then λ is optimal. If $G(\lambda) > 0$, we set $l = \lambda$, and otherwise if $G(\lambda) < 0$, then $u = \lambda$. We iterate in this fashion until the difference between u and l is at most a predetermined value $\epsilon > 0$. This procedure is given in Algorithm 3.

A third approach to computing $a_{s_{it'}}$ is to directly consider the lifting problem given by (3–28), with y-variables relaxed to be continuous and bounded between 0 and 1. Let $\psi \geq 1$ be a feasible parameter lower bound for $s_{it'}$, and define $Z(\psi)$ as:

$$Z(\psi) = \max\{L - H_{t,t'}(s, x, y) : \ (3\text{–}29c) - (3\text{–}29d),\ s_{it'} = \psi\}, \tag{3–30}$$

where $Z(\psi) = -\infty$ if (3–30) is infeasible for ψ. We seek a value of ψ that maximizes $Z(\psi)/\psi$. Note that $Z(\psi)$ is a piecewise-linear concave function, since ψ only appears in the right hand side of the constraints defining the feasible region of the linear program (3–30). To maximize $Z(\psi)/\psi$, we seek the last piecewise linear segment of $Z(\psi)$ (from

left to right) whose slope is at least 1. The right endpoint of this segment gives the optimal ψ. To see this, note that if ψ is located on a segment to the left of its right endpoint, and the slope of this segment is greater than 1, then increasing ψ at a rate of 1 causes the numerator $Z(\psi)$ to grow at a rate of more than 1, thus increasing $Z(\psi)/\psi$. Therefore we increase ψ until we reach a breakpoint such that the slope of the segment to the right of the breakpoint is 1 or smaller.

Algorithm 3: Lifting using Parametric Programming with Improved Bounds

Input: Interval $[l, u]$ that contains the optimal λ

while $u - l \leq \epsilon$ **do**

For $\lambda = l$ and u, compute $G(\lambda)$ and obtain an optimal solution $\left(s^\lambda, x^\lambda, y^\lambda\right)$

$u_{new} = l + (u - l)\frac{G(l)}{G(u)-G(l)}$

Set l=max $\left\{\frac{L-H_{t,t'}(s^l,x^l,y^l)}{s^l_{it'}}, \frac{L-H_{t,t'}(s^u,x^u,y^u)}{s^u_{it'}}\right\}$

Set $u = u_{new}$

Solve $G(\lambda)$ for $\lambda = (u + l)/2$ having optimal solution $\left(s^\lambda, x^\lambda, y^\lambda\right)$

if $G(\lambda) = 0$ **then**

set $a_{s_{it'}} = \lambda$ and stop

else if $G(\lambda) > 0$ **then**

set $l = \lambda$, $G(l) = G(\lambda)$ and $\left(s^l, x^l, y^l\right) = \left(s^\lambda, x^\lambda, y^\lambda\right)$

else

set $u = \lambda$, $G(u) = G(\lambda)$ and $(s^u, x^u, y^u) = \left(s^\lambda, x^\lambda, y^\lambda\right)$

end if

end

Set $a_{s_{it'}} = \lambda$

Given a feasible interval $[l, u]$ that contains an optimal ψ^*, our algorithm performs a bisection method to compute ψ^*. We begin by solving $Z(l)$ and $Z(u)$, and use sensitivity analysis to determine (a) the rightmost point $\bar{l}$ such that the same basis optimizes (3–30) for both $\psi = l$ and $\psi = \bar{l}$, and (b) the leftmost point $\bar{u}$ such that the same basis optimizes (3–30) for both $\psi = u$ and $\psi = \bar{u}$. If $\bar{l} \geq \bar{u}$ or $\left[G(\bar{l}) - G(l)\right]/(\bar{l}-l) = 1$, then $\psi^* = \bar{l}$. Else, if

$[G(u) - G(\bar{u})]/(u - \bar{u}) = 1$, then $\psi^* = \bar{u}$. Otherwise we compute $m = (u + l)/2$ and re-solve the problem, and we perform sensitivity analysis to find the allowable range $[l_m, u_m]$ for which the optimal basis corresponding to the optimal solution for (3–30) given $\psi = m$ does not change. Then we compute the slope given by $[G(u_m) - G(l_m)]/(u_m - l_m)$. If the slope is 1, then the algorithm terminates with $\psi = m$. If the slope is larger than 1, then we set $l = l_m$, while if it is smaller than 1, then we set $r = u_m$. We continue by repeating the algorithm until we reach an optimal ψ-value. Note that this algorithm finitely converges to an optimal solution, in contrast to Algorithms 2 and 3. This procedure is presented in Algorithm 4.

Algorithm 4: Lifting with Sensitivity Analysis

Input: Interval $[l, u]$ that contains an optimal ψ^*
while $l \leq u$ **do**
  Compute $Z(l)$ and $Z(u)$ and use sensitivity analysis to determine
  (1) the rightmost point $\bar{l}$ such that the same basis optimizes (3–30) for both $\psi = l$ and $\bar{l}$;
  (2) the leftmost point $\bar{u}$ such that the same basis optimizes (3–30) for both $\psi = u$ and $\bar{u}$.
  if $\bar{l} \geq \bar{u}$ or $\left[G(\bar{l}) - G(l)\right]/(\bar{l} - l) = 1$ **then**
    set $\psi^* = \bar{l}$ and stop.
  else if $[G(u) - G(\bar{u})]/(u - \bar{u}) = 1$ **then**
    set $\psi^* = \bar{u}$ and stop.
  else
    Compute $m = (r + l)/2$ and $Z(m)$, and use sensitivity analysis to determine the allowable range $[l_m, u_m]$ for which the optimal basis corresponding to the optimal solution for (3–30) given $\psi = m$ does not change.
    if $[G(u_m) - G(l_m)]/(u_m - l_m) = 1$ **then**
      set $\psi^* = m$ and stop.
    else if $[G(u_m) - G(l_m)]/(u_m - l_m) < 1$ **then**
      set $u = l_m$.
    else if $[G(u_m) - G(l_m)]/(u_m - l_m) > 1$ **then**
      set $l = u_m$.
    end if
  end if
end

Example 3. To illustrate our lifting techniques and the strength of the inequalities generated after lifting, consider an instance of MCLSP with $T = 3$, $M = 2$, $c_t =$

$(5,4,4)$, $d_{1t}=(1,2,2)$, $d_{2t}=(2,1,3)$, $s_{1t}=(2033,2198,1926)$, $s_{2t}=(2101,1823,2062)$, $p_{1t}=(109,99,83)$, $p_{2t}=(106,113,103)$ and $h_{it}=10$ for $i=1,2$. The dimension of the polyhedron corresponding to the example problem is found to be 10 by using PORTA.[1]

For $i=1$ the inequalities corresponding to (3–4) for $t=1,2,3$ are:

$$2033y_{11}+109x_{11}+10s_{11}\geq 2142, \tag{3–31}$$

$$\sum_{t=1}^{2}(f_{1t}y_{1t}+p_{1t}x_{1t}+h_{1t}s_{1t})\geq 2380, \tag{3–32}$$

$$\sum_{t=1}^{3}(f_{1t}y_{1t}+p_{1t}x_{1t}+h_{1t}s_{1t})\geq 4472, \tag{3–33}$$

respectively. The dimensions of the faces of the MCLSP defined by the inequalities (3–31), (3–32) and (3–33) are 8, 4 and 3, respectively.

After back-lifting s_{1t} in (3–31), we obtain:

$$2033y_{1t}+109x_{1t}-109s_{1t}\geq 2142. \tag{3–34}$$

After back-lifting y_{12} first and then s_{12} in (3–32), we obtain:

$$f_{11}y_{11}+p_{11}x_{11}+h_{11}s_{11}+(f_{12}-2158)y_{12}+p_{12}x_{12}+(h_{12}-109)s_{12}\geq 2380. \tag{3–35}$$

If we back-lift x_{12} in (3–32), we obtain:

$$f_{11}y_{11}+p_{11}x_{11}+h_{11}s_{11}+f_{12}y_{12}+(p_{12}-594)x_{12}+h_{12}s_{12}\geq 2380. \tag{3–36}$$

After back-lifting y_{12} then s_{13}, in (3–33), we obtain:

$$\begin{aligned} &f_{11}y_{11}+p_{11}x_{11}+h_{11}s_{11}+(f_{12}-284)y_{12}+p_{12}x_{12}+h_{12}s_{12}+\\ &f_{13}y_{13}+p_{13}x_{13}+(h_{13}-93)s_{13}\geq 4472. \end{aligned} \tag{3–37}$$

[1] POlyhedron Representation Transformation Algorithm (by Thomas Christof, Heidelberg Univ.). Routine collection to analyze polytopes and polyhedra. http://www.zib.de/Optimization/Software/Porta.

The dimensions of the polyhedral faces induced by inequalities (3–34), (3–35), (3–36) and (3–37) are 10, 9, 8 and 7, respectively.

Back-lifting inequalities (3–31) and (3–32) results in facet-defining inequalities for the MCLSP instance, and back-lifting inequality (3–33) considerably increases the dimension of the polyhedral face it induces.

The LP relaxation value for (3–1) for this example instance is 5587, while the optimal objective to (3–1) is 9919. All the inequalities (3–31)–(3–37) cut off the optimal LP relaxation solution. By adding lifted inequalities (3–34), (3–35), and (3–37) to (3–1), LP relaxation of (3–1) improves to 7747.33, a 28% improvement.

3.3.1.3 Back-lifting by dynamic programming

For any $t = 1, \ldots, T$, given item i, we obtain the inequality (3–2) by allotting the maximum capacity in each period to item i such that ample capacity remains to satisfy demand for the other items, and solving the single-item DP through period t. Now, given item i and period t' with $t' \leq t$, consider back-lifting the inventory variable $s_{it'}$ in the valid inequality (3–2). Then as we have shown, we back-lift variable $s_{it'}$ by computing:

$$a^*_{s_{it'}} = \max \left\{ \frac{L - H_{t,t'}(s,x,y)}{s_{it'}} : \ (3\text{–}21b) - (3\text{–}21f),\ s_{it'} \geq \max\{1, LB_{it}\} \right\}. \tag{3–38}$$

The only difference between problems (3–28) and (3–38) is that we relax our constraint set to a single item in (3–38). Therefore to compute $a^*_{s_{it'}}$, we need to solve a dynamic program up to period t with updated costs in order to set the inventory cost of item i in period t' to 0, while keeping all the other costs the same. Then, by backtracking from period t to period t', we can compute the minimum cost values at each state at period t and the corresponding state values in period t'. After obtaining the state values in period t' with their minimum corresponding costs in period t, we can compute the objective function given in (3–38) by enumerating all feasible $s_{it'} \geq 1$ and corresponding $H_{t,t'}(s,x,y)$ values and then computing the optimal lifting coefficient $a^*_{s_{it'}}$ to back-lift the variable $s_{it'}$ in (3–2).

Remark 1. It is interesting to note the correspondence between an inequality produced by back-lifting $s_{t'}$ to (3–2) and the first (left-most) convex envelope inequality of the form (3–3). For a given item i and period t' we back-lift $s_{t'}$ to the partial objective inequality (3–4) to obtain a valid inequality of the form:

$$\sum_{j\in\{1,\dots,t\}} (f_j y_j + p_j x_j + h_j s_j) \geq L + \tilde{a}_{s_t} s_t. \tag{3–39}$$

To provide the strongest possible inequality of the form (3–19), we maximize a_{s_t} such that:

$$\tilde{a}_{s_t} s_t \leq \sum_{j\in\{1,\dots,t\}} (f_j y_j + p_j x_j + h_j s_j) - L. \tag{3–40}$$

This leads us to solving the following mixed integer non-linear program:

$$\tilde{a}^*_{s_t} = \min\left\{ \frac{\left(\sum_{j\in\{1,\dots,t\}} (f_j y_j + p_j x_j + h_j s_j)\right) - L}{s_t} : \quad (3\text{–}1b) - (3\text{–}1f),\ s_t \geq \max\{1, LB_t\} \right\}. \tag{3–41}$$

Define:

$$F(s^k_t) = \min \quad \sum_{j\in\{1,\dots,t\}} (f_j y_j + p_j x_j + h_j s_j) \tag{3–42a}$$

$$\text{s.t.} \quad s_{j-1} + x_j - d_j = s_j \qquad j = 1,\dots,T, \tag{3–42b}$$

$$x_j \leq c_j y_j \qquad j = 1,\dots,T, \tag{3–42c}$$

$$s_j, x_j \geq 0 \qquad j = 1,\dots,T, \tag{3–42d}$$

$$y_j \in \{0,1\} \qquad j = 1,\dots,T, \tag{3–42e}$$

$$s_t = s^k_t. \tag{3–42f}$$

Then we can write (3–41) as follows:

$$\tilde{a}^*_{s_t} = \min_{s^k_t \geq \max\{1, LB_t\}} \left\{ \frac{(F(s^k_t) - L)}{s^k_t} \right\}. \tag{3–43}$$

In fact the equation (3–41) gives the minimum slope in our stage t value function for item i when we compute the first convex envelope inequality (3–3). Therefore after

back-lifting $s_{it'}$ in constraint (3–2), we obtain an equivalent cut to the first convex envelope inequality (3–3).

3.3.1.4 Forward-lifting binary variables

Forward-lifting is performed in a similar manner as the back-lifting techniques. For a given item i and period $t' > t$ we lift $y_{it'}$ in the partial objective inequality (3–4) to obtain a valid inequality of the form:

$$\sum_{j\in\{1,\dots,t\}} (f_j y_j + p_j x_j + h_j s_j) \geq L + \tilde{a}_{y_{it'}} y_{it'}. \tag{3–44}$$

To provide the strongest possible inequality of the form (3–44), we maximize $a_{y_{it'}}$ such that:

$$\tilde{a}_{y_{it'}} y_{it'} \leq \sum_{j\in\{1,\dots,t\}} (f_j y_j + p_j x_j + h_j s_j) - L. \tag{3–45}$$

This leads us to solving the following mixed integer linear program:

$$\tilde{a}^*_{y_{it'}} = \min \left\{ \sum_{j\in\{1,\dots,t\}} (f_j y_j + p_j x_j + h_j s_j) - L : \quad (3\text{–}1b) - (3\text{–}1f),\ y_{it'} = 1 \right\}. \tag{3–46}$$

3.3.2 Separation Algorithm

We begin by noting that all of the foregoing inequalities remain valid if we modify the objective function coefficients for use in constraint generation, while retaining the original objective coefficients in (3–1a). In our separation algorithm we begin by seeking partial objective inequalities (3–4) over a subset K of items that are violated by the given fractional solution. We also revise the objective coefficients used to generate the inequalities to obtain partial objective inequalities (3–4) with different orientation. Our heuristic separation algorithm is described below.

Initialization: Begin by setting $t = 1$. Select $\acute{t} \geq 1$ as a parameter that represents the period up to which we generate our cutting planes, and construct problem LP^t exactly as the linear programming relaxation of (3–1).

Step 1. If $t > \acute{t}$, then stop. Else, set $f'_{ij} = f_{ij}$ for all i and j. Solve LP^t and obtain optimal solution $(\hat{s}^t, \hat{x}^t, \hat{y}^t)$ with objective value z^t_{LP}. Set $\kappa = 0$.

Step 2. Consider problem SUB, which is the same as LP^t but with an objective of:

$$\min \quad \sum_{i \in K} \sum_{j=1}^{t} \left(p_{ij} x_{ij} + f'_{ij} y_{ij} + h_{ij} s_{ij} \right). \tag{3–47}$$

Let an optimal solution to SUB^t be $(\tilde{s}^t, \tilde{x}^t, \tilde{y}^t)$ with objective z^t_{SUB}. We then test whether:

$$\frac{\sum_{i \in K} \sum_{j=1}^{t} \left(p_{ij} \hat{x}_{ij} + f'_{ij} \hat{y}_{ij} + h_{ij} \hat{s}_{ij} \right)}{z^t_{SUB}} \leq \epsilon, \tag{3–48}$$

where $\epsilon \geq 1$ is a predetermined parameter. Note that as the ratio given in the left-hand side of (3–48) gets smaller, the lower bound z^t_{SUB} on the partial objective function given in (3–47) gets tighter and the potential of the new lower bound z^t_{MIP} obtained after solving the MIP version of SUB to violate $(\hat{s}^t, \hat{x}^t, \hat{y}^t)$ increases. If (3–48) holds true, then go to Step 3. Else go to Step 5.

Step 3. Solve the MIP version of SUB^t to compute a valid lower bound $z^t_{MIP} \geq z^t_{SUB}$ and obtain the following inequality:

$$\sum_{i \in K} \sum_{j=1}^{t} \left(f'_{ij} y_{ij} + p_{ij} x_{ij} + h_{ij} s_{ij} \right) \geq z^t_{MIP}. \tag{3–49}$$

Proceed to Step 4.

Step 4. If the inequality (3–49) is violated by $(\hat{s}^t, \hat{x}^t, \hat{y}^t)$, then add (3–49) to both formulations LP^t and SUB^t, re-solve LP^t and update the solution $(\hat{s}^t, \hat{x}^t, \hat{y}^t)$. In either case, proceed to Step 5.

Step 5. If $\kappa \geq 1$, then increment t by one and go to Step 1. Otherwise proceed to Step 6.

Step 6. Compute lower and upper objective sensitivity values sl^t_{ij} and su^t_{ij} for each y^t_{ij}, such that $(\tilde{s}^t, \tilde{x}^t, \tilde{y}^t)$ remains optimal if $f'_{ij} \in [sl^t_{ij}, su^t_{ij}]$ and all other objective coefficients are unchanged. Let $\{g_1, g_2, \ldots, g_J\} = \left\{(i,j) : \ i \in K, \ 1 \leq j \leq t, \ 0 < \tilde{y}^t_{ij} < 1\right\}$

indexed in nondecreasing order of j, breaking ties in increasing order of i. Set $\rho = 1$ and go to Step 7.

Step 7. If neither sl_{ig_ρ} nor su_{ig_ρ} is finite, then go to Step 8. Else, if su_{ig_ρ} is finite then set $f'_{ig_\rho} = su_{ig_\rho} + \Delta$; else, set $f'_{ig_\rho} = sl_{ig_\rho} - \Delta$, where Δ represents a perturbation value that we choose. Increment κ by one and go to Step 2.

Step 8. If $\rho < J$, then increment ρ by 1 and go to Step 7. Otherwise increment t by one and go to Step 1.

3.4 Computational Results

Here we present our computational results for testing the effectiveness of the convex hull inequalities (3–11) and (3–12) in solving randomly generated MCLSP instances. For the computational experiments we solve all mathematical programming problems using CPLEX 11.0. An 1800 CPU-second time limit is imposed for all test instances.

The data used in the first set of experiment instances are generated using a similar scheme as in Atamtürk and Muñoz (2004). We generate data as follows: Demands are generated from an integer uniform distribution between 1 and 5 for each item. We fix the number of stages at $T = 60$. To observe the effect of different capacities and cost parameters on the computations, we vary a capacity multiplier parameter $\omega \in \{2.5, 3\}$ and set-up to holding cost ratios $\theta \in \{100, 200, 500\}$, and generate five random instances for each combination for a total of 30 instances. The shared capacities c_t are generated from an integer uniform distribution between $0.75\omega\bar{d}$ and $1.25\omega\bar{d}$, where $\bar{d}$ is equal to 5 (the average demand for an item). The set-up costs s_t are randomly generated from an integer uniform distribution with range $0.9\theta\bar{h}$ and $1.1\theta\bar{h}$, where $\bar{h}$ is the average holding cost. The unit production costs p_t are generated from an integer uniform distribution between 81 and 119. The holding cost h_t is fixed at 10 for each period.

Our implementation executes the forward DP for the two item MCLSP for a limited number of stages to generate our proposed inequalities. We append these inequalities to strengthen the MCLSP formulation (3–1) and solve the resulting model using CPLEX

with its default settings. We compare the efficiency of the following solution approaches in our computational experiments.

- **base**: Formulation (3–1), without adding any user inequalities.
- **convexhull**: Inequalities (3–11) and (3–12) with base.

Table 3-1 summarizes the results of our computational experiments. In Tables 3-1 and 3-2 we report the following data by column:

- $\boldsymbol{\theta}$: Set-up to holding cost ratio.
- **exp**: Solution approach.
- **stage**: The number of forward-recursion stages up to which DP inequalities are generated.
- **DPineq**: Average number of DP based inequalities added to the original formulation.
- **CPXineq**: Average number of CPLEX inequalities added to the original formulation.
- **nodes**: Average number of nodes explored in the branch-and-bound tree.
- **ineqtime**: Average CPU time (in seconds) to generate the DP based inequalities
- **time**: Average CPU time (in seconds) to solve the instance.

Tables 3-1 and 3-2 present results for $T = 60$ and $M = 2$, where each table entry corresponds to the average performance of an algorithm over five instances (five each for $\theta = 100$, 200 and 500). In Tables 3-1 and 3-2, we also report the overall averages for the 15 instances. We observe that the convex hull inequalities (3–11) and (3–12) improve the direct solution of the MCLSP by CPLEX on average. Out of the 30 instances, two of them cannot be solved by base within 1800 CPU seconds. These instances could be solved with the implementation of the inequalities (3–11) and (3–12) with stage = 10 and 20. It is clear that as the number of stages increases, the DP solution time required to generate the convex hull inequalities increases. Therefore an efficient implementation of the DP algorithm to obtain the inequalities (3–11) and (3–12) is a crucial consideration in effectively using these inequalities.

Table 3-1. Summary of experiments for $T = 60$, $M = 2$ and $\omega = 2.5$.

θ	exp	stage	DPineq	CPXineq	nodes	ineqtime	time
100	base		0	278	238705	0.00	219.48
	convexhull	5	2887	250	259909	2.17	239.38
		10	6779	293	114589	10.89	165.04
		15	9829	295	69974	33.07	142.94
		20	12712	202	31440	75.43	86.21
200	base		0	186	584808	0.00	498.09
	convexhull	5	2516	220	611423	1.74	505.90
		10	6035	215	345403	10.12	411.96
		15	9505	150	325938	29.15	433.36
		20	12502	124	213542	69.34	376.54
500	base		0	158	452220	0.00	270.53
	convexhull	5	2417	190	157649	1.71	102.39
		10	6300	115	468041	10.37	326.89
		15	10161	157	237036	32.26	312.26
		20	13562	135	152707	71.87	215.41
Average	base		0	207	425245	0.00	329.37
	convexhull	5	2607	220	342994	1.87	282.56
		10	6371	208	309344	10.46	301.30
		15	9832	201	210983	31.49	296.19
		20	12925	154	132563	72.21	226.05

For the second set of experiments we generate data as follows: Demands are generated from an integer uniform distribution between 1 and 10 for each item. We fix the number of stages at $T = 18$ and number of items at $M = 8$. In this experiment we vary set-up to holding cost ratios $\theta \in \{100, 200, 500\}$, and generate five random instances for each combination for a total of 15 instances. The shared capacities c_t are generated from an integer uniform distribution between $0.75\omega\bar{d}$ and $1.25\omega\bar{d}$, where $\bar{d}$ is equal to 10 (the average demand for an item) and $\omega = 6$. The set-up costs s_t, p_t and h_t are generated exactly in the same way as presented in the first experiment.

Our second implementation solves (3–1) with an objective of (3–47) for five individual items using separation algorithm to generate inequalities (3–49). Each time we obtain an inequality (3–49) we apply an exact lifting of y- and s-variables using Algorithm 2, in which y-variables are restricted to binaries. We restrict our implementation to exact lifting since approximate lifting did not improve solution times. We append the lifted inequalities

Table 3-2. Summary of experiments for $T = 60$, $M = 2$ and $\omega = 3$.

θ	exp	stage	DPineq	CPXineq	nodes	ineqtime	time
100	base		0	330	96587	0.00	101.63
	convexhull	5	2574	208	291548	1.84	219.53
		10	6031	268	89157	10.98	111.70
		15	8682	254	51103	40.79	154.27
		20	11262	228	30819	103.67	196.75
200	base		0	144	957299	0.00	614.40
	convexhull	5	3061	170	668632	1.70	442.33
		10	6883	184	514018	13.13	408.87
		15	10358	212	160230	49.57	295.95
		20	13960	155	85677	128.11	246.81
500	base		0	191	481641	0.00	263.89
	convexhull	5	2787	181	442829	2.03	268.53
		10	6767	160	287881	15.53	216.57
		15	10232	109	101953	59.37	222.71
		20	14102	100	87925	145.63	284.62
Average	base		0	222	511842	0.00	326.64
	convexhull	5	2807	186	467670	1.86	310.13
		10	6560	204	297019	13.21	245.71
		15	9757	192	104429	49.91	224.31
		20	13108	161	68140	125.81	242.73

(3–49) to strengthen the MCLSP formulation and solve the resulting model using CPLEX with its default settings. We compare the efficiency of the following solution approaches in our computational experiments.

- **base**: Formulation (3–1), without adding any user inequalities.
- **seplift**: Lifted inequalities (3–49) with base.

Table 3-3 presents results for $T = 18$ and $M = 8$, where each table entry corresponds to the average performance of an algorithm over five instances (five each for $\theta = 100$, 200 and 500). We use five individual items for which we generate the inequalities (3–49) up to period 6. Here the column (objineq) defines the number of partial objective function inequalities (3–49) that are generated in seplift strategy, while (cuttime) gives the cut generation time in CPU seconds. We observe that the lifted inequalities (3–49) improve the direct solution of the MCLSP by CPLEX on average. These results show that lifting

Table 3-3. Summary of experiments for $T = 18$ and $M = 8$.

θ	exp	objineq	cpxineq	nodes	cuttime	time
100	base	-	322.71	37885	0.0	123.9
	seplift	2.9	323.86	22005	3.3	98.3
200	base	-	264.14	39407	0.0	143
	seplift	5.0	271.29	36770	1.4	115.3
300	base	-	371.57	162927	0.0	562.0
	seplift	3.6	374.71	117051	0.2	458.6
Average	base	-	319.48	80073	0.0	276.3
	seplift	3.8	323.29	58608	1.6	224.1

and separation of these inequalities is promising in terms of improving the computational solution times.

3.5 Summary

In this chapter, we study the MCLSP. We investigate decomposition techniques where we can incorporate the DP based inequalities for solving the general multi-item problem. In addition, we consider exact solution techniques for solving the first t periods of the (MCLSP) with integer programming techniques, and then strengthen the cut coefficients via back-lifting techniques. In addition we develop a separation algorithm using these cutting planes.

CHAPTER 4
PARALLEL EQUIPMENT REPLACEMENT PROBLEM UNDER ECONOMIES OF SCALE (PRES)

4.1 Introduction

The parallel replacement problem is concerned with determining minimum cost replacement schedules for each individual asset in a group of assets that operate in parallel and are economically interdependent. The replacement of assets is generally driven by economics, including increased operating and maintenance costs (O&M) of deteriorating assets and the availability of newer, more efficient assets in the marketplace. Unlike serial (single asset) replacement problems (see Fraser and Posey (1989), for example), parallel replacement problems are combinatorial as groups of assets must be analyzed simultaneously.

This chapter presents valid inequalities and a cut-and-branch solution procedure to an integer programming formulation for the deterministic, parallel replacement problem under economies of scale (PRES) in which a constant number of assets are required for operations in each period over a finite horizon of length T. At the end of each period, an asset may be salvaged or retained, assuming it has not reached its maximum physical life N, at which time it must be retired. Assets may be replaced through the purchase of new assets. The purchase of assets is subject to a fixed charge, regardless of the order size. At the end of the finite horizon, all assets are salvaged. The solution consists of purchase and salvage decisions for each asset over the finite horizon with the objective of minimizing discounted purchase and O&M costs less salvage values.

This study is motivated by the integer programming results of Hartman (2000) and the successful use of cutting planes in solving lot-sizing problems (Barany et al. (1984a)). Conceptually, the parallel replacement problem and the lot-sizing problem are similar. In the lot-sizing problem, inventory purchases are made by trading off a fixed charge (set-up cost) against inventory carry charges. In the parallel replacement problem

under economies of scale, additional fixed charges are incurred if assets are not replaced simultaneously.

The model presented is a fixed-charge minimum cost flow model. Thus, if the binary variables (required for imposing a fixed charge with asset purchases) are fixed, then the optimal solution to the linear programming relaxation of the resulting formulation is integer-valued, if a feasible solution exists. Thus, branch-and-bound procedures must only focus on the T binary variables. In this chapter, we provide valid inequalities that focus on these T variables to further reduce the difficulty of solving PRES.

The valid inequalities are useful from a computational standpoint but they are also interesting as they are derived from a consequence of the "no-splitting rule", which has been used to reduce the computation time in earlier dynamic programming approaches to the problem. The rule states that an optimal solution to PRES exists such that all assets of the same age in the same time period are either kept or replaced as a group. "No-splitting rule" leads to the fact that if an asset is salvaged in a period, then there has to be a purchase in that period. We model our problem using this property such that flow conservation constraints impose that a salvage triggers a purchase in a period. We then use this property to generate valid inequalities tightening constraints, which enforce the fixed charge in each period of an asset purchase.

This chapter makes three contributions to the replacement analysis literature. First, we prove that PRES is NP-hard. Second, a set of valid inequalities is defined for PRES. Their relationship to the result of the"no-splitting rule" stating that a purchase is enforced by a salvage is made clear in their development. Third, computational results show that the incorporation of these inequalities into a cut-and-branch procedure drastically improves the solution time of PRES. This is especially true for large problem instances, such as those from the railroad industry analyzed in Hartman and Lohmann (1997).

4.2 PRES under Constant Demand

We now formulate PRES as an integer program. Unlike Hartman (2000), we define decision variables according to the number of periods an asset is retained as opposed to defining whether an asset is kept or retained after each period. (The approaches are equivalent, but this approach significantly reduces the number of variables and is akin to single asset approaches, as in Oakford et al. (1984).) It is assumed that the number of assets in inventory at time zero is equal to demand and no asset in the initial inventory has an age equal to its maximum service life. This eliminates the automatic decision of having to replace an asset at time zero. Assuming constant demand and no capital budgeting constraints eliminates the need to store assets, as in Hartman (2000).

An asset is defined by its age, $i = 0, 1, \ldots, N$ at the end of time period $j = 0, 1, \ldots, T$. An asset may be retained or salvaged after each period unless it reaches age N at which time it must be salvaged. The problem is solved over T periods, with purchases allowed at the end of periods $0, 1, \ldots, T-1$. All assets are sold at the end of time period T. The decision variables are summarized as follows:

X_{jk} : the number of assets in use from the end of period j to the end of period k;

S_{ij} : the number of i-period old assets owned at time zero (initial inventory) and salvaged at the end of period j;

Z_j : binary variable that equals 1 if a purchase is made in period j, 0 otherwise.

The costs associated with X_{jk} include the purchase cost at time j, O&M costs over the ensuing j through k periods, minus revenue from salvage at time k. Costs associated with S_{ij} are similar and defined as r_{ij} to avoid confusion, but there is no purchase cost as this variable is concerned with assets already owned. The fixed cost k_j is incurred in any period j in which a purchase is made.

Other relevant parameters include:

n_i : the number of i-period old assets available (in inventory) at time zero;

d : the number of assets demanded in each period such that $d = \sum_i n_i$.

To aid in the discussion, the formulation may be visualized as a network, as illustrated in Figure 4-1. The initial inventory of assets owned at time zero is represented as source nodes to the network. These are visually represented as square nodes in the figure, labeled with the age of the initial assets. The value of n_i represents the amount of supply of age i assets at time zero.

An additional $T+1$ nodes represent the periods of the problem, 0 through T, where T is a sink node demanding to receive flow of $\sum_i n_i$ assets. Nodes 0 through $T-1$ represent transshipment nodes in which there is no supply or demand. Note that a supply node i is connected to nodes 0 through $N-i$, representing the decisions to salvage an asset immediately through retaining an asset until age N.

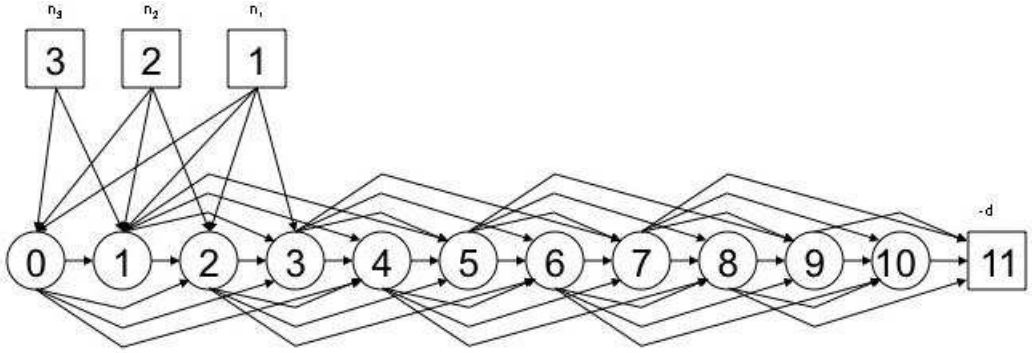

Figure 4-1. Network representation of PRES with flow representing assets in use with $N = 4$.

For a given problem, there are at most $N-1$ supply nodes, 1 demand node, and T transshipment nodes. Furthermore, there are at most $(N^2+N)/2$ arcs connecting the supply to the transshipment nodes, N arcs flowing into the demand node, and fewer than NT arcs connecting transshipment nodes.

With these variables and parameters, the integer programming formulation for PRES follows:

$$\min \quad \sum_{j=0}^{T-1} k_j Z_j + \sum_{i=0}^{T-1} \sum_{j=i+1}^{\min\{i+N,T\}} c_{ij} X_{ij} + \sum_{i=1}^{N-1} \sum_{j=0}^{N-i} r_{ij} S_{ij} \tag{4–1a}$$

$$\text{s.t.} \quad \sum_{j=0}^{N-i} S_{ij} = n_i \;\; \forall\, 1 \le i \le N-1 \tag{4–1b}$$

$$\sum_{i=1}^{N-1} S_{i0} - \sum_{k=1}^{N} X_{0k} = 0 \tag{4–1c}$$

$$\sum_{i=j-\min\{N,j-1\}}^{N-j} S_{ij} + \sum_{i=j-\min\{N,j\}}^{j-1} X_{ij} = \sum_{k=j+1}^{\min\{j+N,T\}} X_{jk} \quad \forall\, 1 \le j \le N-1 \tag{4–1d}$$

$$\sum_{i=j-N}^{j-1} X_{ij} = \sum_{k=j+1}^{\min\{j+N,T\}} X_{jk} \;\forall\, N \le j \le T-1 \tag{4–1e}$$

$$\sum_{j=T-N}^{T-1} X_{jT} = \sum_{i=1}^{N} n_i \tag{4–1f}$$

$$\sum_{j=i+1}^{\min\{i+N,T\}} X_{ij} \le dZ_i \quad \forall\, 0 \le i \le T-1 \tag{4–1g}$$

$$X_{ij}, S_{ij} \in \{0,1,2,\ldots\} \quad \forall\, 0 \le i < j \le T \tag{4–1h}$$

$$Z_j \in \{0,1\} \; 0 \le j \le T-1 \tag{4–1i}$$

With constant demand, PRES is a fixed-charge minimum cost flow problem. The objective function (4–1a) minimizes discounted purchase and O&M costs less salvage values according to life cycle costs c_{ij} and r_{ij}. Constraints (4–1b) provide $\sum_i n_i$ assets to the network in which flow is conserved through Constraints (4–1d) and (4–1e). All flow culminates at the demand node T, as defined by Constraint (4–1f).

Constraint (4–1g) includes the fixed charge variable Z_j such that if any assets are purchased, the fixed charge is imposed. As demand d is constant and the number of initial assets in the system is d, the maximum number of assets that can be purchased in any period is d. The X- and S- variables are restricted to be general integers, and the Z_j being restricted as binary. Note that constraints (4–1h) are not required as integrality is maintained if (4–1i) is enforced.

4.3 Complexity of PRES

Consider a decision version of this problem, which seeks a feasible solution with objective not more than some value G. We refer to the decision problem as DPRES. It

is not hard to show that DPRES is strongly NP-complete from, e.g., the uncapacitated facility location problem when no assumptions are made on the costs. However, this transformation requires that for some node i, $0 \leq i < T$, the set of c_{ij} values can increase or decrease as a function of j. In many realistic applications, though, c_{ij} (and r_{ij}) is nondecreasing as a function of j, since O&M costs generally increase over time, and salvage revenues generally decrease with age. Hence, we prove the stronger result below, which states that even if the c_{ij} and r_{ij} costs are nondecreasing functions of j, DPRES is NP-complete in the strong sense. This result thus implies that the optimization problem PRES is strongly NP-hard. The reduction is from 3SAT, which we define as follows.

3SAT. Consider a set of α Boolean variables, $v_1, \ldots, v_\alpha$, and a set of β disjunctive clauses $\mathcal{C}_1, \ldots, \mathcal{C}_\beta$, where each clause $\mathcal{C}_i$ contains three literals (one of the variables or its negation). The 3SAT problem seeks to assign a value of true or false to each variable, so that each clause contains at least one literal whose value is satisfied in the solution.

Theorem 1. DPRES *is NP-complete in the strong sense, even if* c_{ij} *and* r_{ij} *are nondecreasing functions of* j.

Proof. First, note that DPRES clearly belongs to NP, since a polynomial-size guess (in terms of X-, S-, and Z-variables) can be checked for feasibility and cost in polynomial time. To prove that DPRES is NP-complete, we transform any arbitrary 3SAT instance into an equivalent DPRES instance. This transformation can be visualized in Figure 4-2.

Given the α variables and β clauses from 3SAT, we set $T = (2\alpha)(\alpha + \beta)$ and $N = 2\alpha(\alpha+\beta)$. The time periods can be grouped into $\alpha+\beta+1$ "blocks" of 2α consecutive nodes, plus one extra node representing the finish at time T. Each block contains one node corresponding to each possible variable value. Indexing the nodes of a block as $0, \ldots, 2\alpha - 1$, a true value for v_i corresponds to node $2(i-1)$ and a false value for v_i corresponds to node $2(i-1)+1$, for each $i = 1, \ldots, \alpha$.

The first α blocks, which we call "variable blocks," require us to select at least one value for each variable. The next β blocks (the "clause blocks") correspond to the

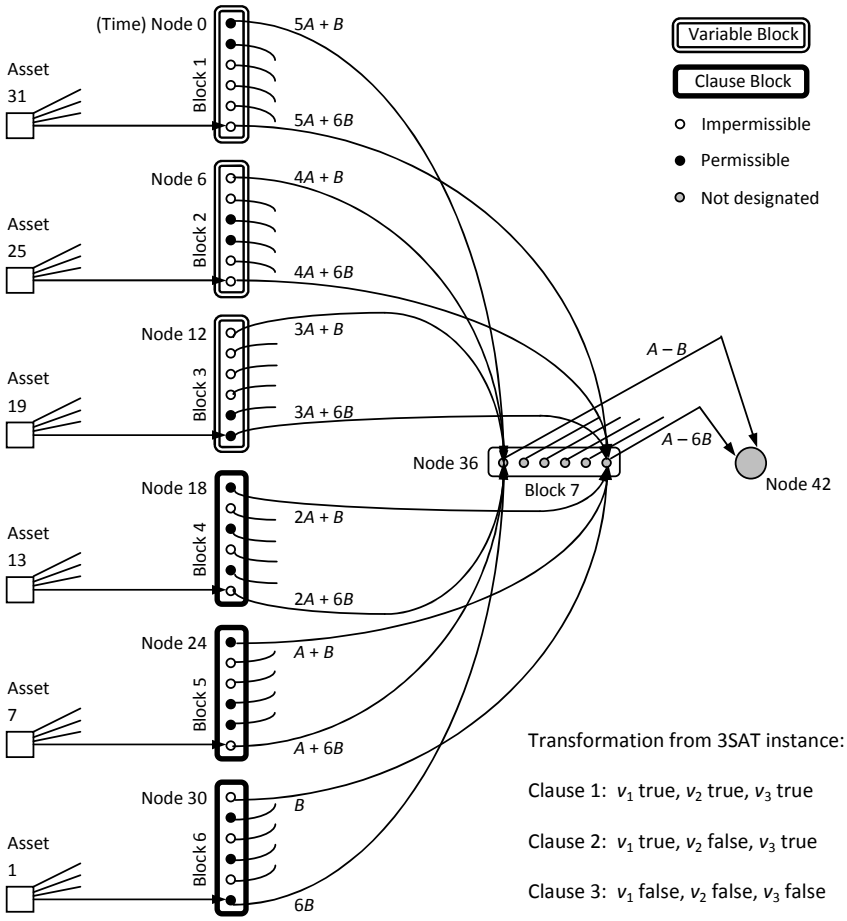

Figure 4-2. Transformation of 3SAT to DPRES

condition that each clause must be satisfied by at least one variable value. The last block will be used to enforce that no more than α total variable values are selected (and hence that every variable takes exactly one value).

We now introduce the initial asset inventory nodes. (We refer to these simply as "asset nodes" below, and to all other nodes as "time nodes" where ambiguity is possible.) We let $n_{2\alpha(\ell-1)+1} = 1$ for $\ell = 1, \ldots, \alpha + \beta$, and $n_i = 0$ for all other values of i. Note that an arc connects $n_{2\alpha(\ell-1)+1}$ to time node $2\alpha(\alpha + \beta - \ell + 1) - 1$. Hence, the asset available with age $2\alpha(\ell - 1) + 1$ is associated with block $\alpha + \beta - \ell + 1$, in the sense that an asset with this age can be replaced at any time period in its associated block (or before). For simplicity, we refer to these initial assets by their associated block.

Next, we state the costs in the transformation. It is helpful to first define $A = 2\alpha(\alpha+1)+1$ and $B = \alpha+1$. The cost goal is set to $G = \binom{\alpha+\beta+1}{2}A + \alpha$. Set all r_{ij}-values to zero. Set the fixed costs to zero for each block except the last one, i.e., $k_j = 0,\ \forall j = 1,\ldots,2\alpha(\alpha+\beta)-1$, and $k_j = 1,\ \forall j = 2\alpha(\alpha+\beta),\ldots,2\alpha(\alpha+\beta+1)-1$. Initialize each c_{ij}-value to $G+1$. We revise these values for some of the outgoing arcs of "permissible nodes." For variable block $i = 1,\ldots,\alpha$, nodes $2(i-1)$ and $2(i-1)+1$ of the block are permissible. For clause blocks $i = \alpha+1,\ldots,\alpha+\beta$, the permissible nodes correspond to the three literal values in clause $i-\alpha$. That is, if clause $i-\alpha$ contains values for variables v_{j_1}, v_{j_2}, and v_{j_3}, then node $2(j_\ell-1)$ in the block is permissible if variable v_{j_ℓ} takes a true value in the clause, and node $2(j_\ell-1)+1$ in the block is permissible if v_{j_ℓ} is false in the clause, for each $\ell = 1,2,3$. All other nodes in blocks $1,\ldots,\alpha+\beta$ are called impermissible. (All asset nodes and time nodes $2\alpha(\alpha+\beta),\ldots,T$ do not receive either designation.)

To revise the c-costs, for each permissible node h corresponding to block $i = 1,\ldots,\alpha+\beta$ and variable value $j \in \{0,\ldots,2\alpha-1\}$, set $c_{h\ell} = (\alpha+\beta-i)A + (j+1)B$, for $\ell = h+1,\ldots,(2\alpha)(\alpha+\beta)+j$. Note that we thus revise the arc costs exiting node h to all nodes up to and including node j of the final block. Also, for node h in the final block corresponding to value $j \in \{0,\ldots,2\alpha-1\}$, set $c_{h\ell} = A-(j+1)B$, $\forall \ell = h+1,\ldots,T$. Note that all costs in the problem are nonnegative, since the smallest of the c-value costs is $A-(2\alpha)B = 1$, and all other r- and k-costs are either 0 or 1.

To establish the equivalence between the 3SAT instance and its transformed DPRES instance, we first need to state and prove the following three claims.

Claim 1. *Disregarding the fixed costs k, the shortest path length from the asset node associated with block $i = 1,\ldots,\alpha+\beta$ to time node T is $(\alpha+\beta-i+1)A$, and uses a three-arc path: (1) from the asset node to a permissible node j in its associated block, (2) from the permissible node to node j in the final block, and (3) from node j in the final block to time node T.*

Suppose by contradiction that the first arc travels to some node in an earlier block $i' = i - j$ for some positive integer j. The least-expensive outgoing arc in i' has cost $(\alpha + \beta - i + j)A + B$, which is greater than the shortest path length listed in the claim. If the first arc in the path goes to an impermissible node, then the next arc would have the prohibitively large cost $G + B$.

We thus have that the first arc visits permissible node $j \in \{0, \ldots, 2\alpha - 1\}$ in block i, and that the second arc's cost is $(\alpha + \beta - i)A + (j + 1)B$. This implies that the cost of the remaining path to node T must not exceed $A - (j + 1)B$. Suppose by contradiction that the second arc terminates in node h corresponding to node $\hat{j} \in \{0, \ldots, 2\alpha - 1\}$ of block $\hat{i}$, where $h \neq 2\alpha(\alpha + \beta) + (j + 1)$. First, if h is greater than this value, then the cost of the arc is $G + B$, which is too large. If h is smaller, then suppose that $\hat{i} = \alpha + \beta + 1$ is in the final block, but $\hat{j} < j$. All outgoing costs from this node are $A - (\hat{j} + 1)B > A - (j + 1)B$ and are thus too expensive to use. Now, suppose that $\hat{i} = \alpha + \beta$. All paths from nodes in block $\alpha + \beta$ to T require at least two arcs, must visit a node in the final block, and have a cost of at least A, which is too expensive. Finally, if $\hat{i} < \alpha + \beta$, then all outgoing arcs from node h have a cost of at least $A + B$, which is also too expensive. Therefore, the second arc must travel to node $2\alpha(\alpha + \beta) + (j + 1)$.

Finally, the third arc must connect directly to T, since the cost from $2\alpha(\alpha+\beta)+(j+1)$ to any larger-indexed node is exactly $A - (j + 1)B$, making the total path length accumulated thus far equal to $(\alpha + \beta - i + 1)A$, and since the only arcs exiting any nodes reachable from h have positive costs.

Claim 2. *Again disregarding fixed costs k, consider any path from the asset node associated with block $i = 1, \ldots, \alpha + \beta$ to time node T. If the path does not follow a three-arc shortest path specified by Claim 1, then its cost is at least $(\alpha + \beta - i + 1)A + B$.*

This claim is true due to the cost inequalities used in proving Claim 1.

Claim 3. *In any feasible solution to* DPRES*, all paths from the asset nodes must follow a shortest path to node T, and exactly α distinct nodes in block $\alpha+\beta+1$ are visited by the paths.*

Note that if all shortest paths are taken from the asset nodes as suggested, the accumulated costs due to these paths (not yet accounting for fixed costs k) is $A(1+\cdots+(\alpha+\beta)) = \binom{\alpha+\beta+1}{2}A$. Note that G is only α more than this term, and hence our remaining costs must not exceed α. Since $B > \alpha$ and all non-shortest paths from the asset nodes to T are at least of length B more than their shortest path lengths, each path from a machine nodes to T is a shortest path. The only other costs are the fixed charges (equal to 1) incurred when a node in block $(\alpha+\beta+1)$ is visited. Since the fixed charge costs we incur must be no more than α, our solution must visit no more than α nodes in block $\alpha+\beta+1$. However, asset nodes corresponding to variable blocks $1,\ldots,\alpha$ all take their shortest paths, thus visiting either node $2(j-1)$ or $2(j-1)+1$ in block $\alpha+\beta+1$, for $j=1,\ldots,\alpha$. Therefore, exactly α nodes in block $\alpha+\beta+1$ are visited.

Given these claims, we now show that if the 3SAT instance has a solution, then DPRES also has a solution. For each variable $i=1,\ldots,\alpha$, if v_i is true, then let the asset node corresponding to variable i use the path with intermediate nodes $j=2(i-1)$ in block i, and $j=2(i-1)$ in block $\alpha+\beta+1$. If false, it uses intermediate nodes $j=2(i-1)+1$ in these two blocks instead. For each clause $i=1,\ldots,\beta$, identify a literal that satisfies the clause and associate it with $j\in\{0,\ldots,2\alpha-1\}$ as described before. For the asset node corresponding to block $\alpha+i$, use intermediate nodes j in blocks $\alpha+i$ and $\alpha+\beta+i$. Note that all paths given are shortest paths. Next, when establishing the paths from asset nodes corresponding to variable blocks, exactly α nodes are visited in block $\alpha+\beta+1$. To see this, observe that when paths are established from the asset nodes corresponding to clause blocks, they use one of the α nodes in block $\alpha+\beta+1$ that was visited in a path from an asset node corresponding to a variable block (since the paths from the clause

blocks correspond to variable values that were selected). Hence, the total cost is $G + \alpha$, and DPRES has a feasible solution.

If the DPRES problem is feasible, Claims 1–3 demonstrate that the form of the solution is exactly as described in the previous paragraph. Hence, a 3SAT solution can be derived by choosing the variable values according to the first node visited by asset nodes corresponding to variable blocks. The solution is verified to be feasible for clause $i = 1, \ldots, \beta$ by finding the first node j in block $\alpha + i$ visited by the path from the asset node corresponding to block $\alpha + i$, and noting that j corresponds to a variable value chosen in the 3SAT solution.

Finally, note that all numerical data used in the problem is polynomially bounded in terms of the 3SAT input size, and so DPRES is strongly NP-complete. This completes the proof. □

4.4 Inequalities for PRES

Luo (2003) proposes valid inequalities associated with the salvages of initial assets and the flow conservation for a different network flow formulation of PRES where each node represents the age of an asset and the corresponding time period. Our inequalities for the minimum cost fixed charge network flow formulation given in Section 4.2 are equivalent to the inequalities of Luo (2003).

Although we derive one class of valid inequalities, we present one subset separately for clarity as follows.

Theorem 2. *The following inequality:*

$$S_{ij} \leq n_i Z_j \quad \forall\ i,\ \forall\ 0 \leq j \leq N - i \tag{4–2}$$

is valid for the PRES formulation (4–1a)–(4–1i).

Proof. First note that this refers to the sale of an initial inventory cluster. As the number of assets in an initial inventory cluster of age i is n_i, then $S_{ij} \leq n_i$. If $S_{ij} > 0$, then $Z_j = 1$ as the sale of an asset requires the purchase of a new asset, which is implied by the

flow balance constraints of the network flow model. Thus, Constraints (4–2) are valid for PRES. □

We refer to these inequalities as "Initial Inventory Inequalities," or IIC. They are illustrated in the following example.

Example 4. Consider an example with $T = 20$, $N = 8$ and $d = 31$ with an initial inventory of assets such that $n_1 = 9$, $n_2 = 8$, $n_3 = 2$, $n_5 = 8$ and $n_6 = 4$. The linear programming relaxation of this problem is such that each asset cluster is kept until its maximum age $N = 8$. This results in the following:

$$Z_j = \{0, 0, 0.13, 0.26, 0, 0.06, 0.26, 0.29, 0, 0, 0.13, 0.26, 0, 0.06, 0.26, 0.29, 0, 0, 0.13, 0.26\}.$$

The six-year old initial asset cluster is kept for two years such that $S_{6,2} = 4$. This results in $Z_2 = 4/31 = 0.13$. The pattern is similar for all clusters as they are all kept to their maximum age. The optimal solution to the linear programming relaxation is 45,470.

We introduce the initial inventory inequalities (4–2) now. For example, for the six-year old cluster, we include:

$$\begin{aligned} S_{6,0} &\leq 4Z_0 \\ S_{6,1} &\leq 4Z_1 \\ S_{6,2} &\leq 4Z_2 \end{aligned}$$

Introducing all IIC (29 in total) results in a linear programming relaxation solution of 48,298 with:

$$Z_j = \{0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0.39, 0, 0, 0.61, 0, 0, 0, 0, 0.39, 0\}.$$

This objective function represents a 6.22% improvement on the lower bound produced by the linear programming relaxation without IIC. In the new solution, the initial clusters of age five and six are retained for two years while the remaining clusters are retained for five years. This results in two clusters of 12 and 19 assets that are retained for 8 years in each

replacement cycle through the end of the horizon. The first 13 periods of the solution with the IIC is given in Figure 4-3.

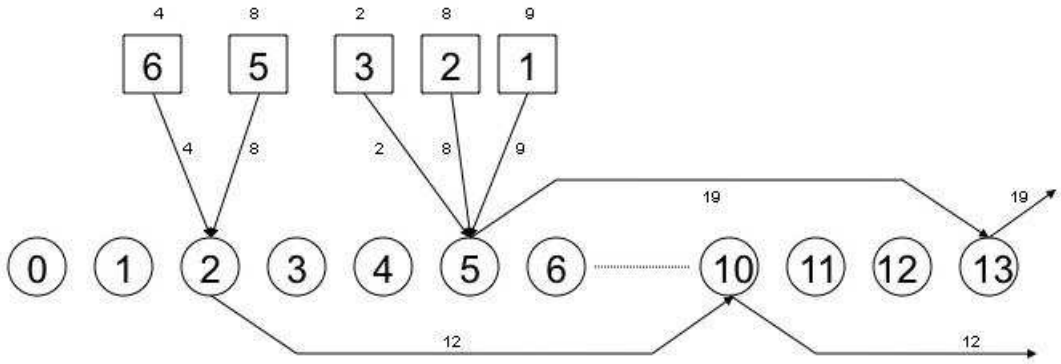

Figure 4-3. Solution network for first 13 periods of the example with IIC.

Our second class of valid inequalities are similar to the traditional flow cover inequalities derived for fixed charge networks (Nemhauser and Wolsey (1988)) in that they are derived by isolating some flow through a node (or set of nodes). However, the inequalities are tailored to the parallel replacement problem in that they use the property stating that every time there is a sale of assets (cluster or clusters), an ensuing purchase must occur as a result of the no-splitting rule (NSR) of Jones et al. (1991) and constant demand. (Recall that for constant demand and homogeneous assets, the NSR states that an optimal solution to PRES exists such that any cluster of same aged assets in the same time period are either kept or replaced in their entirety.) Thus, examining Figure 4-3, we can trace the sale of any cluster throughout the horizon T back to time period 0, when it was merely an initial cluster in inventory. This is critical, as we know the number of assets in each cluster at time zero, and thus we have some information to bound flow through the network. It is these bounds that can be used to tighten Constraints (4–1g).

Flow cover inequalities (Nemhauser and Wolsey (1988)) are derived when the capacity of inflow arcs exceeds the capacity (demand) at a node. These are used to write cover inequalities in order to improve the lower bounds of linear programming relaxations. In our application, there is no situation in which the capacity of inbound arcs exceed the

demand at a node (or equivalently, the capacity of outbound arcs). However, we take a similar tactic by isolating the flow of clusters (on arcs) and identifying their original source, inventory at time zero. We motivate the approach by continuing the previous example.

Example 5. Given the solution posed in the previous example, we note that 4 six-year old assets and 8 five-year old assets at time zero are both replaced at the end of time period two and 12 assets are replaced at time period 10 with age 8, as illustrated in Figure 4-3. These correspond to parameters n_5 and n_6, respectively. Consider nodes labeled 2 and 10 as one "super node". If we only consider a subset of the inflow to this super node, such as $S_{5,2}$ and $S_{6,2}$, then:

$$S_{5,2} + S_{6,2} \leq \sum_{j=3}^{9} X_{2,j} + \sum_{j=11}^{18} X_{10,j}.$$

Further, we know that $S_{5,2}$ and $S_{6,2}$ are bounded by the initial assets at time zero, such that:

$$S_{5,2} + S_{6,2} \leq n_5 + n_6.$$

Our fractional value of Z_{10} resulted from the purchase of assets at time period 10. We isolate all the outflow from node 10 given by $\sum_{j=11}^{18} X_{10,j}$, which is clearly bounded by the demand $d = 31$, such that:

$$S_{5,2} + S_{6,2} - X_{2,3} - X_{2,4} - \ldots - X_{2,9} \leq \sum_{j=11}^{18} X_{10,j} \leq 31 Z_{10}.$$

However, we know that $X_{5,2}$ and $X_{6,2}$ are bounded by n_5 and n_6, respectively, such that: Including this constraint cuts off the fractional value $Z_{10} = 0.39$ in the linear programming relaxation. To cut off the fractional Z_{13} value, the following constraint can also be added:

$$S_{1,5} + S_{2,5} + S_{3,5} - X_{5,6} - X_{5,7} - \ldots - X_{5,12} \leq (n_1 + n_2 + n_3)\, Z_{13} = 19 Z_{13}.$$

Including these two constraints increases the lower bound to 48,555, a 0.532% improvement over the previous lower bound. The network solution is given in Figure 4-4.

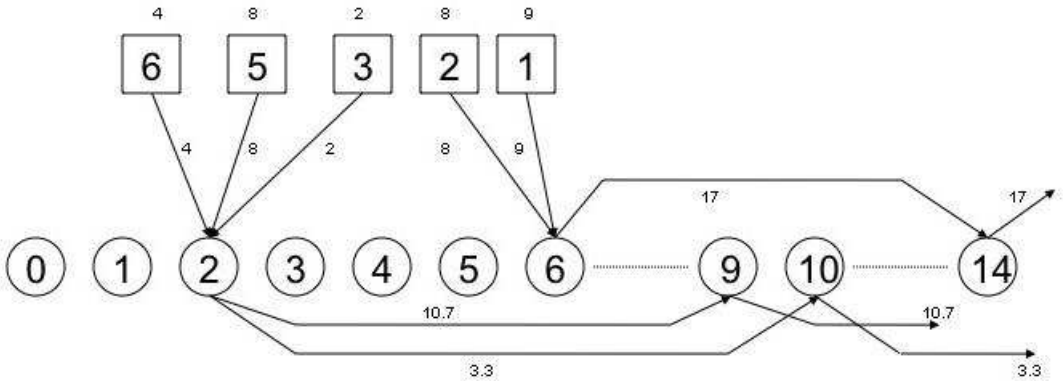

Figure 4-4. Solution network for first 13 periods with IIC and two NSRC inequalities.

As illustrated in the example, each inequality requires three components: the definition of the "super node," the designated flow into the super node and the isolated flow from the super node. To facilitate these definitions, define the following:

1. Inventory supply nodes S defined by $i \in \{1, 2, \ldots, N-1\}$.
2. Transshipment nodes set R defined by $j \in \{0, 1, \ldots, T-1\}$.

With these definitions, further define the following sets:

1. The super node is defined as a set of nodes $P \subseteq R$.
2. Set of inflow nodes $I \subseteq S$.
3. Set of outflow nodes $O \subseteq P$.
4. Set of isolated arcs A defined as (i, j) such that $i \in O$ and $j \in R \backslash O$.

Given these definitions, we can write the following NSRC inequality.

Theorem 3. *The following inequality:*

$$\sum_{i \in I, j \in P} S_{ij} - \sum_{j \in P, k \notin P:(j,k) \notin A} X_{jk} \leq \left(\sum_{i \in I} n_i \right) \sum_{j \in O} Z_j \tag{4–3}$$

is valid for the PRES formulation (4–1a)–(4–1i).

Proof. To prove that this inequality is valid for PRES, one must note that, by definition:

$$\sum_{i \in I, j \in P} S_{ij} \leq \sum_{i \in I} n_i,$$

and thus:

$$\sum_{i\in I, j\in P} S_{ij} - \sum_{j\in P, k\notin P:(j,k)\notin A} X_{jk} \leq \sum_{i\in I} n_i,$$

as all variables are non-negative.

Further note that the sale of assets in O results in a purchase, such that:

$$\sum_{(j,k)\in A} X_{jk} \leq d \sum_{j\in O} Z_j.$$

As:

$$\sum_{i\in I, j\in P} S_{ij} - \sum_{j\in P, k\notin P:(j,k)\notin A} X_{jk} \leq \sum_{(j,k)\in A} X_{jk},$$

and:

$$\sum_{i\in I} n_i \leq d,$$

we can write our valid inequality:

$$\sum_{i\in I, j\in P} S_{ij} - \sum_{j\in P, k\notin P:(j,k)\notin A} X_{jk} \leq \left(\sum_{i\in I} n_i\right) \sum_{j\in O} Z_j$$

□

4.5 Computational Experiments

In this section we present computational experiments to test the effectiveness of the inequalities proposed in the previous section on randomly generated PRES instances. All computations were performed on a computer running Windows XP with a 3.4 GHz CPU and 2 GB memory, using CPLEX 11.0 with an 1800 CPU-second time limit.

4.5.1 Instance Generation

Our instance generation methodology is similar to the instance generation technique of Chen (1998). Specifically, we generate data as follows: We fix the horizon $T = 500$ and consider two sizes for the number of initial clusters, $g = 10$ and 20. The maximum service life of the asset is given by:

$$N = g + [0, \lambda(T - g)]^+,$$

where the parameter $\lambda \in \{0.2, 0.6\}$. For problems with smaller λ, the maximum life N is relatively small compared with the solution horizon T, while for problems with larger λ, the maximum life N is relatively large. These parameters yield large-scale instances that are challenging to solve. As it is commonly assumed that the horizon should be considerably larger than the maximum age of an asset (Bean et al. (1985)) to ensure optimal time zero decisions (for an infinite horizon problem), problems with maximum ages of 50 periods (200 quarters) and horizons of 100 periods (400 quarters) are not uncommon, leading to the need to solve large-scale problems. Other parameters used to generate the problem data are given in Table 4-1 where $U[a, b]$ denotes an integer number drawn uniformly from the interval $[a, b]$. We also consider the fixed cost to purchase cost ratio $f \in \{10, 50, 100\}$. For each of the 12 possible combinations of g, λ, f, five instances, for a total of 60 instances, are generated.

Table 4-1. Data generation for PRES.

Parameters	Data
Age of assets in cluster α_i	$U[0, N]$
Size of cluster n_i	$U[2, 10]$
Fixed cost K	$U[100f, 500f]$
Unit purchase cost p_j	$U[100, 500]$
Unit O&M cost c_{ij}	$U[50, 100]$ and increasing $\beta = U[0, 10]$ each period
Unit salvage value s_{ij}	70 percent of p_j decreasing 30 percent each period

4.5.2 Implementation and Experimental Design

We have implemented a cut-and-branch framework to solve PRES using our two sets of cutting planes, IIC and NSRC. We append the inequalities (4–2) to strengthen the PRES formulation (4–1a)–(4–1i) while we generate the inequalities (4–3) based on the initial optimal LP relaxation solution and feed them to CPLEX to use as cutting planes. We then solve the resulting model using CPLEX with its default settings.

We performed numerous experiments, as there is clearly a tradeoff between the number of inequalities generated and the entire solution time of the algorithm. The solutions are highly dynamic as changes to the solution in the first few periods of the

horizon have a significant effect on solutions in the latter portion of the horizon. This is due to the cyclic nature of the solution (as evident in the network). This fact motivates us to generate valid inequalities early in the horizon before generating inequalities later in the horizon. Thus, in our implementation, we generate as many valid inequalities as we can that cut off the initial optimal fractional solution. To generate NSRC cuts, we pick a fractional Z-value as a part of the initial LP relaxation solution. We then select the time period node corresponding to the fractional Z-value as an outflow node, from which all the outgoing flows are isolated. By backtracking to the initial inventory assets, we determine the nodes and the flows that we use to generate NSRC cuts. We add both IIC and NSRC cuts to the root node, and then solve the augmented formulation by CPLEX.

We compare the efficiency of solving the following models in our computational experiments.

- **base**: Formulation (4–1a)–(4–1i), without adding any user inequalities.
- **iic**: Inequalities (4–2) with base.
- **nsrc**: Inequalities (4–3) with base.
- **iic+nsrc**: Inequalities (4–2) and (4–3) with base.

4.5.3 Summary of Experimental Results

Tables 4-2 and 4-3 summarize the results of our computational experiments. Averages over 5 problem instances are given for:

- **exp**: solution approach,
- $\boldsymbol{\lambda}$: parameter determining the length of the maximum age,
- **g**: number of initial clusters,
- **N**: maximum service life,
- **f**: fixed cost to purchase cost ratio,
- **initgap**: initial solution gap as percentage of the lower bound,

- **gapimp**: improvement in the initial gap over base after adding the IIC and/or NSRC inequalities,
- **ineq**: number of inequalities added to the original formulation,
- **CPXineq**: number of CPLEX inequalities added in the search tree,
- **node**: number of nodes that CPLEX explores in the branch-and-bound tree,
- **time**: total CPU seconds required to solve the problem including inequality generation time.

The required time to generate the proposed cutting planes is less than 0.01 CPU seconds for each instance.

Table 4-2 presents results for $T = 500$ and $\lambda = 0.2$, where each table entry corresponds to the average performance of an algorithm over 5 instances (five each for $f = 10$, 50 and 100). We observe that both of the IIC and NSRC inequalities improve upon the direct solution of PRES by CPLEX. The iic+nsrc strategy appears to provide the most benefit, with a factor of 3.5 improvement in CPU run time over base, averaging over all instances.

Table 4-3 presents results for $T = 500$ and $\lambda = 0.6$, where each table entry corresponds to the average for five instances. We observe that both of the IIC and NSRC inequalities improve upon the direct solution of PRES by CPLEX. Among all the implementations, the iic+nsrc strategy performs best, with a factor 1.5 improvement in the solution time on average.

In Tables 4-2 and 4-3, we observe that as λ and g, and thus N values increase, the instances get harder. In addition, the instances tend to become harder to solve as the fixed cost to purchase cost ratio f increases. The average gap improvement due to the inequalities (4–2) is 46%–50%, while the gap improvement due to the inequalities (4–3) is 3%. Although we generate only between zero and three inequalities (4–3) per instance, we observe that they are quite effective at improving the solution times. In addition, we observe that as the maximum age N decreases, the number and the efficiency of the NSRC

Table 4-2. Summary of experiments for $T = 500$, $\lambda = 0.2$.

	$\lambda = 0.2$								
exp	g	N	f	initgap	gapimp	node	ineq	CPXineq	time
base	10	108	10	0.03	0.00	0	0	7	24.6
iic				0.01	63.33	0	5885	1.6	0.2
nsrc				0.03	2.96	0	0.2	7	19.8
iic+nsrc				0.00	66.29	0	5885.2	1.6	0.2
base			50	0.31	0.00	3055	0	19.6	94.6
iic				0.23	62.57	4011	5885	1.8	70.2
nsrc				0.31	12.39	2039	1	19	62.8
iic+nsrc				0.23	63.02	1983	5885.6	1.8	48.6
base			100	0.38	0.00	338	0	26.2	38.6
iic				0.25	47.75	338	5885	2.4	14.6
nsrc				0.38	0.81	342	1.6	24.8	19.4
iic+nsrc				0.25	48.88	339	5886	1.8	13.8
base	20	116	10	0.04	0.00	0	0	13.6	29.4
iic				0.02	31.56	0	6785	2.2	1.6
nsrc				0.03	1.83	0	1.6	13.4	7
iic+nsrc				0.02	32.20	0	6787.4	2.2	1.6
base			50	0.22	0.00	156	0	13.8	34.4
iic				0.17	37.56	160	6785	0.8	6.6
nsrc				0.22	5.33	149	1.4	13.8	7.8
iic+nsrc				0.17	35.57	159	6786.2	0.6	6.6
base			100	0.16	0.00	16	0	14.8	30
iic				0.12	53.89	25	6785	0.4	3.2
nsrc				0.16	12.14	16	1.2	14.8	13.8
iic+nsrc				0.11	55.20	25	6785.6	0.4	3
Average									
base				0.19	0.00	594	0	15.83	41.93
iic				0.13	49.44	756	6335	1.53	16.07
nsrc				0.19	5.91	424	1.17	15.47	21.77
iic+nsrc				0.13	50.19	418	6336	1.40	12.30

inequalities (4–3) increases. One may consider implementing them in a branch-and-cut algorithm, in which the inequalities (4–3) are generated based on the LP relaxation solutions at each node in the branch-and-bound tree. As our experiments only produced a few instances that required more than 30 minutes of solution time, we only implemented NSRC at the root node to save from the overhead of obtaining LP relaxation solutions at the nodes of the branch-and-bound tree.

4.6 Summary

In this chapter, we study PRES, and show that PRES is NP-hard. We then provide cutting planes for the integer programming formulation of PRES considering constant

Table 4-3. Summary of experiments for $T = 500$, $\lambda = 0.6$.

	$\lambda = 0.6$								
exp	g	N	f	initgap	gapimp	node	ineq	CPXineq	time
base	10	304	10	0.00	0.00	0	0	2.2	59.2
iic				0.00	0.00	0	46359	0	3
nsrc				0.00	0.00	0	0	2.2	59.4
iic+nsrc				0.00	0.00	0	46359	0	3
base			50	0.01	0.00	0	0	5	60
iic				0.00	40.00	0	46359	0	5
nsrc				0.01	0.00	0	0	5	60.4
iic+nsrc				0.00	40.00	0	46359	0	5.2
base			100	0.45	0.00	4907	0	21	583
iic				0.35	65.24	5041	46359	4.4	489
nsrc				0.45	0.03	4868	0.6	21.4	557.8
iic+nsrc				0.35	65.29	5074	46359.4	6.4	516.4
base	20	308	10	0.00	0.00	0	0	0	60.6
iic				0.00	40.00	0	47585	0	2.8
nsrc				0.00	0.00	0	0	0	62
iic+nsrc				0.00	40.00	0	47585	0	2.8
base			50	0.26	0.00	3924	0	22.4	370.8
iic				0.24	43.30	3797	47585	4.4	289.2
nsrc				0.25	0.29	3924	1.2	22.4	334
iic+nsrc				0.24	43.69	3533	47585.6	5.6	289.2
base			100	0.22	0.00	3416	0	39	358.4
iic				0.18	51.79	3676	47585	4	367.8
nsrc				0.22	0.29	3633	1	39	339.6
iic+nsrc				0.18	52.14	3332	47585.6	2.4	293.8
Average									
base				0.16	0.00	2041	0	14.93	248.67
iic				0.13	46.72	2086	46972.00	2.13	192.80
nsrc				0.16	0.10	2071	0.47	15.00	235.53
iic+nsrc				0.13	46.85	1990	46972.27	2.40	185.07

demand. Computational experiments suggest that they are effective in solving the PRES instances.

CHAPTER 5
PARALLEL REPLACEMENT PROBLEM UNDER TECHNOLOGICAL CHANGE AND DETERIORATION

5.1 Introduction

The parallel replacement problem under technological change and deterioration aims to determine minimum cost replacement schedules for a group of assets that operate in parallel and are economically interdependent. As the assets are utilized over time, they may become worn and deteriorate, resulting in increased O&M costs and/or reduced capacity. Assets may also become obsolete, as technological improvements make it possible for newer assets to operate more efficiently, such that O&M costs are lower and capacity is higher than current assets. As noted in Chapter 4 parallel replacement problems are combinatorial since determining the optimal policy requires examining the replacement of groups of assets over time, greatly complicating the analysis when compared to single asset problems.

The literature in parallel replacement analysis generally considers technological improvements and deterioration only in terms of the objective function – costs and revenues. Here, we consider technological change and deterioration in the parallel replacement problem (PRP) both in terms of costs and capacity gains and/or losses. Rajagopalan (1992) studies this in a capacity expansion problem. We model this problem as an integer program and analyze the model to demonstrate how technology and deterioration affect the optimal policy and the optimal cost.

This chapter is outlined as follows. In Section 5.2, we provide an integer programming formulation for the PRES under technological change and deterioration. In Section 5.3, we experimentally analyze the effects of technology and deterioration on the optimal replacement policy. In Section 5.4, we analyze the optimal solution characteristics of PRES under technological change and then derive inequalities based on the optimal solution properties. Section 5.5 discusses implementation strategies that employ these

inequalities and follows with computational results that illustrate the efficiency of our approach.

5.2 Model

We modify the program from the parallel replacement problem in Hartman (2000) with constant demand. We again assume that the capacity at time zero is sufficient to meet demand in the first period, and that no asset in the initial inventory is equal to its maximum service life. We also assume that O&M costs are non-decreasing and salvage values are non-increasing as a function of age.

An asset is identified by its age, $i = 0, 1, \ldots, N$ at the end of time period $j = 0, 1, \ldots, T$. We may retain or salvage an asset after each period unless it reaches its maximum age N, at which time it must be salvaged. The problem is solved over T periods, with purchases allowed at the end of periods $0, 1, \ldots, T-1$. All assets are sold at the end of time period T.

The decision variables are given as follows:

X_{ij} : the number of i-period old assets in use from the end of period j to period $j+1$;

Y_{ij} : the number of i-period old assets in inventory from the end of period j to period $j+1$;

S_{ij} : the number of i-period old assets sold at the end of period j;

B_j : the number of assets purchased at the end of period j;

Z_j : binary variable equal to 1 if a purchase is made in period j, 0 otherwise.

The deterministic costs associated with each of these decisions are defined as follows:

p_j : per-unit cost for a new asset purchased at the end of period j;

k_j : fixed cost for any asset purchase at the end of period j;

c_{ij} : O&M cost for an i-period old asset in use from the end of period j to period $j+1$;

h_{ij} : holding cost for an i-period old asset in inventory from the end of period j to period $j+1$;

r_{ij} : revenue for an i-period old asset salvaged at the end period j.

Other relevant parameters used in this problem are:

a_{ij} : technological change and deterioration parameter for an i-period old asset at period j.

n_i : the number of i-period old assets available (in inventory) at time zero;

d : demand in each period such that $d = \left\lfloor \sum_{i=0}^{N-1} a_{i0} n_i \right\rfloor$.

With these variables and parameters, the integer programming formulation for PRES under technological change and deterioration follows.

$$\min \quad \sum_{j=0}^{T-1} (p_j B_j + k_j Z_j) + \sum_{i=0}^{N-1} \sum_{j=0}^{T-1} (c_{ij} X_{ij} + h_{ij} Y_{ij}) - \sum_{i=1}^{N} \sum_{j=0}^{T} r_{ij} S_{ij} \tag{5–1a}$$

$$\text{s.t.} \quad \sum_{i=0}^{N-1} a_{ij} X_{ij} \geq d \qquad 0 \leq j \leq T-1, \tag{5–1b}$$

$$X_{i0} + Y_{i0} + S_{i0} = n_i \qquad 1 \leq i \leq N-1, \tag{5–1c}$$

$$X_{(i-1)(j-1)} + Y_{(i-1)(j-1)} - X_{ij} - Y_{ij} = S_{ij}$$

$$1 \leq i \leq N-1, 1 \leq j \leq T-1, \tag{5–1d}$$

$$X_{(i-1)(j-1)} + Y_{(i-1)(j-1)} = S_{ij}$$

$$i = N, 1 \leq j \leq T;\ 1 \leq i \leq N,\ j = T, \tag{5–1e}$$

$$X_{0j} + Y_{0j} = B_j \qquad 0 \leq j \leq T-1, \tag{5–1f}$$

$$B_j \leq \lceil d/a' \rceil Z_j \qquad 0 \leq j \leq T-1, \tag{5–1g}$$

$$X_{ij}, Y_{ij} \geq 0 \qquad 0 \leq i \leq N-1, \quad 0 \leq j \leq T, \tag{5–1h}$$

$$S_{ij} \in \{0, 1, 2 \ldots\} \quad 1 \leq i < j \leq T, \tag{5–1i}$$

$$B_j \in \{0, 1, 2 \ldots\}, Z_j \in \{0, 1\} \quad 0 \leq j \leq T-1, \tag{5–1j}$$

where $a' = a_{N,\min\{j+N,T\}}$ for $j = 0, \ldots, T-1$.

The objective function (5–1a) minimizes discounted purchase, inventory and O&M costs less salvage revenues. Constraints (5–1b) ensure that we have enough capacity to satisfy periodic demands. Constraints (5–1c) through (5–1e) are referred to as flow conservation constraints (see network interpretation below). Constraint (5–1f) enforces that purchased assets can either be used or placed directly in inventory. Constraint (5–1g)

includes the fixed charge variable Z_j such that if any assets are purchased, the fixed charge is imposed. As demand d is constant and the corresponding technological change parameter is $a' = a_{N,\min\{j+N,T\}}$ for $j = 0, \ldots, T-1$, the maximum number of assets that can be purchased in any period is $\lceil d/a' \rceil$ since a' gives the minimum possible capacity in an asset's life, which is purchased in period j. B_j and S_{ij} are non-negative integers with Z_j being restricted as binary. Other decision variables are nonnegative continuous variables. A fractional X-value is interpreted as a partial utilization of some asset during a period.

We illustrate the formulation as a network given in Figure 5-1. The network is drawn on two axes with the y-axis representing the age of an asset ($i = 1, \ldots, N$) and the x-axis representing the end of the time period ($j = 1, \ldots, T$). Figure 5-1 represents a problem with $N = 3$ and $T = 5$. The nodes are labeled according to the age of an asset and the end of time period, (i, j), although the labels have been removed from the figure for clarity. The labels a_{01}, a_{12}, a_{23} and a_{34} on top of each node through the diagonal $(0, 1)$ to $(3, 4)$ represent the technological change and deterioration parameters for an i-period old asset at period j. Note that we removed the a-value labels from the remainder of the figure for clarity. We can interpret a_{ij} as the proportion of capacity provided by X_{ij} towards meeting demand. We assume without loss of generality that $a_{00} \neq 1$ (since otherwise a-values can be normalized with demand making the percentage change in capacity more obvious). We define a-values based on the technological change and deterioration as follows.

Technological Change:	a_{0j} is nondecreasing in j,
	a_{i0} is nonincreasing in i,
	$a_{ij} = a_{(i-1),(j-1)}$ for $\forall\ i, j\ (i, j \neq 0)$;
Deterioration:	a_{i0} is nonincreasing in i,
	$a_{ij} \leq a_{(i-1),(j-1)}$ for $\forall\ i, j\ (i, j \neq 0)$;
Deterioration and Technological Change:	a_{0j} is nondecreasing in j,
	a_{i0} is nonincreasing in i,
	$a_{ij} \leq a_{(i-1),(j-1)}$ for $\forall\ i, j\ (i, j \neq 0)$;
No Tech. Change and No Deterioration:	$a_{ij} = 1$ for every i and j.

Referring to Figure 5-1, the number of i-period old initial inventory assets (n_i) is represented by the supply at each node $(i, 0)$, $i > 0$ and $i < N$. (Allowing $n_N > 0$ forces a replacement at time zero, which can be incorporated easily.) The flow into each node $(0, j)$, $j < T$ represents the purchase of assets at the end of time period j (B_j). For this model it is assumed that a purchased asset is new and it must be retained for a minimum of one period before being eligible for salvage. The sale or salvage of an asset is represented as flow, S_{ij}, from any node (i, j), $i > 0$ and $j \geq 0$.

Flow between nodes (i, j) and $(i+1, j+1)$ represents assets in use (X_{ij}) or in inventory (Y_{ij}) from the end of period j to the end of period $j + 1$, at which time the assets are $i + 1$ periods old. Utilized assets represented by dashed arcs in Figure 5-1 contribute to meeting demand while stored assets are represented by curved arcs and do not contribute to meeting demand. We assume here that stored assets "age" each period in storage. As X_{ij} and Y_{ij}-variables are continuous, $X_{ij}/(X_{ij} + Y_{ij})$ represents the utilization level of each asset in operation in a given period assuming demand is spread equally over each cluster.

5.3 Analysis and Insights

This section presents experiments and sensitivity analysis to provide insight into replacement problems under technological change and deterioration. Specifically, we illustrate the model and the optimal decision policy produced under the assumptions of

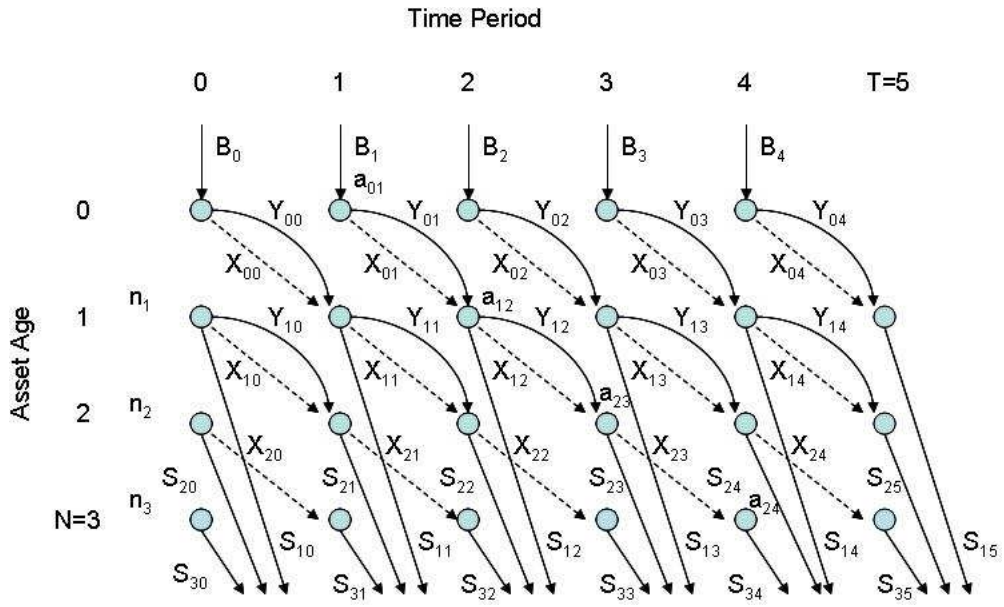

Figure 5-1. Representation of PRES under technology and deterioration as a network with flow representing purchase (B), utilization (X), storage (Y), and salvage (S) variables, initial inventory supply n and technological change and deterioration parameter (a).

no technological change, technological change, deterioration, and both deterioration and technological change.

Experiment 1. The instances of this experiment are generated as follows. We let $T = 30$, $N = 10$, and let the size of the initial inventory of assets be generated from the integer distribution $U[1, 10]$. Demand is constant and equal to the sum of the initial number of assets. The purchase price of a new asset is generated from integer $U[100, 500]$. The fixed charge for asset purchases is generated from integer $U[1000, 5000]$. Per-unit inventory costs for any asset is generated from integer $U[25, 100]$. O&M costs for a new asset are $U[5, 50] * e^{U[0.01,0.1]}$, increasing 10 percent each year. The salvage value of a one-year old asset are 70 percent of the initial price (rounded to an integer value), and decreases by $U[1, 5]$ in each ensuing year. For the example problem, $a_{00} = 1$ and a-values increase by 1 percent each period for the technological change case while decreasing 1 percent for each age increase in the deterioration case.

We generate 10 instances for each of the 4 different cases of technological change, deterioration, both and neither. We solve the instances using CPLEX for each case and illustrate the solution differences.

Case 1: No Technological Change and No Deterioration

In this case the average total cost is \$95053.16, the average replacement age is 6.18 and the average inventory is zero.

Case 2: Technological Change and No Deterioration

In this case the average total cost is \$74289.98, the average replacement age is 6.32 and the average inventory is 0.3.

Case 3: Deterioration and No Technological Change

In this case the average total cost is \$74299.46, the average replacement age is 5.33 and the average inventory is 0.99.

Case 4: Deterioration and Technological Change

In this case the average total cost is \$119880, the average replacement age is 5.38 and the average inventory is 1.39.

Experiment Conclusions. We can draw a number of conclusions from this small experiment. First, we observe that under similar cost structures, the introduction of technology improvements decreases costs while deterioration increases costs. Second, under deterioration, the average replacement age decreases, implying a more frequent replacement policy when compared to no deterioration, while under technological change, the average replacement age increases. Third, average inventory decreases with the introduction of technology improvements while it increases with deterioration. Fourth, CPLEX solution times indicate that incorporating deterioration into the model tends to complicate the solution procedure more than incorporating technological change. This motivates proposing specialized cutting planes for strengthening the problem structure.

Experiment 2. In this experiment, we performed sensitivity analysis on the a-values. We define 100α as the percentage increase in a-values. We generate 5 instances with 5

different values of $\alpha \in \{0.01, 0.15, 0.02, 0.254, 0.5, 0.1\}$, thus a total of 25 instances for each case. By solving these instances, we analyze the effects of the technological change and deterioration on the average replacement age and average inventory. Figures 5-2, 5-3 and 5-4 illustrate the change in the average replacement age as we change the α value for the technological change case, deterioration case and both, respectively. As we increase α-values from 1 percent to 10 percent, the average replacement age increases in the technological change case. From Figure 5-3, we observe that we tend to replace more frequently in the deterioration case, thus the average replacement age reduces as we increase the α value. In Figure 5-4, we observe that the average replacement age decreases as 100α increases from 1 percent to 10 percent at most.

In the technological change case, as we increase the change in a-values from 1 percent to 10 percent, the average inventory reduces, while under deterioration, as we increase the change in a-values from 1 percent to 10 percent, the average inventory increases. Under the case where we have both deterioration and technological change, average inventory increases as we increase the change in a-values from 1 percent to 10 percent.

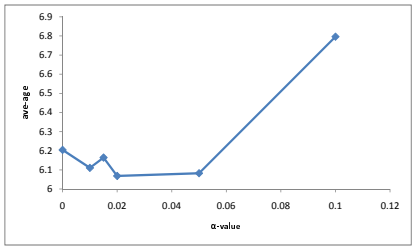

Figure 5-2. Average replacement age vs. α value for the technological change case

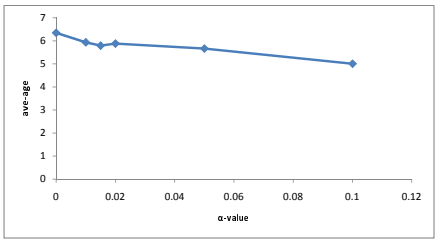

Figure 5-3. Average replacement age vs. α value for the deterioration case

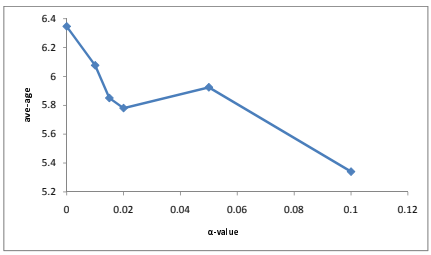

Figure 5-4. Average replacement age vs. α for both the deterioration and technological change case

Conclusions. In this section, we provide insights into the effects of technological change and deterioration on the optimal replacement policy and optimal costs by studying the integer programming formulation for solving the parallel replacement problem under technological change and deterioration given in Section 5.2. We observe that as the change in parameter a increases, the average replacement age of clusters increases in the technological change case, while the average replacement age of clusters decreases under deterioration. Under deterioration, more assets are held in reserve to meet future demand when expecting losses in capacity as we increase deterioration. On the other hand, in the technological change case, fewer assets are needed in inventory as we increase technological change, implying that as technology improves, we tend to keep lower inventory while taking advantage of new technology. We also observe that technological change reduces costs, while deterioration increases the total costs. In addition, incorporating technological change and deterioration into the model tends to complicate the solution procedure.

5.4 Optimization Approaches to PRES under Technological Change

In this section, we investigate the optimal solution characteristics of PRES under technological change without deterioration. The results in this section also apply to the PRES without technological change and deterioration, since it is a special case of PRES under technological change.

5.4.1 Optimal Solution Characteristics

In this section we prove that under some mild cost assumptions, given a period involving a salvage, there must be a purchase in that period. We then use this property to generate cutting planes for PRES under technological change in the next section.

Lemma 3. *If $h_{ij} \leq c_{ij}$ for all i and j, then there exists an optimal solution to PRES such that $\sum_{i=0}^{N-1} a_{ij} X_{ij} = d$ for all j.*

Proof. For any feasible solution that does not obey this lemma, we can construct a new solution X'_{ij} by putting excess capacity $\sum_{i=0}^{N-1} a_{ij} X'_{ij} > d$ into inventory. Then the new solution retains feasibility without increasing cost. □

Theorem 4. *Assume salvage values are non-increasing in age, O&M costs are non-decreasing in age for a fixed period and inventory costs are positive and less than O&M costs. In addition, assume that a_{0j} is nondecreasing in j, a_{i0} is nonincreasing in i and $a_{ij} = a_{(i-1),(j-1)}$ for $\forall\ i, j$ such that $i, j \neq 0$. Then, in any optimal solution to PRES, $Y_{ij} < 1\ \forall\ i$ and j.*

Proof. This is a proof by contradiction. Assume there exists an optimal solution to an instance of PRES such that $Y_{ij} \geq 1$ for some i and j. We will consider two cases: (1) $S_{i+1,j+1} > 0$ and (2) $S_{i+1,j+1} = 0$.

Consider (1). In this case, note that we can define a new solution (using prime notation) where $S'_{ij} = S_{ij} + \min\{Y_{ij}, S_{i+1,j+1}\}$, $I'_{ij} = Y_{ij} - \min\{Y_{ij}, S_{i+1,j+1}\}$ and $S'_{i+1,j+1} = S_{i,j} - \min\{Y_{ij}, S_{i+1,j+1}\}$. Our savings with the new solution is $(r_{i+1,j+1} - r_{ij} + h_{ij}) \min\{Y_{ij}, S_{i+1,j+1}\}$. As $r_{i+1,j+1} \leq r_{ij}$ and costs associated with inventory are reduced, and this constructed solution has a lower cost while retaining feasibility. Thus, the original solution cannot be optimal.

Consider (2). In this case we assume $S_{i+1,j+1} = 0$. $S_{k+1,j+1} > 0$ as only one i, j exists with $Y_{ij} \geq 1$. In this situation, there must exist another age $k \neq i$ such that $S_{k+1,j+1} > 0$ as the demand constraint holds at equality and by assumption, $Y_{l,j+1} < 1$ for all l. Assume

$k < i$ and note the first period t in which assets from the cluster Y_{ij} are salvaged (denote as S_{at}). In this case, construct a new solution such that all, or a portion, of the assets that were in inventory and then utilized, are removed:

$$\begin{aligned}
X'_{i+1,j+1} &= X_{i+1,j+1} - \min\{Y_{ij}, S_{k+1,j+1}\} \\
S'_{i+1,j+1} &= \min\{Y_{ij}, S_{k+1,j+1}\} \\
S'_{i+2,j+2} &= S_{i+2,j+2} - \min\{Y_{ij}, S_{k+1,j+1}\} \\
S'_{k+1,j+1} &= S_{k+1,j+1} - \min\{Y_{ij}, S_{k+1,j+1}\} \\
X'_{k+1,j+1} &= X_{k+1,j+1} + \min\{Y_{ij}, S_{k+1,j+1}\} \\
S'_{k+2,j+2} &= S_{k+2,j+2} + \min\{Y_{ij}, S_{k+1,j+1}\}.
\end{aligned}$$

The assets utilized in demand have been shifted to lower cost assets with higher salvage value and feasibility is maintained. Therefore, the original solution cannot be optimal. Furthermore, this defines a new solution in which there is a sale after assets have been in inventory (Case (1)), which can be further improved. Our savings are given by:

$$\begin{aligned}
(c_{i+1,j+1} - c_{k+1,j+1}) \min\{Y_{ij}, S_{k+1,j+1}\} + (r_{i+1,j+1} - r_{i+2,j+2}) \min\{Y_{ij}, \\
S_{k+1,j+1}\} + (r_{k+1,j+1} - r_{k+2,j+2})(-\min\{Y_{ij}, S_{k+1,j+1}\}).
\end{aligned}$$

Now consider the case where $k > i$. In this case, construct a new solution such that all, or a portion, of the assets that were in inventory are utilized:

$$\begin{aligned}
Y'_{ij} &= Y_{ij} - \min\{Y_{ij}, S_{k+1,j+1}\} \\
X'_{ij} &= X_{ij} + \min\{Y_{ij}, S_{k+1,j+1}\}.
\end{aligned}$$

and the older assets are salvaged sooner:

$$\begin{aligned}
S'_{k,j} &= S_{k,j} + \min\{Y_{ij}, S_{k+1,j+1}\} \\
S'_{k+1,j+1} &= S_{k+1,j+1} - \min\{Y_{ij}, S_{k+1,j+1}\}
\end{aligned}$$

$$X'_{k,j} \quad = \quad X_{k,j} - \min\{Y_{ij}, S_{k+1,j+1}\}.$$

Again, this occurs at lower cost and is feasible. Our savings is given by the term $(-c_{ij} + c_{kj}) \min\{Y_{ij}, S_{k+1,j+1}\} + (r_{kj} - r_{k+1,j+1}) \min\{Y_{ij}, S_{k+1,j+1}\}$.

This construction can be accomplished if there is more than one i and j such that $Y_{ij} \geq 1$. Therefore, by contradiction, an optimal solution to PRES cannot contain an i and j such that $Y_{ij} \geq 1$. □

Corollary 1. *Assume salvage values are non-increasing in age, O&M costs are non-decreasing in age for a fixed period and inventory costs are nonzero and less than O&M. In addition, assume that a_{0j} is nondecreasing in j, a_{i0} is nonincreasing in i and $a_{ij} = a_{(i-1),(j-1)}$ for $\forall\ i, j$ such that $i, j \neq 0$. Then, in any optimal solution, if $S_{ij} > 0$, $B_j > 0$.*

Proof. Follows directly from Theorem 5 and Lemma 3. □

5.4.2 Inequalities

In this section, we extend an equivalent version of the IIC and NSRC inequalities proposed in the study of Luo (2003) to the PRES under technological change problem. By Corollary 1, we know that in an optimal solution, any salvage requires a purchase. Using this property, we modify IIC and NSRC cuts for the PRES under technological case formulation without cutting off any optimal solutions, although they might cut off feasible non-optimal solutions. Hence, while we can employ these inequalities in optimizing PRES under technological change, these are not valid inequalities.

Theorem 5. *The following inequalities:*

$$S_{ij} \leq n_{i-j} Z_j \quad \forall\ i > j,\ \forall\ 0 \leq j \leq N - i \tag{5–2}$$

satisfy all optimal PRES under technological change solutions under the cost and a-value assumptions given in Theorem 5.

Proof. Note that with $i > j$, the inequality (5–2) implies that the number of assets salvaged cannot be greater than the size of the initial inventory cluster. As the number of

assets in an initial inventory cluster of age $i-j$ is n_{i-j}, then:

$$S_{ij} \leq n_{i-j},$$

By Corollary 1 in any optimal solution, if $S_{ij} > 0$, then $Z_j = 1$ when salvage values are non-increasing in age, O&M costs are non-decreasing in age for a fixed period and inventory costs are nonzero and less than O&M costs. Thus, constraints (1) satisfy all optimal PRES under technological change solutions under the specified cost assumptions.

□

Now, we extend the NSRC inequalities given in Chapter 4 for the PRES under technological change. As discussed in Chapter 4, each inequality requires three components:

1. A super node which is a set P' of nodes $(0, j)$ with all of their associated "diagonal nodes" $(1, j+1), (2, j+2), \ldots, (N, N+j)$ for each j defined by all $(0, j)$ nodes.
2. A set I' of inflow nodes with at least one (i, j), $i > j$ for each $(0, j) \in P'$.
3. A set O' of outflow nodes (i, j), $j \geq i$ and $O' \subseteq P'$.

Given these definitions, we can write the following.

Theorem 6. *All optimal PRES under technological change solutions are satisfied by the following inequality:*

$$\sum_{(i,j)\in I'} \frac{a_{i-j,0}}{a_{0j}} S_{ij} - \sum_{(i,j)\in P'\setminus O'} S_{ij} \leq \min_{\forall j:(0,j)\in P'} \left\{ \sum_{(i,j)\in I'} \frac{a_{i-j,0}}{a_{0j}} n_{i-j}, \lceil d/a_{0j} \rceil \right\} \sum_{(i,j)\in O'} Z_j \qquad (5\text{–}3)$$

under the cost and a-value assumptions given in Theorem 5.

Proof. To prove that this inequality satisfies all optimal solutions for PRES under technological change, one must note that, by the equivalence of capacity outflow and capacity inflow:

$$\sum_{(i,j)\in I'} a_{i-j,0} S_{ij} \leq a_{0j} B_j \quad \forall\ j,$$

and by definition:

$$B_j = \sum_{i=1}^{N} S_{ij+i} \quad \forall\ j.$$

In addition, purchased assets are bounded by

$$B_j \leq \lceil d/a_{0j} \rceil\, Z_j \quad \forall\ j.$$

Then the sale of assets in O' results in a purchase, such that:

$$\sum_{(i,j)\in O'} S_{ij} \leq \lceil d/a_{0j} \rceil \sum_{(i,j)\in O'} Z_j.$$

As:

$$\sum_{(i,j)\in I'} \frac{a_{i-j,0}}{a_{0j}} S_{ij} \leq B_j \quad \forall\ j,$$

and since the assets represented by the inflow nodes are bounded by the initial assets at time zero, we have:

$$\sum_{(i,j)\in I'} \frac{a_{i-j,0}}{a_{0j}} S_{ij} \leq \sum_{(i,j)\in I'} \frac{a_{i-j,0}}{a_{0j}} n_{i-j},$$

and we can write our cutting plane:

$$\sum_{(i,j)\in I'} \frac{a_{i-j,0}}{a_{0j}} S_{ij} - \sum_{(i,j)\in P'\setminus O'} S_{ij} \leq \min_{\forall j:(0,j)\in P'} \left\{ \sum_{(i,j)\in I'} \frac{a_{i-j,0}}{a_{0j}} n_{i-j}, \lceil d/a_{0j} \rceil \right\} \sum_{(i,j)\in O'} Z_j$$

□

We define these cutting planes as NSRCTech. They are illustrated in the following example.

Example 6. Consider an example with $T = 20$, $N = 8$ and $d = 32$ with an initial inventory of assets such that $n_1 = 9$, $n_2 = 6$, $n_3 = 3$, $n_5 = 10$ and $n_6 = 4$. The linear programming relaxation of this problem is such that each asset cluster is kept until its maximum age $N = 8$. For this example problem $a_{00} = 1$ and a-values increase by 4 percent in the first period and then 2 percent for each succeeding period for the technological

change case while decreasing by 2 percent for each age (initial inventory) of asset at time period zero. This results in the following fractional solution:

$$Z_j = \{0.07, 0.48, 0, 0, 0, 0, 0.97, 0, 0, 0, 0, 0, 0.98, 0, 0, 0, 0, 0, 0, 0\}.$$

The fractional values corresponds to replacements in periods 0, 1, 6 and 12. The optimal solution to the linear programming relaxation is 115,496.48.

We introduce the initial inventory inequalities (5–2) now. For example, for the two-year old cluster, we include:

$$S_{j+2,j} \quad \leq \quad 6Z_j \; \forall \; j = 0, \ldots, 5.$$

Introducing all IICTech (36 in total) results in a linear programming relaxation solution of 124,461.58 with:

$$Z_j = \{0.08, 0.88, 0, 0, 0, 0, 0.97, 0, 0, 0, 0, 0, 0.98, 0, 0, 0, 0, 0, 0, 0\}.$$

This objective function represents a 25.18% improvement on the lower bound produced by the linear programming relaxation without nsrctech. Given this solution, we note that the 16.7 assets replaced at time period 1 are comprised of the 9 one-year old assets, 3 three-year old assets, 10 five-year old assets and 4 six-year old assets at time zero that are all replaced at the end of time period one. These correspond to parameters n_1, n_3, n_5 and n_6, respectively. Consider nodes, labeled $(i, j), (0, 1), (1, 2), \ldots, (8, 9)$ as one "super node". The flow into this node is:

$$\sum_{j=1}^{N} S_{i,1},$$

while the flow out is:

$$\sum_{j=1}^{N} S_{i,i+1}$$

Purchased capacity should be at least as much as the capacity lost. Thus we have:

$$a_{1,0}S_{2,1} + a_{3,0}S_{4,1} + a_{5,0}S_{6,1} + a_{6,0}S_{7,1} \leq a_{0,1}B_1,$$

and by definition:

$$B_1 = \sum_{i=1}^{8} S_{i,i+1}.$$

Now if we only consider a subset of the inflow to this super node (inflow nodes with fractional values), such as $S_{2,1}$, $S_{4,1}$, $S_{6,1}$ and $S_{7,1}$, then:

$$\frac{(a_{2,1}S_{2,1} + a_{4,1}S_{4,1} + a_{6,1}S_{6,1} + a_{7,1}S_{7,1})}{a_{0,1}} \leq \sum_{i=1, i\neq 5}^{8} S_{i,i+1} + S_{5,6}.$$

Further, we know that $S_{2,1}$, $S_{4,1}$, $S_{6,1}$ and $S_{7,1}$ are bounded by the initial assets at time zero, such that:

$$S_{2,1} + S_{4,1} + S_{6,1} + S_{7,1} \leq n_1 + n_3 + n_5 + n_6$$

and thus:

$$\frac{(a_{2,1}S_{2,1} + a_{4,1}S_{4,1} + a_{6,1}S_{6,1} + a_{7,1}S_{7,1})}{a_{0,1}} \leq \frac{(a_{2,1}n_1 + a_{4,1}n_3 + a_{6,1}n_5 + a_{7,1}n_6)}{a_{0,1}}.$$

Our fractional value of Z_6 resulted from the purchase of assets at time period 6, which was precipitated by asset sales in the same period, or $S_{5,6}$. We isolate this value of $S_{5,6}$, which is clearly bounded by the demand $d/a_{01} = 30.76$, such that:

$$\frac{(a_{2,1}S_{2,1} + a_{4,1}S_{4,1} + a_{6,1}S_{6,1} + a_{7,1}S_{7,1})}{a_{0,1}} - S_{1,2} - S_{2,3} - \cdots - S_{8,9} \leq S_{5,6} \leq 30.76Z_6$$

However, we know that $S_{2,1}$, $S_{4,1}$, $S_{6,1}$ and $S_{7,1}$ are bounded by n_1, n_3, n_5 and n_6, respectively, such that:

$$\begin{aligned} &0.94S_{2,1} + 0.90S_{4,1} + 0.86S_{6,1} + 0.84S_{7,1} - S_{1,2} - S_{2,3} - \cdots - S_{8,9} \\ &\leq \frac{(0.98n_1 + 0.94n_3 + 0.90n_5 + 0.88n_6)}{1.04} Z_6 = 23.23Z_6. \end{aligned} \tag{5–4}$$

Including this constraint cuts off the fractional value $S_{6,12} = 28.1$ in the linear programming relaxation. To cut off the fractional $S_{6,12}$ value, the following constraint can also be added:

$$0.86S_{7,6} + 0.84S_{8,6} - S_{1,7} - S_{2,8} - S_{3,9} - S_{4,10} - S_{5,11} - S_{7,13} - S_{8,14} \leq 12.79Z_{12}.$$

Including these two constraints increases the lower bound to 124,483.58, a 0.07% improvement over the previous lower bound.

5.5 Computational Experiments

In this section we present computational experiments to test the effectiveness of the inequalities proposed in the previous section on randomly generated PRES under technological change instances. All computations were performed on a personal computer running Windows XP with a 3.4 GHz CPU and 2 GB memory, using CPLEX 11.0 with an 1800 CPU-second time limit.

5.5.1 Instance Generation

We loosely follow the instance generation technique of Chen (1998). Specifically, we generate data as follows: We fix the horizon to $T = 100$ and consider two sizes for the number of initial clusters, $g = 10$ and 20. The physical life of the asset is given as:

$$N = g + [0, \lambda(T - g)]^+,$$

where the parameter $\lambda \in \{0.2, 0.6\}$. As before, problems with smaller λ result in a maximum life N that is relatively small compared with the solution horizon T, while for larger λ, the maximum life N is relatively large. Other parameters used to generate the problem data are given in Table 5-1, where $U[a, b]$ denotes an integer number drawn uniformly from the interval $[a, b]$. We also consider the fixed cost to purchase cost ratio $f \in \{10, 50, 100\}$, and the change parameter for a-values $\alpha \in \{0.02, 0.03\}$. For each of the 24 possible combinations of g, λ, α and f, five instances are randomly generated for a total of 120 problems.

Table 5-1. Data generation for PRES under technological change.

Parameters	Data
Age of assets in an initial cluster g_i	$U[0, N]$
Size of cluster n_i	$U[2, 10]$
Unit purchase cost p_j	$U[100, 500]$
Fixed cost k_j	$U[100f, 500f]$
Unit inventory holding cost h_{ij}	$U[1, 50]$
Unit O&M cost c_{ij}	$U[50, 100]$ and increasing $\beta = U[0, 10]$ each period
Unit salvage value r_{ij}	70 percent of p_j decreasing 30 percent each period
a-value	$a_{0,0} = 1$, $a_{i,j} = a_{i-1,j-1}$ and $a_{0,j} = a_{0,j-1} + U[0, \alpha]$

5.5.2 Implementation and Experimental Design

We implemented a cut-and-branch framework to solve PRES using our two sets of cutting planes, IICTech (5–2) and NSRCTech. We add inequalities to the cut pool of CPLEX and solve the PRES under technological change formulation (5–1a)–(5–1j) using CPLEX with its default settings. Since decisions earlier in the horizon have a significant effect on the solutions later in the horizon, we generate valid inequalities and (5–3) early in the horizon. In our implementation, we generate as many NSRCTech inequalities as we can that cut off the initial optimal fractional solution. To generate NSRCTech cuts, we pick a fractional Z-value as a part of the initial LP relaxation solution. We then select the time period node corresponding to the fractional Z-value as an outflow node, from which the flow corresponding to the salvage is isolated. By backtracking to the initial inventory assets, we determine the nodes and the flows that we use to generate NSRCTech cuts. We add both IICTech and NSRCTech cuts to the root node, and then solve the augmented formulation by CPLEX.

We compare the efficiency of the following solution approaches in our computational experiments.

- **base**: Formulation (5–1a)–(5–1j), without adding any user inequalities.
- **iictech**: Inequalities (5–2) with base.
- **nsrctech**: Inequalities (5–3) with base.
- **iictech+nsrctech**: Inequalities (5–2) and (5–3) with base.

5.5.3 Summary of Experimental Results

Tables 5-2 and 5-5 summarize the results of our computational experiments. Averages over 5 problem instances are given for:

- **exp**: solution approach,
- **λ**: parameter determining the length of the maximum age,
- **(g,N)**: number of initial clusters and corresponding maximum age,
- **f**: fixed cost to purchase cost ratio,
- **initgap**: initial solution gap as percentage of the lower bound,
- **gapimp**: improvement in the initial gap over base after adding the IICTech and/or NSRCTech cuts,
- **nodes**: number of nodes that CPLEX explores in the branch-and-bound tree,
- **cutnumber**: number of inequalities added to the original formulation,
- **CPXcuts**: number of CPLEX cuts added in the search tree,
- **cuttime**: the total CPU seconds required to generate the proposed cutting planes,
- **time**: total CPU seconds required to solve the problem including inequality generation time.

Tables 5-2 and 5-3 presents results for $T = 100$ and $\alpha = 0.02$, where each table entry corresponds to the average performance of the algorithm over 5 instances (five each for $f = 10$, 50 and 100). We observe that both of the iictech and nsrctech strategies improve upon the direct solution of PRES under technological change by CPLEX, while the inequalities (5–2) and their combination with (5–3) perform well, improving the results by a factor of 3.5 in CPU run time over base, averaging over all instances. Although we generate only 1.5 NSRCTech cuts per instance on average, their implementation improves the solution time by a factor of 2.5.

We present results for $T = 100$ and $\alpha = 0.03$ in Tables 5-4 and 5-5. We observe that both of the iictech and nsrctech strategies improve the solution of PRES under technological change over CPLEX by a factor of 2, on average.

Table 5-2. Summary of experiments for $T = 100$, $\alpha = 0.02$ and $\lambda = 0.2$.

(g,N)	exp	f	initgap	gapimp	nodes	cutnumber	CPXcuts	cuttime	time
(10,28)	base	10	1.49	0.00	18842	0	159.4	0	79.4
	iictech		1.43	3.82	6526	405.4	132.4	0	30.2
	nsrctech		1.49	0.01	3341	1.4	126.6	0	18.4
	iictech+nsrctech		1.43	3.84	6526	406.8	132.4	0	30.2
	base	50	2.16	0.00	5431	0	151.4	0	45
	iictech		1.98	7.17	13468	406	117.6	0	41.2
	nsrctech		2.16	0.34	14116	2	118.2	0	37.8
	iictech+nsrctech		1.98	7.17	23188	408	117.4	0	71.6
	base	100	4.48	0.00	870	0	143.4	0	19.2
	iictech		3.61	21.02	1377	406	107	0	6
	nsrctech		4.46	0.55	1377	2	107	0	6
	iictech+nsrctech		3.60	21.10	1377	408	107	0	6
(20,36)	base	10	0.85	0.00	2961	0	150	0	29.8
	iictech		0.81	4.31	2251	666	141.6	0	16
	nsrctech		0.85	0.01	1124	1.2	147.4	0	10.6
	iictech+nsrctech		0.81	4.31	2251	667.2	141.6	0	15.8
	base	50	0.92	0.00	533	0	133.2	0	18.6
	iictech		0.84	9.03	1233	666	121.4	0	7.4
	nsrctech		0.92	0.01	337	1.4	120.2	0	4.2
	iictech+nsrctech		0.84	9.03	1233	667.4	121.4	0	7.4
	base	100	1.72	0.00	51158	0	127.2	0.2	370.6
	iictech		1.47	12.54	4785	663.4	102.2	0	22.8
	nsrctech		1.72	0.03	3591	1.8	101.8	0	17.2
	iictech+nsrctech		1.47	12.54	4785	665.2	102.2	0.2	22.8
average	base		1.94	0.00	13299.23	0.00	144.10	0.03	93.77
	iictech		1.69	9.65	4940.07	535.47	120.37	0.00	20.60
	nsrctech		1.93	0.16	3980.97	1.63	120.20	0.00	15.70
	iictech+nsrctech		1.69	9.66	6559.97	537.10	120.33	0.03	25.63

In Tables 5-4 and 5-5 we observe that the difficulty of an instance is positively correlated with λ and g, and thus, N, values. Additionally, the fixed cost to purchase cost ratio f has a positive effect on the difficulty of the instances. The average gap improvement due to the inequalities (5–2) and (5–3) is negatively correlated with N. It is also clear that as the change in α-value parameter α increases from 0.02 to 0.03, the instances tend to become harder to solve.

5.6 Summary

In this chapter, we provide an integer programming formulation for solving the PRES under technological change and deterioration. We analyze the optimal solution characteristics of PRES under technological change, and using these characteristics we extend the cutting planes proposed for PRES in Chapter 4 to the technological change

Table 5-3. Summary of experiments for $T = 100$, $\alpha = 0.02$ and $\lambda = 0.6$.

(g,N)	exp	f	initgap	gapimp	nodes	cutnumber	CPXcuts	cuttime	time
(10,64)	base	10	1.86	0.00	39860	0	132.4	0	369.6
	iictech		1.85	0.90	32134	2074.4	129.2	3.2	365.8
	nsrctech		1.86	0.03	34711	1.2	145.2	0	339.4
	iictech+nsrctech		1.85	0.92	32134	2075.6	129.2	3.2	365.4
	base	50	3.11	0.00	42608	0	101.2	0	237.8
	iictech		3.06	1.36	251	2026.6	120.8	3	7.6
	nsrctech		3.11	0.03	94653	0.8	99.2	0	347.6
	iictech+nsrctech		3.06	1.36	251	2027.4	120.8	3	7.4
	base	100	3.43	0.00	38632	0	141.6	0	438.2
	iictech		3.33	2.38	163	2080	105.6	3	8
	nsrctech		3.42	0.05	138	1.6	105.8	0	3.6
	iictech+nsrctech		3.33	2.38	163	2081.6	105.6	3	7.8
(20,68)	base	10	1.11	0.00	316	0	138.4	0	17.4
	iictech		1.10	0.95	165	2346	119	4	9
	nsrctech		1.11	0.00	146	1.2	119.8	0	3.8
	iictech+nsrctech		1.10	0.96	165	2347.2	119	4	9
	base	50	1.94	0.00	78703	0	132.2	0	888.4
	iictech		1.90	1.71	2177	2337	128.4	3.8	30.8
	nsrctech		1.94	0.08	6657	1.6	136.2	0	56.8
	iictech+nsrctech		1.90	1.72	2177	2338.6	128.4	4	30.8
	base	100	2.02	0.00	42750	0	126.6	0	633.2
	iictech		1.95	3.43	44435	2337.2	126	3.8	373.8
	nsrctech		2.02	0.11	44546	1.4	119.6	0.6	371
	iictech+nsrctech		1.95	3.43	44384	2338.6	126	3.6	373.8
average	base		2.25	0.00	40478.27	0.00	128.73	0.00	430.77
	iictech		2.20	1.79	13220.87	2200.20	121.50	3.47	132.50
	nsrctech		2.24	0.05	30141.83	1.30	120.97	0.10	187.03
	iictech+nsrctech		2.20	1.79	13212.50	2201.50	121.50	3.47	132.37

case. The computational experiments with these cutting planes suggest that they are quite effective in solving PRES under technological change problems in a cut-and-branch algorithm.

Table 5-4. Summary of experiments for $T = 100$, $\alpha = 0.03$ and $\lambda = 0.2$.

(g,N)	exp	f	initgap	gapimp	nodes	cutnumber	CPXcuts	cuttime	time
(10,28)	base	10	1.80	0.00	8727	0	148.2	0	25.2
	iictech		1.66	6.73	1394	400.6	125.2	0	5.8
	nsrctech		1.79	0.05	4361	1.2	132.6	0	12.6
	iictech+nsrctech		1.66	6.73	1394	401.8	125.2	0	5.8
	base	50	3.56	0.00	104799	0	138.4	0	491.8
	iictech		3.33	6.84	26290	406	134.6	0	67
	nsrctech		3.55	0.07	24249	2.4	140.4	0	54.2
	iictech+nsrctech		3.33	6.84	26294	408.4	134.6	0	67
	base	100	3.83	0.00	1018	0	116.6	0	15.2
	iictech		3.05	18.19	355	401.2	99	0	2.8
	nsrctech		3.81	0.31	358	2.2	99.4	0	2.8
	iictech+nsrctech		3.05	18.23	355	403.4	99	0	3
(20,36)	base	10	0.88	0.00	450	0	135.8	0	11.6
	iictech		0.81	7.88	410	664.8	131	0	4.8
	nsrctech		0.88	0.01	389	1.4	130.8	0	3.8
	iictech+nsrctech		0.81	7.88	410	666.2	131	0	4.8
	base	50	1.44	0.00	103867	0	112.8	0.2	724.6
	iictech		1.27	10.06	61626	666	108.4	0	200
	nsrctech		1.44	0.06	135084	1.6	99.2	0	489.4
	iictech+nsrctech		1.27	10.06	61626	667.8	108.8	0.2	200.6
	base	100	2.45	0.00	138440	0	117	0.2	726.6
	iictech		2.39	3.16	94198	662.2	104.4	0	346.6
	nsrctech		2.45	0.19	69003	1.2	106.6	0	238.6
	iictech+nsrctech		2.39	3.20	94198	663.4	104.4	0.2	349.4
average	base		2.33	0.00	59550.37	0.00	128.13	0.07	332.50
	iictech		2.09	8.81	30712.13	533.47	117.10	0.00	104.50
	nsrctech		2.32	0.12	38907.40	1.67	118.17	0.00	133.57
	iictech+nsrctech		2.09	8.83	30712.90	535.17	117.17	0.07	105.10

Table 5-5. Summary of experiments for $T = 100$, $\alpha = 0.03$ and $\lambda = 0.6$.

(g,N)	exp	f	initgap	gapimp	nodes	cutnumber	CPXcuts	cuttime	time
(10,64)	base	10	2.36	0.00	31007	0	148.6	2.4	386.6
	iictech		2.30	3.26	3324	2071.8	141.6	3	38.6
	nsrctech		2.36	0.05	4764	1.4	148.2	0	45.8
	iictech+nsrctech		2.30	3.33	3324	2073.2	141.6	3	38.4
	base	50	2.92	0.00	88642	0	135	3	1455.8
	iictech		2.84	2.87	128696	2061	131.4	3.2	1239.8
	nsrctech		2.92	0.27	108612	1.4	132.6	0	1252.2
	iictech+nsrctech		2.85	2.86	133610	2062.4	130.4	3	1239.8
	base	100	4.54	0.00	82841	0	121.2	1.8	859.4
	iictech		4.31	4.95	76188	2080	121.8	3.4	561.8
	nsrctech		4.54	0.03	76062	1.8	125.2	0	572.4
	iictech+nsrctech		4.31	4.95	75033	2081.8	121.8	3	562.4
(20,68)	base	10	1.25	0.00	3576	0	152.8	2.4	45.4
	iictech		1.23	2.87	384	2346	129.2	3	9.6
	nsrctech		1.25	1.34	4582	1.2	131.4	0	24.2
	iictech+nsrctech		1.23	2.87	384	2347.2	129.2	3	9.4
	base	50	2.15	0.00	42094	0	172.4	2.4	416.6
	iictech		2.12	1.27	38900	2335	143.4	3	375.8
	nsrctech		2.15	0.01	38580	1.2	144.8	0	371.4
	iictech+nsrctech		2.12	1.28	38721	2336.2	143.4	3	375.4
	base	100	3.03	0.00	33597	0	118.4	2.4	1451.2
	iictech		2.91	3.83	67335	2333.6	126	3.2	553.2
	nsrctech		3.03	0.09	82672	2.2	129.6	0	662.8
	iictech+nsrctech		2.91	3.83	77435	2335.8	125.8	3.2	631.4
average	base		2.71	0.00	46959.57	0.00	141.40	2.40	769.17
	iictech		2.62	3.17	52471.10	2204.57	132.23	3.13	463.13
	nsrctech		2.71	0.30	52545.10	1.53	135.30	0.00	488.13
	iictech+nsrctech		2.62	3.19	54751.30	2206.10	132.03	3.03	476.13

CHAPTER 6
CONCLUSIONS AND FUTURE RESEARCH DIRECTIONS

We have presented mixed integer programming approaches for solving capacitated lot-sizing and parallel asset replacement problems. In particular, our study has yielded cutting planes that are effective in solving these problems within a variety of cutting plane algorithms. In this concluding chapter, we summarize our contributions and discuss some promising future research directions.

In Chapter 2, we study the single item capacitated lot-sizing problem (CLSP). We introduce a set of dynamic programming (DP)-based inequalities that can be used to augment the CLSP mixed integer programming formulation. We test several possible implementations that employ these inequalities. Our best implementation appends an initial set of DP based inequalities to the mixed integer programming formulation based on stage information from the partial execution of a forward DP recursion, and then adds lifted cutting-plane inequalities in a cut-and-branch fashion at the root node. Computational experiments show that these valid inequalities drastically reduce the problem solution time. Additionally, we demonstrate that our approach is more efficient than alternative integer programming-based algorithms.

In Chapter 3, we extend our results in Chapter 2 to the multi-item capacitated lot-sizing problem (MCLSP). We use DP based inequalities generated on MCLSP relaxed formulations to tighten the MCLSP formulation. Furthermore, we use integer programming techniques to provide partial objective inequalities over subsets of items and time periods. We analyze lifting techniques for improving these valid inequalities and prescribe a separation algorithm that allows these inequalities to be generated in a cutting plane algorithm.

The computational experiments with the partial objective inequalities suggest that they are effective in solving multi-item lot-sizing problems when used as cutting planes. It is of interest to study the strength of these inequalities. In addition, the

separation heuristic used for finding violated cuts in our computations can be improved. Furthermore, investigation of new computationally effective techniques to compute the lifting coefficients is an important research direction.

The results in Chapters 2 and 3 can be extended to the problem domains that contain the CLSP and MCLSP constraints as a substructure. Our inequalities may also be effective for more general versions of the CLSP and MCLSP, such as variations that include inventory fixed costs or bounds, and variable lower bounds on production and backorders, since these features do not worsen the complexity of the dynamic programming algorithms that are used to generate our proposed inequalities. Furthermore, since the CLSP is a special type capacitated fixed-charge network flow problem, we believe that a better understanding of how the objective function is used to obtain valid inequalities for lot-sizing problems will provide us a better understanding of how to effectively generate valid inequalities to strengthen fixed-charge network flow formulations.

Additionally, there exist several other problems aside from lot-sizing and fixed charge network flow, such as knapsack problems and equipment replacement problems, where one can employ the techniques presented in Chapters 2 and 3. Particularly, some of the results for the partial objective inequalities presented in Chapter 3 have the potential to be extended to general mixed integer programming models. Future research includes generalizing DP based and partial objective inequalities as far as possible to maximize the breadth of problems that can benefit from these approaches.

In Chapter 4, we define valid inequalities for an integer programming formulation for the parallel replacement problem under economies of scale (PRES), which includes fixed and variable costs. PRES is concerned with the replacement schedule (periodic keep and replace decisions) for each individual asset in a group of assets that operate in parallel and are economically interdependent. Specifically, we examine the case where a fixed charge is incurred in each period when an asset is purchased, assuming constant demand. The valid inequalities are motivated by an implication of the "no-splitting

rule" from earlier research, which states that an optimal solution exists such that assets of the same age in the same time period are either kept or replaced as a group. As an asset purchase is triggered in any period in which an asset is sold, our inequalities strengthen the constraints which imposes the fixed charge in these periods. This work was further motivated by proving that the problem was NP-hard, which had not been shown previously in the literature. Computational results show that the valid inequalities substantially improve the solution time, especially for large-scale instances.

Chapter 4 focuses on the homogeneous asset case in which all assets are similar over time and there is only one type of asset available for purchase in each time period. It should be clear that the problem becomes more complicated in the heterogeneous asset case where multiple types of assets are available in each period for replacement over time. This is clearly a more realistic instance as manufacturers and service providers generally have a number of suppliers from which to choose their equipment. Future research includes modifying the valid inequalities for this case.

In Chapter 5, we present an integer programming formulation for solving the PRES under technological change and deterioration. We also provide insights into the effects of technological change and deterioration on the optimal replacement policy and optimal costs. We analyze the optimal solution characteristics of PRES under technological change, and then extend the cutting planes proposed for PRES to the technological change case based on these optimal solution characteristics. The computational experiments with these cutting planes suggest that they are quite effective in solving PRES under technological change problems in a cut-and-branch algorithm. Further research includes varying demands and developing solution approaches for the deterioration case.

REFERENCES

Aggarwal, A., J. K. Park. 1993. Improved algorithms for economic lot size problems. *Operations Research* **41**(3) 549–571.

Atamtürk, A., D. S. Hochbaum. 2001. Capacity acquisition, subcontracting, and lot sizing. *Management Science* **47** 1081–1100.

Atamtürk, A., S. Küçükyavuz. 2005. Lot sizing with inventory bounds and fixed costs: Polyhedral study and computation. *Operations Research* **53**(4) 711–730.

Atamtürk, A., S. Küçükyavuz. 2008. An $O(n^2)$ algorithm for lot sizing with inventory bounds and fixed costs. *Operations Research Letters* **36**(3) 297–299.

Atamtürk, A., J. C. Muñoz. 2004. A study of the lot-sizing polytope. *Mathematical Programming* **99**(3) 443–465.

Baker, K. R., P. Dixon, M. J. Magazine, E. A. Silver. 1978. An algorithm for the dynamic lot-size problem with time-varying production capacity constraints. *Management Science* **24**(16) 1710–1720.

Balas, E. 1975. Facets of the knapsack polytope. *Mathematical Programming* **8** 146–164.

Balas, E., A. Saxena. 2008. Optimizing over the split closure. *Mathematical Programming* **113**(2) 219–240.

Barany, I., T. J. Van Roy, L. A. Wolsey. 1984a. Strong formulations for multi-item capacitated lot sizing. *Management Science* **30**(10) 1255–1261.

Barany, I., T. J. Van Roy, L. A. Wolsey. 1984b. Uncapacitated lot sizing: The convex hull of solutions. *Mathematical Programming Study* **22** 32–43.

Bean, J. C., J. R. Lohmann, R. L. Smith. 1985. A dynamic infinite horizon replacement economy decision model. *The Engineering Economist* **30** 99–120.

Bean, J. C., J. R. Lohmann, R. L. Smith. 1994. Equipment replacement under technological change. *Naval Research Logistics* **41** 117–128.

Bellman, R.E. 1955. Equipment replacement policy. *Journal of the Society for the Industrial Applications of Mathematics* **3** 133–136.

Belvaux, G., L. A. Wolsey. 2000. bc–prod: A specialized branch-and-cut system for lot-sizing problems. *Management Science* **46**(5) 724–738.

Belvaux, G., L. A. Wolsey. 2001. Modelling practical lot-sizing problems as mixed integer programs. *Management Science* **47**(7) 993–1007.

Brahimi, N., S. Dauzere-Peres, N. M. Najid. 2006. Capacitated multi-item lot-sizing problems with time windows. *Operations Research* **54**(5) 951–967.

Chan, T. 1996. Optimal output-sensitive convex hull algorithms in two and three dimensions. *Discrete and Computational Geometry* **16** 361–368.

Chand, S., T. McClurg, J. Ward. 2000. A model for parallel replacement with capacity expansion. *European Journal of Operational Research* **121** 519–531.

Chen, H. D., D. W. Hearn, C. Y. Lee. 1994. A new dynamic programming algorithm for the single item capacitated dynamic lot size model. *Journal of Global Optimization* **4**(3) 285–300.

Chen, W. H., J. H. Thizy. 1991. Analysis of relaxations for the multi-item capacitated lot-sizing problem. *Annals of Operations Research* **1–4** 29–72.

Chen, Z. 1998. Solution algorithms for the parallel replacement problem under economy of scale. *Naval Research Logistics* **45** 279–295.

Childress, S., P. Durango-Cohen. 2005. On parallel machine replacement problems with general replacement cost functions and stochastic deterioration. *Naval Research Logistics* **52** 409–419.

Chung, C. S., J. Flynn, C. H. M. Lin. 1994. An effective algorithm for the capacitated single item lot size problem. *European Journal of Operational Research* **75**(2) 427–440.

Constantino, M. 1996. A cutting plane approach to capacitated lot-sizing with start-up costs. *Mathematical Programming* **75** 353–376.

Constantino, M. 2000. A polyhedral approach to a production planning problem. *Annals of Operations Research* **96** 75–95.

Crowder, H. P., E. L. Johnson, M. W. Padberg. 1983. Solving large-scale zero-one linear programming problems. *Operations Research* **31** 803–834.

de Farias, I. R., E. L. Johnson, G. L. Nemhauser. 2000. A generalized assignment problem with special ordered sets: A polyhedral approach. *Mathematical Programming* **89** 187–203.

de Farias, I. R., E. L. Johnson, G. L. Nemhauser. 2002. Facets of the complementarity knapsack polytope. *Mathematics of Operations Research* **27** 210–226.

de Farias, I. R., G. L. Nemhauser. 2001. A family of inequalities for the generalized assignment polytope. *Operations Research Letters* **29** 49–51.

Diaby, M., H. C. Bahl, M. H. Karwan, S. Zionts. 1992. Capacitated lot-sizing and scheduling by Lagrangean relaxation. *European Journal of Operational Research* **59** 444–458.

Dinkelbach, W. 1967. On nonlinear fractional programming. *Management Science* **13** 492–498.

Easton, T., K. Hooker, E. K. Lee. 2003. Facets of the independent set polytope. *Mathematical Programming* **98** 177–199.

Eppen, G. D., R. K. Martin. 1987. Solving multi-item capacitated lot sizing problems using variable redefinition. *Operations Research* **35**(6) 832–848.

Federgruen, A., M. Tzur. 1991. 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.

Florian, M., M. Klein. 1971. Deterministic production planning with concave costs and capacity constraints. *Management Science* **18**(1) 12–20.

Florian, M., J. K. Lenstra, A. H. G. Rinnooy Kan. 1980. Deterministic production planning: Algorithms and complexity. *Management Science* **26**(7) 669–679.

Fraser, J. M., J.W. Posey. 1989. A framework for replacement decisions. *European Journal of Operational Research* **40** 43–57.

Hartman, J. C. 2000. The parallel replacement problem with demand and capital budgeting constraints. *Naval Research Logistics* **47** 40–56.

Hartman, J. C., J. R. Lohmann. 1997. Multiple options in parallel replacement analysis: Buy, lease or rebuild. *The Engineering Economist* **42** 223–248.

Hopp, W. J., S. K. Nair. 1991. Timing replacement decisions under discontinuous technological change. *Naval Research Logistics* **38** 203–220.

Hritonenko, N., Y. Yatsenko. 2007. Optimal equipment replacement without paradoxes: A continuous analysis. *Operations Research Letters* **35** 245–250.

Ibaraki, T. 1983. Dynamic programming based inequalities for the capacitated lot-sizing problem. *Mathematical Programming* **26** 345–362.

Jans, R., Z. Degraeve. 2004. An industrial extension of the discrete lot-sizing and scheduling problem. *IIE Transactions* **36** 47–58.

Jarvis, R. A. 1973. On the identification of the convex hull of a finite set of points in the plane. *Information Processing Letters* **2** 18–21.

Jones, P. C., J. L. Zydiak, W. J. Hopp. 1991. Parallel machine replacement. *Naval Research Logistics* **38** 351–365.

Küçükyavuz, S., Y. Pochet. 2009. Uncapacitated lot sizing with backlogging: the convex hull. *Mathematical Programming* **118**(1) 151–175.

Leung, J. M. Y., T. L. Magnanti, R. Vachani. 1989. Facets and algorithms for capacitated lot sizing. *Mathematical Programming* **45**(2) 331–359.

Loparic, M., H. Marchand, L. A. Wolsey. 2003. Dynamic knapsack sets and capacitated lot-sizing. *Mathematical Programming* **95** 53–69.

Love, S. F. 1973. Bounded production and inventory models with piecewise concave costs. *Management Science* **20**(3) 313–318.

Luo, S. 2003. Investigations in parallel replacement analysis. Ph.D. thesis, Lehigh University.

Martin, R. K., R. L. Rardin, B. A. Campbell. 1990. Polyhedral characterization of discrete dynamic programming. *Operations Research* **38**(1) 127–138.

McClurg, T., S. Chand. 2002. A parallel replacement model. *Naval Research Logistics* **49** 275–287.

Miller, A. J. 1999. Polyhedral approaches to capacitated lot-sizing problems. PhD thesis, Georgia Institute of Technology.

Miller, A. J., G. L. Nemhauser, M. W. P. Savelsbergh. 2000. On the capacitated lot-sizing and continuous 0-1 knapsack polyhedra. *European Journal of Operational Research* **125**(2) 298–315.

Miller, A. J., G. L. Nemhauser, M. W. P. Savelsbergh. 2003a. A multi-item production planning model with setup times: Algorithms, reformulations, and polyhedral characterization for a special case. *Mathematical Programming* **95** 71–90.

Miller, A. J., G. L. Nemhauser, M. W. P. Savelsbergh. 2003b. On the polyhedral structure of a multi-item production planning model with setup times. *Mathematical Programming* **94** 375–405.

Nemhauser, G. L., J. Wolsey. 1988. *Integer and Combinatorial Optimization*. John Wiley and Sons, New York.

Oakford, R.V., J.R. Lohmann, Salazar A. 1984. A dynamic replacement economy decision model. *IIE Transactions* **16** 65–72.

Padberg, M. W., T. J. van Roy, L. A. Wolsey. 1984. Valid linear inequalities for fixed charge problems. *Operations Research* **32**(4) 842–861.

Padberg, M. W., T. J. van Roy, L. A. Wolsey. 1985. Valid linear inequalities for fixed charge problems. *Operations Research* **33**(4) 842–861.

Pochet, Y. 1988. Valid inequalities and separation for capacitated economic lot sizing. *Operations Research Letters* **7**(3) 109–115.

Pochet, Y., L. A. Wolsey. 1991. Solving multi-item lot-sizing problems using strong cutting planes. *Management Science* **37**(1) 53–67.

Pochet, Y., L. A. Wolsey. 1993. Lot-sizing with constant batches: Formulation and valid inequalities. *Mathematics of Operations Research* **18**(4) 767–785.

Pochet, Y., L. A. Wolsey. 2006. *Production Planning by Mixed Integer Programming*. Springer.

Rajagopalan, S. 1992. Deterministic capacity expansion under deterioration. *Management Science* **38** 525–539.

Rajagopalan, S. 1998. Capacity expansion and equipment replacement: A unified approach. *Operations Research* **46** 846–857.

Regnier, E., G. Sharp, C. Tovey. 2004. Replacement under technological progress. *IIE Transactions* **36** 497–508.

Richard, J-P., I. R. de Farias, G. L. Nemhauser. 2002. Lifted inequalities for 0-1 mixed integer programming: Basic theory and algorithms. *Lecture Notes in Computer Science* 161–175.

Smith-Daniels, V. L, E. D. Smith-Daniels. 1986. A mixed integer programming model for lot sizing and sequencing packaging lines in the process industries. *IIE Transactions* **18**(3) 278–285.

Tang, J., K. Tang. 1993. A note on parallel machine replacement. *Naval Research Logistics* **40** 569–573.

Tempelmeier, H., M. Derstroff. 1996. A lagrangean-based heuristic for dynamic multilevel multiitem constrained lotsizing with setup times. *Management Science* **42**(5) 738–757.

Trigeiro, W. W, L. J. Thomas, J. O. McClain. 1989. Capacitated lot sizing with setup times. *Management Science* **35**(3) 353–366.

Van Hoesel, C. P. M., A. P. M. Wagelmans. 1996. An $O(T^3)$ algorithm for the economic lot-sizing problem with constant capacities. *Management Science* **42**(1) 142–150.

Van Hoesel, C. P. M., A. P. M. Wagelmans. 2001. Fully polynomial approximation schemes for single-item capacitated economic lot-sizing problems. *Mathematics of Operations Research* **26**(2) 339–357.

Wagelmans, A., S. van Hoesel, A. Kolen. 1992. Economic lot sizing: An $O(n \log n)$ algorithm that runs in linear time in the Wagner-Whitin case. *Operations Research* **40** S145–S156.

Wagner, H. M. 1975. *Principles of Operations Research*. Prentice-Hall, Inc., Englewood Cliffs, NJ.

Wagner, H. M., T. M. Whitin. 1958. Dynamic version of the economic lot size model. *Management Science* **5** 89–96.

Wolsey, L. A. 1958. Solving multi-item lot-sizing problems with an mip solver using classification and reformulation. *Management Science* **48**(12) 1587–1602.

Wolsey, L. A. 1975. Faces for a linear inequality in 0-1 variables. *Mathematical Programming* **8** 165–178.

BIOGRAPHICAL SKETCH

İ. Esra Büyüktahtakın was born in Konya, Turkey 1980, to Ayşe and Adem Büyüktahtakın. She graduated from Meram Lycee of Science in 1998. She received her BS degree in industrial engineering from Fatih University in 2002 and MS degree in industrial engineering from Bilkent University in Turkey 2005. After graduation from Bilkent University, she enrolled in the Department of Industrial and Systems Engineering at Lehigh University, where she completed her MS degree in management science in 2007. In August 2007, she joined the Department of Industrial and Systems Engineering at the University of Florida to pursue her PhD degree. Following graduation in August 2009, she will join the Systems and Industrial Engineering Department at the University of Arizona as a Visiting Assistant Professor.