COMPUTATIONAL METHODS FOR BIG BUCKET PRODUCTION PLANNING PROBLEMS: FEASIBLE SOLUTIONS AND STRONG FORMULATIONS

By

Kerem Akartunalı

A Dissertation submitted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

(Industrial Engineering)

at the

UNIVERSITY OF WISCONSIN – MADISON

2007

Abstract

In a global economy, production planning provides significant cost reduction opportunities for manufacturing companies. Since the seminal paper of Wagner and Whitin [1958] on the uncapacitated single-item lot-sizing problem, various forms of production planning problems have been studied by researchers and practitioners. Although widely used in practice, the early MRP (Material Requirements Planning) approach and its successors MRP-II (Manufacturing Resource Planning) and ERP (Enterprise Resource Planning) systems fail to account accurately for capacity restrictions and therefore they often generate infeasible production plans. Improvements in MIP (Mixed Integer Programming) tools and techniques are promising to overcome these challenges.

Given parameters such as costs, demands and capacities, we can define production planning problems as seeking a minimum-cost production plan that satisfies all the constraints of the finite horizon under consideration. Of particular interest for this thesis are multi-level production planning problems with big bucket capacities, i.e., in which multiple items compete for the same resources. These problems frequently occur in practice, yet remain daunting in their difficulty to solve. In this thesis, we study different aspects of these problems.

First, we study how to generate feasible solutions efficiently in practice, and we propose a heuristic framework based on mathematical programming techniques. This simple and flexible framework is not only capable of producing good solutions but also competitive lower bounds compared to some other methods. However, as high duality gaps in the computational results indicate, a polyhedral analysis is necessary for better

understanding of these problems and for more efficient techniques.

Therefore, we next study the polyhedral structure of these problems for a better understanding of why these problems are so hard to solve. Traditional and new methodologies are discussed for obtaining lower bounds, and both theoretical and computational comparisons are provided. The results show that the bottleneck of these problems lies in the multi-item single-level single-machine problems. This provides us the motivation for the analysis of the following chapter.

Finally, we propose a method to generate all valid inequalities based on the convex hull closure of the 2-period subproblems, which may be the simplest model that captures the basis of the difficulty of these problems. The proposed method, "2-period convex hull closure", does not need any predefined inequalities, but it uses a new approach that applies some theory developed using duality. As computational results indicate, these two period closures improve lower bounds significantly by approximating the convex hull of the problems creating the bottleneck. We also analyze some of the generated cuts to achieve a better understanding of these problems in general.

Our main contribution is a thorough analysis of these difficult problems, for which only limited results exist, from both feasible solutions and lower bounds aspects, and generality is an important advantage of our approaches. In particular, the methodology of 2-period convex hull closure is an original approach for production planning problems that also has significant potential to extend to other MIP problems.

Keywords: Production Planning, Integer Programming, Heuristics, Valid Inequalities, Strong Formulations, Convex Hull Closure, Column Generation

Acknowledgements

I would like to express my gratitude to my advisor Andrew Miller. He has always supported me, both mentally and financially, during my five years in Madison, been very patient with my "flawful" writings, and was always ready to meet and discuss. I know very few people who could intellectually inspire me, and I enjoyed every discussion with him whether it was work-related or not. If I know anything today about integer programming and more importantly, how to be a critical researcher, I owe these to him. Thanks for everything, Andrew!

I am grateful to Stephen Robinson for joining my committee and for his feedback and discussions that led to a significant improvement of this thesis. I also would like to thank Michael Ferris, Leyuan Shi and Steve Wright for their willingness to be on my committee, for their comments and for teaching me most of the things I know today.

Special thanks goes to Leena Suhl for introducing me to the fascinating area of operations research and making it so interesting that I am where I am today.

The IE department is a great place with wonderful people. I would like to thank all of my colleagues and friends, particularly Debasis, Jag, Julien, Pat, Shu, Tapana, Turgay, Vahid, Yi-Chun and Yun, for all our conversations and their friendship. Special thanks to Ahmet for being so helpful in my first semester here, to Addi for his amazing willingness to discuss any time, to Mustafa for being the best "Ubuntu provider" ever, to Rikki for being a great friend and to Sushanta for all the tennis games. I also would like to thank ISyE staff Patty, Lisa and Myrna for making my life so much easier.

Finishing a PhD is not all about the school. I am very happy to have joined the

Hoofer Outing Club, which has been an invaluable experience and part of my life, and where I met amazing people. I will also miss the Friday afternoon ultimate frisbee games with "Tons of Fun". And of course, I would like to thank all my friends, in particular April, Jack, Joe, Joie, Julia, Mike, Patricia, Pinar and Tiana, for being part of my life and for their direct and indirect support. Special thanks to Joel for deciding to go sea kayaking one day a few years ago, and to Alp and Ersan for being there all the time.

Finally, I thank my parents for their unconditional love and for being always so supportive, even from thousands of miles away. I am very lucky to have you!

Kerem Akartunali

Madison, August 2007

Dedicated to my parents:

Müzeyyen & Yaşar'a

Abbreviations

B&B	Branch-and-Bound
B&C	Branch-and-Cut
BOM	Bill-Of-Materials
EOQ	Economic Order Quantity
ERP	Enterprise Resource Planning
FL	Facility Location reformulation
JIT	Just-In-Time
LB	Lower Bound
LD	Lagrangian Dual
LP	Linear Programming
LPR	Linear Programming Relaxation
LR	Lagrangian Relaxation
MC	Multi-Commodity reformulation
MIP	Mixed Integer Programming
MIR	Mixed Integer Rounding
MRP	Material Requirements Planning
MRP-II	Manufacturing Resource Planning
SH	Stadtler's Heuristic
SP	Shortest Path reformulation
TDS	Tempelmeier - Derstroff - Stadtler test sets
UB	Upper Bound

Notation and Symbols

$\mathbb{R}^n$	The n-dimensional space of real numbers
$\mathbb{R}^n_+$	The n-dimensional space of nonnegative real numbers
$\mathbb{Z}^n$	The n-dimensional space of integer numbers
$\{0,1\}^n$	The n-dimensional space of binary numbers
$conv(Q)$	Convex hull of the set of points Q
$dim(Q)$	Dimension of a polyhedron Q
$proj_x(Q)$	Projection of a set Q onto the x space
NI	Number of items in the production planning problem
NK	Number of machines in the production planning problem
NT	Number of periods in the production planning problem
$O()$	Big-O notation for problem complexity
$\mathcal{NP}$	The complexity class NP

List of Figures

List of Tables

Contents

Chapter 1

Introduction

1.1 Motivation

In today's business world, manufacturing companies experience fierce competition. With globalization, companies that were in far-away countries just a decade ago have become competitors. Comparatively low wages and costs in some countries is pushing prices of many manufactured goods down. Although manufacturing companies in the U.S. have been achieving average profit margins in the healthy range of 7.2%-8.5% for the last three years (U.S. Census Bureau USC [2006]), it is critical for them to be as efficient as possible, from the strategic level to the operational level.

In this competitive environment, decision making has an increasingly crucial role for manufacturers, probably much more important than it was in the past. In addition, production environments have become very complex, due to technical advances that have created systems that can not be easily managed by unaided human beings, and due to complicated interactions between a company and its suppliers, customers and competitors. This has resulted in a variety of decision support tools created to help the decision makers. Thanks to significant improvements in computer technology, researchers and practitioners have even been able to implement and use some methodologies that were considered too complex in the past. With the help of decision support tools, today's

decision makers have the ability to improve many aspects of their production systems continuously and to achieve cost-efficient results.

One area that has potential for cost improvement is production planning. Production planning investigates different ways of resource allocation to different products (or "items" in the system) and aims to find the most cost-efficient production plan. More specifically, the company's goal is to make many simultaneous decisions, such as how much to produce in a period of a particular product or how much to hold in inventory, with the aim of not losing demand from customers. When achieving this goal, the total cost is minimized. Typical costs in a production environment include but are not limited to production, inventory holding, backordering, fixed setup and changeover costs. In addition to capacities, other complications might arise from requirements such as setup or changeover times, and we will discuss these in more detail in the remainder of the thesis.

These production planning decisions can be achieved at different levels. "Aggregate planning", where many items are aggregated into a few categories according to product similarities, is usually performed on a monthly basis at the strategic level, whereas weekly or daily planning activities are at the operational level. In practice, plans are made at different levels and hence interaction between different levels is necessary. The frameworks described in this thesis are generic and flexible since they are not defined for a particular level only and can be applied to any level of planning.

Since they are obviously present in any manufacturing environment, production planning problems have interested researchers and practitioners for decades. However, many methods used so far in practice often fall short of finding even "good quality" solutions, let alone optimal solutions, mostly due to oversimplifications and too many assumptions. On the other hand, most previous studies have investigated only simplified problems,

such as systems with a single item or separate resources for different items or without capacities. Although some of these studies have significantly helped to understand underlying structures and difficulties of these problems, we cannot make these assumptions in a real world case.

The main focus of this thesis is *big bucket* production planning problems. In big bucket problems multiple products compete for shared resources, i.e., the same resource is used by more than one product, and each resource has a limited capacity. Typical capacitated resources in a production environment are machines, workers, and floor space. This problem class provides a realistic general perspective through which to view production planning problems, and it extends the work accomplished by previous researchers. We will first formulate these problems using Mixed Integer Programming (MIP), and we will investigate two important aspects of these problems: Generating feasible solutions and finding strong lower bounds. It is aimed not only to find efficient solution techniques for these problems, but also to study the underlying difficulties these problems pose, as it is crucial to prove solution quality. The results presented in this thesis are also computationally tested to a great extent using test problems available in the literature. We also believe that the contributions of this work will not be limited to production planning problems only, since concepts for a particular type of problem can be extended to other problems.

The rest of this chapter is organized as follows. In the next section, we give an overview of production planning systems that have been used in industry, and discuss briefly their advantages and disadvantages. Section 3 is devoted to give a fundamental background of Mixed Integer Programming (MIP) that will be used throughout the thesis. In Section 4, we give an MIP formulation for big bucket production planning

problem and discuss in detail possible extensions and characteristics of the problem. Finally, we outline the rest of the thesis.

1.2 Production Planning Systems

Since the industrial revolution, manufacturing has grown from small and family owned businesses to complex systems that have the capability of running thousands of operations at the same time. These complexities have drawn the attention of many practitioners and researchers for decades, and different scientific modeling and solution approaches have been proposed and implemented for production planning. Now we will briefly discuss some of these methodologies that have been milestones in the history of manufacturing.

An early approach that simplifies manufacturing systems significantly is the *Economic Order Quantity* (EOQ) model of Harris [1913]. The model basically assumes constant demand for a single product (or multiple products that do not interact with each other), and it also ignores capacities. According to the EOQ model, the cost function for a period can be defined as follows:

$$C(Q) = \frac{hQ}{2} + \frac{fd}{Q}$$

Here, Q is the variable for the order size (or lot-size), and d is the demand rate (in units per period). The parameters h and f represent the holding (per unit per period) and fixed setup (per order) costs, respectively. Because of the constant demand assumption, the average inventory is $\frac{Q}{2}$, and $\frac{d}{Q}$ gives the number of orders in the period. The optimal lot size, which minimizes total cost, can be calculated by taking the first

derivative and setting it to 0, and the result is

$$Q^* = \sqrt{\frac{2fd}{h}}$$

Obviously, as a single product model with the assumption of constant demand, EOQ is not a realistic model and can be implemented in almost no manufacturing system. However, as an early scientific study of manufacturing systems, it still provides a simple but important insight that there is a tradeoff between inventory and order sizes. Another caveat of the model is that setup costs are usually not well-defined in manufacturing systems.

The constant demand assumption of the EOQ model is generalized by the *Dynamic Lot-Size* model. Dynamic lot-sizing basically approaches the problem of finding the optimal production plan by defining a mathematical model, where demand must be satisfied on time and total cost is minimized, as follows:

$$\min \quad \sum_{t=1}^{NT}(h_t s_t + f_t y_t) \tag{1.1}$$

$$\text{s.t.} \quad s_{t-1} + x_t - s_t = d_t \qquad \forall t = 1, ..., NT \tag{1.2}$$

$$x_t \leq M y_t \qquad \forall t = 1, ..., NT \tag{1.3}$$

$$x_t, s_t \geq 0 \qquad \forall t = 1, ..., NT \tag{1.4}$$

$$y_t \in \{0, 1\} \qquad \forall t = 1, ..., NT \tag{1.5}$$

Here, x_t, s_t and y_t represent the lot-size, inventory held and binary setup variables for each period t, respectively, where the horizon consists of NT periods. (1.2) ensures demand d_t to be satisfied on time, and a big number M in (1.3) provides that setup variable will be forced to become 1 if production occurs in a given period. This model is also referred as the "Uncapacitated Single-Item Lot-Sizing Problem" in the literature, and

it is the basis for many production planning models, which we will discuss throughout this thesis.

This model allows a changing demand rate and the possibility of changing cost figures and lot-sizes from period to period throughout the horizon. However, the system modeled does not differ much from the EOQ model as it still considers a system with a single item, no capacities and well-defined setup costs. However, there is a key observation to this model, known as *Wagner-Whitin Property*, which states that in the optimal production plan, for every period t, either $s_{t-1} = 0$ or $x_t = 0$, i.e., the incoming inventory and the production in a period cannot both be positive. This important insight allows a dynamic programming algorithm, known as *Wagner-Whitin Algorithm*, to solve this problem in polynomial time. For the proof of the property and the details of the algorithm, refer to Wagner and Whitin [1958].

Although EOQ and Wagner-Whitin algorithm are limited, they have provided some important fundamentals for the next generation production planning system, *Material Requirements Planning* (MRP). MRP emerged in the 1960s and has become the principal production planning tool since the early 1970s, particularly thanks to computerized production environments and to the 1972 launch of the "MRP Crusades" of the American Production and Inventory Control Society (APICS) (currently called "The Association for Operations Management"). We will now briefly overview MRP, its immediate successor *Manufacturing Resource Planning* (MRP-II) and its current version *Enterprise Resource Planning* (ERP), which is implemented for example through SAP R/3 and is used worldwide in the manufacturing industry.

One of the important aspects of MRP systems is the so-called "Bill-of-Materials" (BOM). BOM defines the relationship and dependency between different items produced

in the system. Figure 1.1 gives two BOM examples of production structures. Here, each node represents an item and the arcs represent the dependencies between items. The numbers next to the arcs indicate how many of a predecessor item is required to produce a successor item, e.g. only one unit of item 2 is necessary to produce item 1 in (a). There are "end items", which are the final products of the system, such as items 1 and 2 in (b), and these items have external demands. On the other hand, the other items have "internal demand", i.e., demand due to the demand of the end items, but they might also have external demand, e.g. due to spare parts requirements. For example, in (a), if item 1 has an external demand of 10 units in a period, item 3 has an internal demand of 20 units.

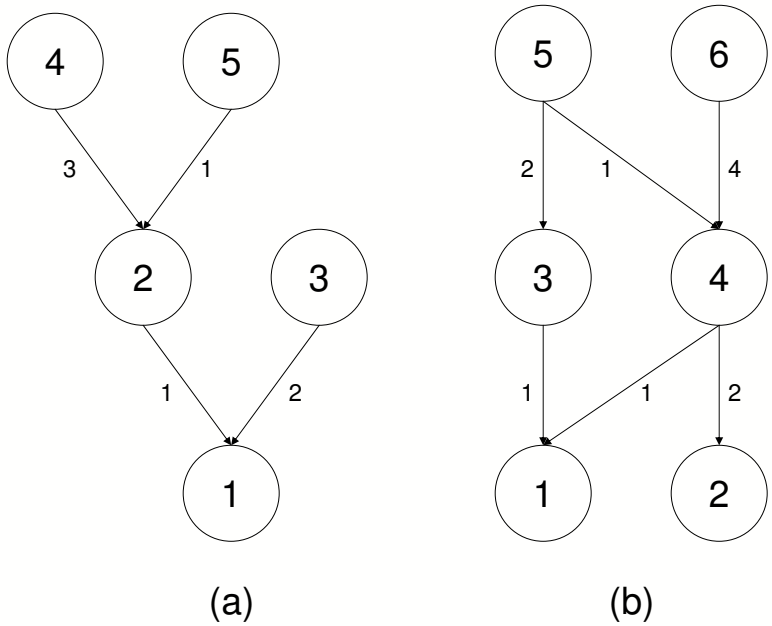

Figure 1.1: BOM structures: Assembly (a) and General (b)

Another common BOM structure is the serial system, where each item has either a successor item or is an end item. In Figure 1.2, we can see such a BOM structure in (a).

One can note that we can see production planning problems as flows on a network and such a network is shown in (b) for the serial BOM of (a). Here, vertical incoming arcs represent production amounts and horizontal arcs represent inventory carried from a period to another (the problem formulation discussion of Section 1.4 will make it clearer). Observe that we can see (a) and (b) as views of a 3-dimensional structure from different angles.

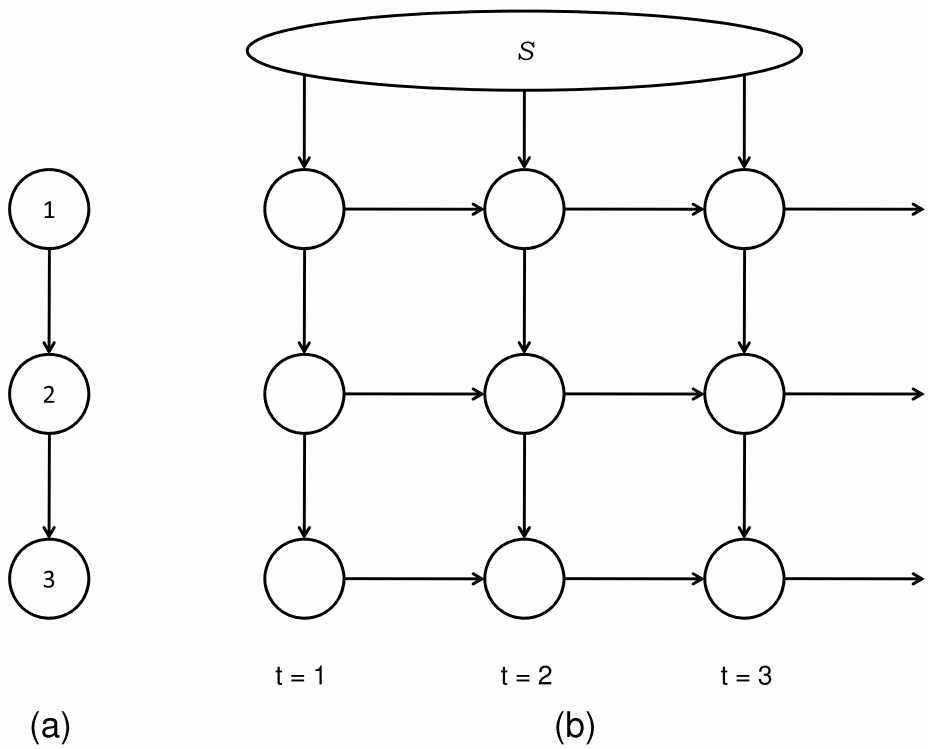

Figure 1.2: Serial BOM structure (a) and network view (b)

The MRP system can be briefly described as follows: First of all, the "Master Production Schedule" (MPS), which is primarily the plan for gross requirements for all the items and for all the periods, is obtained. The MPS also contains information about the on-hand inventory and outstanding orders. Then, the MRP system calculates the net

requirements using the data obtained from the MPS. The next step of MRP is lot-sizing, i.e., the production plan, and it is done separately for each item. Next, the effect of lead times is applied to the production plan. Finally, using the BOM, MRP repeats these steps for all the items, starting from the end items and processing other items next because of internal demand calculations.

Gross requirements for items in the MPS are calculated depending on the system. For example, in a make-to-stock production environment, these figures are achieved through forecasting. The MRP step of interest is the lot-sizing step where decomposition of the main problem is observed since lot-sizing is done separately for each item. For this reason, this step basically aims to solve the uncapacitated lot-sizing problem we discussed for the Wagner-Whitin procedure. However, an important remark to make here is that no MRP system in practice has implemented the Wagner-Whitin procedure that solves the problem optimally. Instead, MRP systems have used different rules and heuristics. For example, the "lot for lot" rule sets the production amounts equal to net requirements to eliminate inventories, and the "fixed order quantity" rule has a predetermined lot size, particularly designed for environments where palettes or carts have a fixed size. For a detailed review of different lot-sizing rules, see Baker [1993]. The final output of MRP is a production plan with release and due dates and number of units for all the items. Finally, note that users can update the inputs of the system, such as gross requirements or on-hand inventories, at any time due to changes in orders, late arrivals of raw materials, and so on. However, troubleshooting can frequently result in inefficiencies as the MRP system generates new production plans.

Although MRP has been a widely used decision support tool in manufacturing for decades, it has some major drawbacks. Probably the most important drawback is the

infinite capacity planning assumption. Since production plans are achieved item by item, this ignores an important characteristic of the production environments, namely sharing of resources by different items. MRP systems can produce production plans that are near or over capacity, and these problems have been tackled manually before the introduction of the MRP-II systems. Another drawback related to the single-item decomposition is that the MRP systems usually provide sub-optimal plans, although some lot-sizing rules has been efficient in practice. One final drawback of MRP is related to lead times. As lead times are constant inputs in the MRP systems and managers are tempted to input "safe" lead times to prevent not meeting customer demand on time, long lead times result in higher inventory levels and less production flexibility for orders.

The next generation production planning system, *Manufacturing Resource Planning* (MRP-II), is defined by APICS as "a method for the effective planning of all resources of a manufacturing company". MRP-II is not only an updated version of MRP, but it is also a complete manufacturing management system with many tools that cover all the areas in a manufacturing company. The production planning aspect of the MRP-II system is hierarchical, i.e., it varies from long-term to short-term planning. The system has MRP as a part of its system, but it has supporting tools to improve its performance.

MRP-II involves some tools to tackle problems with infinite capacity planning of MRP. "Rough-Cut Capacity Planning" (RCCP) is one such tool that works parallel to the MPS, where it basically checks the feasibility of the gross requirements on critical resources. However, it is not as good a tool as "Capacity Requirements Planning" (CRP), which works parallel to MRP instead and hence takes the net amounts and also lead times into consideration. Even though both of these tools provide analysis of capacities and point the user to the problematic resources, they are still not solving the problem of

infinite capacity planning, as the production plans are still generated without capacity restrictions as in MRP, and neither RCCP nor CRP resolve this problem. Hence, the fundamental drawbacks of MRP listed before remain the same for MRP-II.

The successor of these MRP and MRP-II systems have been the *Enterprise Resource Planning* systems. ERP, as its name suggests, is not only a manufacturing management system, but a system designed to integrate the whole of an organization, from manufacturing to financial and HR departments, and its use is not only limited to manufacturing companies. The major attraction of ERP is that it provides full integration of an organization with real time data flow. With the availability of faster computing technologies and with supply chains and production systems becoming much more complex, ERP systems have become popular worldwide, although the implementation of ERP is expensive and might be troublesome for some companies. On the other hand, ERP did not improve any of the drawbacks of the MRP system in the aspect of production planning.

While MRP systems were dominating the manufacturing industry in the U.S., *Just-in-Time* (JIT) evolved in Japan, with the work of Taiichi Ohno at Toyota. As Hopp and Spearman [2000] noted, JIT is "an assortment of attitudes, philosophies, priorities and methodologies that have been collectively labeled JIT", rather than a well-defined manufacturing management system. The major goal of JIT is to have zero inventories, hence very smooth production plans need to be achieved throughout the horizons, although small variances in the production levels will always occur and that will not affect the JIT system. JIT has definitely drawn attention to some important aspects of manufacturing, such as the deadly effects of high Work-in-Process (WIP) levels, the gains of cross training and total quality management, and probably most importantly, the achievability and importance of setup time reductions. However, JIT was particularly

suitable for Japanese industry, which suffered space issues and had close accessibility to suppliers, and it has significantly profited from personal efforts of the people applying the JIT methodologies.

Although we will focus on deterministic systems in this thesis, it is important to note the milestones of stochastic systems that have been proposed for production environments. The first and simplest model we will briefly discuss is the "*Newsboy Problem*", where demand is uncertain, with a known probability distribution (usually assumed to be normal). The model seeks to find the optimal production (or order) quantity for a single period that minimizes the expected sum of overage and underage costs, where the product has no value once the period is over. Using probability theory, one can find the optimal production quantity. The basic insight of the model is that the production quantity depends both on the distribution function and the ratios between per unit overage and underage costs.

The single period approach is not helpful for most realistic systems, and hence more advanced models have been studied that aim to find optimal values of production (or order) quantities and reorder points, i.e., the level of on-hand inventory at which to start to produce (or order). Two models that need to be mentioned here are the *Basestock Model* and the *(Q,r) Model*, where time is modeled continuously. The Basestock Model considers demands occuring one at a time, and the only aim of the model is to find the reorder point, with the availability of backordering. On the other hand, the *(Q,r)* Model extends this model to the possibility of batch demands and finding the optimal values of both production quantity Q and reorder point r is the model's objective. Both models assume fixed lead times, and in addition to minimizing expected costs, achieving high service levels is mainly an additional objective of these models. One of

the main insights these models provide is that safety stocks prevent stockouts. Other insights are that higher holding costs decrease order quantity and reorder level, whereas higher backordering costs or higher service level requirements increase these quantities. Note that these models assume an infinite production capacity, and this is their major drawback. Despite this drawback, these models are widely employed in practice for inventory control systems, see e.g. Tersine [1982].

To conclude this section, it is important to note that every model and system we have seen so far provide some important insights into manufacturing systems, with some disadvantages and assumptions. It is possible that even with their drawbacks, these systems can fit some manufacturing environments quite well, since manufacturing systems are not the same among different companies. However, as noted in Section 1.1, the aim of this thesis is to discuss a generic model that is flexible enough to be employed in different production environments. This is achieved with a mathematical model of the production planning problem, which has been developed from the basic dynamic lot-sizing problem of Wagner and Whitin [1958] discussed in this section. Before discussing the initial formulation of this model, some necessary knowledge of Mixed Integer Programming is provided next for understanding the rest of the thesis. Finally, this section is a very brief overview of manufacturing systems and models, and it is no way complete. Interested reader should refer to the list of literature provided in Section 2.1.

1.3 Mixed Integer Programming

With the great advances in computing technologies and continuously improving commercial and academic solvers available, *Mixed Integer Programming* (MIP) has become a more popular tool in recent years. MIP is the natural way of modeling many real-world and theoretical problems, including some combinatorial optimization problems (for an excellent and complete resource on combinatorial optimization, see Schrijver [2003]). In this section, we will briefly describe what an MIP problem is, and summarize important definitions and theorems that will be useful for the remainder of the thesis. More details on MIP, such as proofs of the stated propositions, can be found for example in Nemhauser and Wolsey [1999], Wolsey [1998] or Schrijver [1986].

An MIP problem is an optimization problem with a linear objective function, linear constraints and some of the variables restricted to take only integer values. Note that we do not refer to problems with a nonlinear objective function or constraints as MIP throughout this thesis. A typical MIP problem can be stated as follows:

$$\min_{x,y}\{c^T x + h^T y | (x,y) \in X\} \quad \text{where} \quad X = \{Ax + Dy \leq b, x \in \mathbb{R}^n, y \in \mathbb{Z}^p\}$$

Here, $x = (x_1, x_2, ..., x_n)$ represents continuous variables, and $y = (y_1, y_2, ..., y_p)$ represents integer variables. The vectors c, h and b are of size n, p and m, respectively. A and D are matrices of size m x n and m x p, respectively. Note that variable bounds can be among the constraints defined by matrices. If $n = 0$, the problem is called an *Integer Program* (IP), and if all integer variables are restricted to be binary, then it is called a *0-1 Program*. X represents the feasible region associated with this problem. Let z be the optimal value of this problem for future reference. First, we review some basics related to polyhedra.

Definition 1.1 *Let $x_1, x_2, ..., x_k \in \mathbb{R}^n$ be points. x is a* ***convex combination*** *of these points if there exist nonnegative numbers λ_i such that $\sum_{i=1}^{k} \lambda_i x_i = x$ and $\sum_{i=1}^{k} \lambda_i = 1$.*

Definition 1.2 *Let $C \subseteq \mathbb{R}^n$ be a set. C is a* ***convex set*** *if any convex combination of any two points $x_1, x_2 \in C$ is in the set C.*

Definition 1.3 *Given a set $C \subseteq \mathbb{R}^n$, the* ***convex hull*** *of C, denoted as $conv(C)$, is the set of all points that are convex combinations of points in C.*

One remark on our notation is for convex hull descriptions. For example, $\{(x, y)|(1.12) \cap conv((1.11))\}$ in our notation represents $\{(x, y)|(1.12) \cap conv(\{(x, y)|(1.11)\})\}$ (or equivalently $\{(x, y)|(1.12), 0 \leq y \leq 1\}$ from our formulation in the next section).

Definition 1.4 *C is a* ***polyhedron*** *if $C \subseteq \mathbb{R}^n$ is a set of points that satisfies a finite number of linear inequalities. A bounded polyhedron is called a* ***polytope****.*

Proposition 1.5 *A polyhedron is a convex set.*

This simple fact is a key in the establishment of polyhedral theory and it ensures solution procedures of Linear Programming (LP) can find an optimal solution. Now, we will focus briefly on X, the feasible region of the general MIP problem. The following propositions are very important results for understanding MIP problems.

Proposition 1.6 *The convex hull of X is a polyhedron.*

Proposition 1.7 *Solving an MIP problem is equivalent to solving it over the convex hull of its feasible region, i.e.,*

$$\min_{x,y}\{c^T x + h^T y | (x, y) \in X\} \quad = \quad \min_{x,y}\{c^T x + h^T y | (x, y) \in conv(X)\}$$

Note that minimizing the objective function $c^T x + h^T y$ over $conv(X)$ is an LP problem and hence $conv(X)$ seems to be the "ideal formulation" of the MIP problem, so one could insist on defining $conv(X)$ to solve the MIP problem. However, this has two main drawbacks. One drawback is that it is generally very difficult to generate all or even some of the inequalities that define the convex hull of X, and the other one is that $conv(X)$ may be defined by an exponential number of inequalities. On the other hand, it is computationally useful to generate at least some good "valid inequalities" for the description of X so that the formulation becomes stronger and hence the MIP problem can be solved more easily.

Definition 1.8 *The inequality $\pi x + \mu y \leq \pi_0$ is a **valid inequality** for X if and only if it is satisfied by all points of X.*

Here, π and μ are vectors of sizes n and p respectively, and π_0 is a scalar. An important remark is that if an equality is valid for X, it is also valid for $conv(X)$. Let $dim(P)$ indicate the dimension of a polyhedron P.

Definition 1.9 *Let $\pi x + \mu y \leq \pi_0$ be a valid inequality for X. Then, $F = \{x, y \in conv(X) | \pi x + \mu y = \pi_0\} \neq \emptyset$ is a **face** of $conv(X)$. If $dim(F) = dim(conv(X)) - 1$, then F is a **facet** of $conv(X)$.*

Facet-defining inequalities dominate any other valid inequalities and hence it is in our best interests to be able to generate some or all of the facets for a better understanding of the structure of an MIP problem.

The **LP relaxation** of the MIP problem can be stated as follows:

$$\min_{x,y}\{c^T x + h^T y | (x, y) \in X^{LP}\} \quad \text{where} \quad X^{LP} = \{Ax + Dy \leq b, x \in \mathbb{R}^n, y \in \mathbb{R}^p\}$$

If we let z^{LP} be the optimal solution of the LP relaxation, it is a known fact that $z \geq z^{LP}$ since $conv(X) \subseteq X^{LP}$.

Definition 1.10 *Given a point* $(x^*, y^*) \in \mathbb{R}^n$ x $\mathbb{R}^p$*, the* ***separation problem*** *for* X *is as follows: Either* (1) *confirm* $(x^*, y^*) \in conv(X)$ *or* (2) *find a violated inequality for* (x^*, y^*)*, i.e., a valid inequality* $\pi x + \mu y \leq \pi_0$ *for* X *such that* $\pi x^* + \mu y^* > \pi_0$*.*

These last definitions provide the necessary structure for cutting plane methods. More specifically, in the cutting plane method, first the LP relaxation of the MIP problem is solved, and then the solution obtained is checked whether it is in the convex hull of the feasible region, and in case it is not, a valid inequality is generated that separates the LP relaxation solution from the convex hull of the feasible region. In this approach, only the valid inequalities that are violated by the LP relaxation solution are generated and added to the formulation. It is important to note that cutting plane methods do not necessarily provide the full description of the convex hull. Cutting plane methods can be used to generate valid inequalities for general MIP problems, as well as to generate specific families of valid inequalities for some particular types of problems.

The traditional MIP solution approach is the "Branch&Bound" (B&B) algorithm, which is basically a tree where each node of the tree is an LP problem. At the root node, B&B solves the LP relaxation, and in case a fractional solution k for an integer variable y is obtained, a constraint $y \leq \lfloor k \rfloor$ or $y \geq \lceil k \rceil$ is added to the LP relaxation to obtain two "child" nodes. While this branching scheme is applied to generate deeper nodes of the tree, an integer solution obtained by a node is a feasible solution for the MIP problem and can be used for bounding purposes. In commercial solvers, cutting planes are usually employed within the Branch&Bound Algorithm, resulting in the "Branch&Cut" (B&C)

algorithm. We will not address these topics in more detail in this thesis, however, interested reader is referred to the literature listed at the beginning of this section.

Next, a significant result from the complexity theory that relates the optimization and separation problems is presented.

Proposition 1.11 *Optimization problem and separation problem are polynomially equivalent, i.e., the following statements are equivalent:*

- *Solving* $\min_{x,y}\{c^T x + h^T y | (x, y) \in conv(X)\}$ *is solvable in polynomial time.*
- *Separating* $(x^*, y^*) \in \mathbb{R}^n$ x $\mathbb{R}^p$ *over* $conv(X)$ *is solvable in polynomial time.*

This result simply states that if we can optimize over a polyhedron in polynomial time, then separation over the same polyhedron is achievable in polynomial time as well.

So far, we discussed the description of a polyhedron by valid inequalities. Another approach to describe a polyhedron is to use extreme points and extreme rays. Although we will omit here the details of how to use extreme points and rays for the description of a polyhedron and will refer the reader again to the listed references (particularly to "Minkowski's Theorem"), short definitions of these terms follow.

Definition 1.12 *Let* $C = \{x \in \mathbb{R}^n | Ax \leq b\}$. *A point* $\bar{x} \in C$ *is an* ***extreme point*** *of* C *if* $\bar{x}$ *cannot be written as a convex combination of any other points in* C .

Extreme points can also be seen as 0-dimensional faces of a polyhedron.

Definition 1.13 *If* $C = \{x \in \mathbb{R}^n | Ax \leq b\} \neq \emptyset$, *any extreme point of the polyhedron* $\{x \in \mathbb{R}^n | Ax \leq 0\}$ *is an* ***extreme ray*** *of* C.

Finally, we will define the "Lagrangian relaxation and duality", and make a remark on this topic. First, let's restate our MIP problem as follows, where the constraints are split into two sets of constraints:

$$\min_{x,y}\{c^T x + h^T y | A_1 x + D_1 y \leq b_1, (x,y) \in \bar{X}\}$$

$$\text{where } \bar{X} = \{A_2 x + D_2 y \leq b_2, x \in \mathbb{R}^n, y \in \mathbb{Z}^p\}$$

Here, the first set of constraints (let k constraints) is assumed to be "complicating constraints" and the second set is assumed to be "nice constraints", in the sense that an MIP problem only with these constraints can be solved easily.

For any constant $\lambda \in \mathbb{R}^k_+$, the **Lagrangian relaxation** of the MIP problem can be defined as $z(\lambda) = \min_{x,y}\{c^T x + h^T y - \lambda(b_1 - A_1 x - D_1 y) | (x,y) \in \bar{X}\}$. Then, the **Lagrangian dual** is defined as $z_{LD} = \max_\lambda\{z(\lambda) | \lambda \geq 0\}$.

Lagrangian relaxation is clearly a relaxation of the original MIP problem. The Lagrangian dual aims to find the highest lower bound for the MIP problem over all Lagrangian relaxations. The following proposition (adapted from Wolsey [1998]) that evaluates the strength of the Lagrangian dual concludes this section.

Proposition 1.14 $z_{LD} = \min_{x,y}\{c^T x + h^T y | A_1 x + D_1 y \leq b_1, (x,y) \in conv(\bar{X})\}$

1.4 Problem Formulation

We consider the following general production planning problem:

$$\min \sum_{t=1}^{NT}\sum_{i=1}^{NI} f_t^i y_t^i + \sum_{t=1}^{NT}\sum_{i=1}^{NI} h_t^i s_t^i \tag{1.6}$$

$$\text{s.t. } x_t^i + s_{t-1}^i - s_t^i = d_t^i \qquad t \in [1, NT], i \in endp \tag{1.7}$$

$$x_t^i + s_{t-1}^i - s_t^i = \sum_{j \in \delta(i)} r^{ij} x_t^j \qquad t \in [1, NT], i \in [1, NI] \backslash endp \tag{1.8}$$

$$\sum_{i=1}^{NI} (a_k^i x_t^i + ST_k^i y_t^i) \leq C_t^k \qquad t \in [1, NT], k \in [1, NK] \tag{1.9}$$

$$x_t^i \leq M_t^i y_t^i \qquad t \in [1, NT], i \in [1, NI] \tag{1.10}$$

$$y \in \{0, 1\}^{NT \text{x} NI} \tag{1.11}$$

$$x \geq 0 \tag{1.12}$$

$$s \geq 0 \tag{1.13}$$

Here M_t^i is the maximum amount of item i that can be produced in t, and can be defined formally as follows:

$$M_t^i = \min(d_{t,NT}^i, \frac{C_t^k - ST_k^i}{a_k^i}) \qquad i \in endp \tag{1.14}$$

$$M_t^i = \min(\sum_{j \in endp} r^{ij} d_{t,NT}^j, \frac{C_t^k - ST_k^i}{a_k^i}) \qquad i \in [1, NI] \backslash endp \tag{1.15}$$

In this formulation, NT, NI and NK are the number of periods, items and machines, respectively, and the set *endp* includes all the end-items, which have only external demand, whereas other items have only internal demand. On the other hand, x_t^i, y_t^i and s_t^i are production, setup and inventory holding variables for item i in period t, respectively. The setup and inventory cost coefficients are represented by f_t^i and h_t^i for each period t and item i. The parameter $\delta(i)$ represents the set of immediate successors of item i,

and the parameter r^{ij} represents the number of items required of i to produce one unit of j. The parameter d_t^i is the demand for end-product i in period t and $d_{t,t'}^i$ is the total demand for end-product i from period t to t', i.e., $d_{t,t'}^i = \sum_{\bar{t}=t}^{t'} d_{\bar{t}}$. The parameter a_k^i represents the variable time necessary to produce one unit of i and ST_k^i the setup time for item i on machine k, which has a capacity of C_t^k in period t.

The objective function incorporates setup and holding costs only, but this is flexible and can be changed and extended if necessary, as discussed later in this section. Constraints (1.7) and (1.8) ensure the production balance and demand satisfaction for end-items and other items respectively, (1.9) are the capacity constraints, (1.10) ensure that the setup variables are set to be 1 if there is positive production, and finally (1.11), (1.12) and (1.13) provide the integrality and nonnegativity requirements. Note that the lead times are assumed to be zero for the simplicity of the formulation, however, it is a matter of changing indices in the balance constraints to include lead times.

For an alternative formulation of the problem that has the ability to incorporate the single-item results, echelon demand parameters D_t^i and echelon stock variables E_t^i are defined (as used by others such as Pochet and Wolsey [1991]):

$$D_t^i = d_t^i + \sum_{j \in \delta(i)} r^{ij} D_t^j \qquad t \in [1, NT], i \in [1, NI] \tag{1.16}$$

$$E_t^i = s_t^i + \sum_{j \in \delta(i)} r^{ij} E_t^j \qquad t \in [1, NT], i \in [1, NI] \tag{1.17}$$

Note that echelon and original demands and stocks are equal for the end-items. Substituting (1.17) into (1.7) and (1.8) for s_t^i, and using the definition (1.16), we obtain the following equation, which can replace (1.7) and (1.8) in the original formulation:

$$x_t^i + E_{t-1}^i - E_t^i = D_t^i \qquad t \in [1, NT], i \in [1, NI] \tag{1.18}$$

To satisfy (1.13), we can define the following constraints:

$$E_t^i \geq \sum_{j \in \delta(i)} r^{ij} E_t^j \qquad t \in [1, NT], i \in [1, NI] \tag{1.19}$$

$$E \geq 0 \tag{1.20}$$

Finally, to eliminate the original inventory variables s, we define echelon inventory holding costs as $H_t^j = h_t^j - \sum_{i=1}^{NI} r^{ij} h_t^i$, and replace the objective function (1.6) with the following function:

$$\sum_{t=1}^{NT} \sum_{i=1}^{NI} f_t^i y_t^i + \sum_{t=1}^{NT} \sum_{i=1}^{NI} H_t^i E_t^i \tag{1.21}$$

Hence, we can define the feasible region as $X = \{(x, y, E) | (1.9) - (1.12), (1.18) - (1.20)\}$, which is the echelon stock reformulation of the original formulation and will be used in the remainder of the thesis as the "basic formulation". The advantage of this reformulation is that it provides the single-item problem structure except level-linking constraints and hence single-item results can be extended, as we will see later. The problem can be defined as: $\min\{(1.21) | (x, y, E) \in X\}$.

We could easily include the possibility of overtime in the problem statement by updating the capacity constraint (1.9) and adding overtime costs to the objective function. Moreover, the possibility of backlogging can be incorporated into the model by defining backlogging variables b_t^i. Aside from the objective function, the only change in the initial formulation would be the replacement of (1.7) with

$$x_t^i + s_{t-1}^i - s_t^i + b_t^i - b_{t-1}^i = d_t^i \qquad t \in [1, NT], i \in endp$$

Similarly, we can include backlogging in the echelon stock reformulation by replacing

(1.18) with the following set of constraints:

$$x_t^i + E_{t-1}^i - E_t^i + \sum_{j \in endp} r^{ij}(b_t^j - b_{t-1}^j) = d_t^i \qquad t \in [1, NT], i \in [1, NI]$$

Note also that the first term in the definition of big-M quantity used in (1.10) should be updated to $D_{1,NT}^i$ since backlogging allows satisfying demand not only for later periods but all periods. Other additional characteristics could be incorporated as well into the model, such as startup and unit production costs, and inventory capacities. However, we will keep the formulation as it is for the sake of visibility, and also note that some of the test problems discussed in the following chapters incorporate some of these factors. The interested reader should refer to Pochet and Wolsey [2006] for an excellent review of production planning models and formulations. We will also provide an extensive literature review of different extensions in the next chapter.

1.5 Outline of the Thesis

Chapter 2 provides a review of the relevant literature. The chapter starts first with the review of the manufacturing systems described in Section 1.2. The rest of the chapter is devoted to the survey of mathematical programming methods, particularly MIP, where both production planning models and more general methodologies are discussed.

In Chapter 3, we define a simple heuristic method using MIP and combining some basic ideas to solve the big bucket production planning problems. The proposed heuristic is flexible enough to handle many types of production planning problems. To show its computational strength, it is tested extensively on different test problems.

Chapter 4 starts with a review of a variety of current methods used to generate lower bounds for production planning problems. Additionally, we propose a few other methods.

We first investigate the theoretical relationships of these techniques and establish some interesting results. These results are also supported by extensive computational tests. In addition to comparisons between different techniques, this chapter also highlights where the bottleneck of the production planning problems is.

In Chapter 5, we propose a new technique to improve the lower bounds of Chapter 4. This technique combines different methods such as column generation and duality, and it aims to generate all the violated cuts that separate a fractional solution from the convex hulls of the 2-period subproblems and therefore obtain the closures of these subproblems. The main advantage of the proposed technique is that it does not depend on predefining families of valid inequalities to generate cuts, although it is one of our goals to use the results of this framework to define facet-defining inequalities for these subproblems. We establish the theoretical background on why and how this method generates valid inequalities, and we present computational results that indicate significant increase in the lower bounds. A thorough analysis and discussion of these results follow.

Chapter 6 is primarily devoted to the main conclusions from the preceding chapters. We also address possible future research directions that can be extended from this thesis.

Chapter 2

Literature Review

In this part, we will first review the literature on manufacturing systems we discussed in Section 1.2 briefly. The second part of the section reviews mixed integer programming (MIP) approaches, which are not only limited to production planning problems but also involve general MIP techniques and recent developments that are related to the work accomplished in this thesis.

2.1 Planning in Manufacturing Systems

The original paper on the *Economic Order Quantity* (EOQ) dates back to Harris [1913]. The paper not only states the basic EOQ model and its solution, but it also points the key insight of the dependency between the order sizes and the inventory levels. This early and very brief paper has been a key motivation for the operations management and management science literature, although as Erlenkotter [1990] noted, EOQ has been falsely cited to other papers for decades due to some misinformation and miscommunication.

Dynamic lot-sizing problem has been stated in the 1950s with the development of the field of operations research. Wagner and Whitin [1958] was one of the first publications on this newly stated problem. The paper proves the optimality condition of having

either production or incoming inventory but not both in each period, and then the simple dynamic programming algorithm, called the *Wagner-Whitin Algorithm* after the authors of this famous paper, is proposed. We will discuss production planning models that extend the results of this paper in the next section, as these models use primarily MIP techniques.

Material Requirements Planning (MRP) was born in the 1960s with the efforts of Joseph Orlicky and his colleagues at IBM. Although there has been a vast amount of literature on the topic of MRP, Orlicky [1975] should be particularly given credit as the key publication from its creator. The next generation of MRP, *Manufacturing Resource Planning* (MRP-II), is primarily the work of Wight [1984], who extended the MRP system to the company-wide approach we previously discussed. Baker [1993] provides an excellent review of models used for different components of the MRP frameworks. With the start of the MRP-II era, these systems have gained diversity due to different software providers; e.g. SAP has been the major software provider for MRP-II and currently *Enterprise Resource Planning* (ERP) systems, with competitors such as Oracle and People-Soft. Therefore, no literature review on ERP systems is provided here. However, interested reader should refer to Davenport [1998] for a general review of advantages and valuable critics on the ERP systems, such as organizational difficulties experienced when these systems are installed.

Just like in MRP, there is a vast amount of literature on *Just-in-Time* (JIT) and Toyota's production system. However, these can be best understood from JIT's father, Ohno [1988]. On the other hand, the *Newsvendor Problem* has been in the literature for a long time with many different variants. Petruzzi and Dada [1999] provides a detailed review of the simple model, variants and extensions on it, with particular emphasis

on the models where pricing is a dependent variable on demand. Stochastic inventory control methods such as *Basestock and (Q,r) Models* has also been investigated since the 1950s. Lee and Nahmias [1993] provide a brief and concise review on these models, along with other single-product models such as EOQ and Wagner-Whitin model. Tersine [1982] provides detailed insight into inventory theory. He discusses both deterministic and probabilistic systems, with an emphasis on inventory control systems. Next, we will discuss some general references that can provide more insight and detail into manufacturing.

Vollmann et al. [1992] is an excellent source for manufacturing planning and control. It investigates many aspects of the manufacturing environment, and provides a solid background on how to plan resources and suppliers. Nahmias [2001] is an excellent textbook for production planning, with detailed reviews of EOQ, MRP and statistical inventory systems and special emphasis given to aggregate planning. It also reviews the basic concepts of project management and supply chain management. Russell and Taylor III [1995] is a strong operations management book that provides background on most aspects of the manufacturing environment other than production planning. It discusses quality management and statistical quality control, product and service design, and facility layout planning, among many other topics.

Viswanadham and Narahari [1992] is an excellent source on performance analysis of manufacturing systems. It reviews important measures for a manufacturing system, such as throughput and cycle time, and presents solid methodologies for calculating these measures using analytical methods. Hopp and Spearman [2000] not only reviews the history of manufacturing and the production planning systems we discussed in Section 1.2,

but it also provides excellent critiques and discussions of these production planning systems. It also presents the authors' new concepts and approaches on manufacturing based on queueing theory considerations, after a detailed analysis of the systems. Similarly, Suri [1998] develops new methodologies to apply to manufacturing systems, with the emphasis given to lead times, after an analysis of dependencies and relationships between performance measures. Finally, as we have been referring partially to its chapters, we will give credit to Graves et al. [1993] as an excellent review of mathematical methods used in the areas of production and inventory planning.

2.2 Mathematical Programming Approaches

The early MRP approach, while accounting for bill-of-material structures, follows decision rules that are too simple to achieve feasible plans (let alone high quality plans) consistently. The more advanced systems of MRP-II and ERP combine different methodologies such as MRP and aggregate planning, but they fail to account accurately for capacity. In this thesis, we investigate multi-level capacitated production planning problems, for which MRP-II and ERP systems aim to find feasible production plans, and as we will use mathematical programming (particularly mixed integer programming) approaches for modeling and solving these problems, this section is intended in general to give a detailed review of the literature on related subjects. The simple lot-sizing problem and its extensions we will review in this section provide the necessary core for general production planning problems, as these simple problems exist as substructures in the general production planning problems.

As noted before, the seminal paper of Wagner and Whitin [1958] is the milestone

that started the analysis of various forms of production planning problems. The paper investigates the single-item uncapacitated lot-sizing problem with constant production costs and nonnegative inventory holding costs. After the crucial observation that incoming inventory and production in a period both cannot be positive, an $O(n^2)$ dynamic programming algorithm (n expressing number of periods) is established, by decomposing the horizon in consideration into several subsets where subsets are defined by periods in which inventory is zero (hence building a shortest path problem). We will now define two terms that are named after this paper and used widely in the literature, using our notation from the Section 1.4. Let p_t^i and Δ_t^i be the unit production and backlogging costs of item i in period t, respectively.

Definition 2.1 *The property that $x_t s_{t-1} = 0$, i.e., either $x_t = 0$ or $s_{t-1} = 0$ or both, is called the* ***Wagner-Whitin Property****.*

Definition 2.2 *A lot-sizing problem that does not have any speculative costs, i.e., in which $p_t^i + h_t^i \geq p_{t+1}^i$ and $p_{t+1}^i + \Delta_t^i \geq p_t^i$ for all t, is said to have* ***Wagner-Whitin Costs****.*

Even though some special cases of lot-sizing problems can be solved in polynomial time, real-world problems are much more complicated and computationally challenging, and efficient methods are needed to tackle these difficulties. Moreover, as Florian et al. [1980] have proven, the capacitated version of even the single-item lot-sizing problem is $\mathcal{NP}$-hard, so these problems are theoretically hard to solve as well.

The solution approaches proposed so far for the production planning problems vary from exact approaches based on mathematical programming to heuristic methods. Mathematical programming in general attempts to improve the formulations so that these can

be solved faster and more efficiently, and in principle, these methods provide exact solutions with lower bounds. On the other hand, heuristics are practical methods to quickly discover good solutions, not necessarily the best, and these methods usually do not provide lower bounds that would guarantee solution quality.

Mathematical programming provides tools for exact solution approaches for MIP problems, and we will give an overview of such approaches for production planning models in this section. First, we will review polynomial-time algorithms (such as the Wagner-Whitin algorithm) for the special cases of the problem, and then provide a detailed review of polyhedral approaches. The third part is devoted to heuristic methods (although these methods do not necessarily employ mathematical programming) and it is concluded with a short literature review of some MIP techniques we will use later in the thesis.

2.2.1 Polynomial Algorithms: Special Cases of Lot-Sizing

Exact polynomial-time algorithms in lot-sizing are usually based on dynamic programming, in a similar fashion to the Wagner-Whitin algorithm, where a property of the optimal solution in a special case is used for decomposing the problem into smaller problems and then these problems are solved recursively. In many cases, network algorithms such as shortest path algorithms are used after a special structure is obtained. Hence, we will refer the interested reader to Ahuja et al. [1993] as an excellent resource for the details of network optimization.

Early studies with exact algorithms are based on the basic extensions of the uncapacitated single-item problem. Zangwill [1969] studies the single-item lot-sizing problem

with backlogging and the serial multi-level problem, both with no capacities and general concave cost functions. Using the Wagner-Whitin property and single-source flow networks, the author proposes a dynamic programming framework that works for both of the problems. Florian and Klein [1971] extend the case to single-item problems with capacities. The authors first discover the optimality properties for these problems, both for the cases with and without backlogging. The remarkable observation of the study in case of constant capacities is that in an optimal production plan, the horizon can be decomposed into separate intervals, in which production in a period is either 0 or at full capacity, and in at most one period, it is strictly between these. This property allows an $O(n^4)$ dynamic programming algorithm to solve these problems exactly. A special case of this problem, production with constant batches in a constant capacity environment, is studied by Pochet and Wolsey [1993], with special emphasis given to polyhedral analysis. They propose a dynamic programming algorithm for this case with a running time of $O(n^3)$.

The basic algorithms of Wagner and Whitin [1958] and Florian and Klein [1971] were not improved for decades. Three independent studies in the early 1990s have improved the $O(n^2)$ time of the Wagner-Whitin algorithm to $O(nlogn)$ (and to $O(n)$ in some special cases): Federgruen and Tzur [1991] study the problem with linear and fixed costs, and propose a forward-recursion dynamic programming algorithm. The authors also propose algorithms for the special cases of Wagner-Whitin costs and nondecreasing setup costs that run in $O(n)$ time. Wagelmans et al. [1992] investigate the case with linear cost functions, with the possibility of negative costs. The paper provides a simple backward-recursion algorithm that has a running time $O(n)$ for the special case of Wagner-Whitin costs. Aggarwal and Park [1993] accomplish a detailed study of the

single-item uncapacitated problem with different cost structures, and with and without backlogging. In addition to matching results with the other two papers using dynamic programming and some search techniques, they also accomplish an improvement from $O(n^3)$ of Zangwill [1969] algorithm to $O(n^2)$ for the case with backlogging and general concave cost functions. On the other hand, van Hoesel and Wagelmans [1996] propose an $O(n^3)$ dynamic programming algorithm for the constant capacity single-item problem with concave production costs and linear holding costs. Although this is an improvement over the $O(n^4)$ algorithm of Florian and Klein [1971], it is important to note that the linearity of holding costs is a crucial assumption for the algorithm to work.

Next, we will review the literature related to recent extensions of lot-sizing problems with polynomial algorithms. Vanderbeck [1998] presents an $O(n^6)$ dynamic programming algorithm for the single-item constant-capacity lot-sizing problem with start-ups, which has a reduced complexity of $O(n^4)$ in the case of Wagner-Whitin costs. The author uses these algorithms in a column generation approach to solve multi-item problems. Hsu [2000] adds the concept of perishable inventory into lot-sizing problems, where inventory holding costs are age-dependent. The main contribution of the paper is an $O(n^4)$ dynamic programming algorithm for the single-item problem with concave production costs, and without production capacities and backlogging. Ahuja and Hochbaum [2006] study the single-item lot-sizing problem with linear costs, i.e., no fixed costs, and with capacities not only on production but also on inventory and backlogging. Due to the small number of arcs, the authors obtain a sparse network, and they develop an $O(nlogn)$ network algorithm to solve this problem.

Recently in Lee et al. [2001], in contrast to fixed times on demands in the classical lot-sizing model, a model that allows the orders to be delivered without a penalty after

an earliest date and before a due date, called a "delivery window", is studied. The authors propose two polynomial-time dynamic programming algorithms for the backlogging and no backlogging cases of single-item, uncapacitated problem with Wagner-Whitin costs. Similarly, Brahimi et al. [2006] investigates the extension of the classical lot-sizing problem with the "production window", i.e., the production of an order is to be accomplished not earlier than the earliest date and before the due date. Proposing and using dynamic programming algorithms for single-item problems, the authors employ Lagrangian heuristics for capacitated, multi-item problem and provide extensive computational results. The similarities of these two recent models with windows have drawn attention. Wolsey [2007] provides detailed polyhedral analysis of these models, and proposes inequalities valid for both models. The author also provides tight extended reformulations and dynamic programming algorithms for some special cases of these problems.

To conclude this section, we will mention two recent papers that might provide extensions for the future. The work of van Hoesel and Wagelmans [2001] is a rare study of approximation algorithms for lot-sizing problems. The authors study the single-item problem with capacities and general costs, and provide dynamic programming algorithms that are polynomial in terms of the inverse of an error factor ϵ. On the other hand, van Hoesel et al. [2005] investigate the optimization problem in a serial supply chain with production capacities, with an approach that is using and extending the single-item lot-sizing model. The authors propose polynomial-time dynamic programming algorithms for different constant capacity and concave cost cases, particularly for two-level supply chain. Their framework can also be extended to general multi-level cases with exponential running time, although the polynomial-time property holds for some

special cases, such as Wagner-Whitin transportation costs.

2.2.2 MIP Studies on Lot-Sizing

For production planning problems, polyhedral analysis has been useful in attempts to solve challenging models and to provide strong lower bounds. Although most of the literature on this area is limited to simple problems such as single-item problems, we should remark that these simple problems are substructures in more complicated problems and hence these techniques can usually be extended.

The first group of these exact methods strengthens the original formulation by adding valid inequalities. The inequalities are generated either using exact polynomial separation algorithm (if there exist such algorithms) or heuristic separation. In either case, the formulations obtained are stronger and hence the Branch&Bound (or Branch&Cut) can work more efficiently.

The milestone study in this area is of Barany et al. [1984a], where the authors study the uncapacitated single-item lot-sizing problem (recall the Wagner-Whitin model from Section 1.2) and propose the following valid inequalities for the problem:

Definition 2.3 *For any $\ell \in [1, NT]$ and $S \subseteq [1, \ell]$, the family of inequalities*

$$\sum_{t \in S} x_t \leq \sum_{t \in S} d_{t,\ell} y_t + s_\ell \tag{2.1}$$

is called the (ℓ, S) ***inequalities****.*

The validity proof is quite straightforward and can be found in the cited paper. Although it seems as if there exist an exponential number of (ℓ, S) inequalities due to all possible subsets of $[1, \ell]$, a simple polynomial separation algorithm exists, as it can be seen in Algorithm 1.

Algorithm 1: (ℓ, S) separation
Input: LP relaxation solution (x^*, y^*, s^*)
Output: Violated (ℓ, S) inequalities
for $\ell = 1$ **to** NT
  Initialize $S \leftarrow \{\,\}$
  for $t = 1$ **to** ℓ
    if ${x^*}_t > d_{t,\ell}{y^*}_t$
      $S \leftarrow S \cup \{t\}$
  if $\sum_{t \in S} {x^*}_t > \sum_{t \in S} d_{t,\ell}{y^*}_t + {s^*}_\ell$
Add the violated (ℓ, S) inequality

The most important characteristic of these inequalities is given in the companion paper of the same authors (Barany et al. [1984b]), as below. Let $P = \{(x, y, s)|(1.2) - (1.5)\}$ and $P_{LS} = \{(x, y, s)|(1.2) - (1.4), 0 \leq y \leq 1^{NT}, (2.1)\}$, i.e., feasible region of the original MIP problem and polyhedron representing the LP relaxation of P with the addition of (ℓ, S) inequalities, respectively.

Proposition 2.4 $P_{LS} = conv(P)$

In words, (ℓ, S) inequalities define the convex hull of the uncapacitated single-item lot-sizing problem. Hence, solving the LP relaxation of the uncapacitated problem with violated (ℓ, S) inequalities added will be sufficient to obtain the optimal solution.

We address these inequalities in detail now as they are used throughout the thesis. As can be seen in Pochet and Wolsey [1991], these inequalities can easily be extended to multi-level lot-sizing problems using echelon demands and stocks defined in Section 1.4. More importantly, these inequalities are not only valid for extensions such as capacities and multiple items but they are also computationally efficient. We provide more detail on these inequalities in the next two chapters.

Pochet [1988] investigates the capacitated single-item problem, particularly the case

when capacities are constant over the horizon. The author proposes a family of facet-defining inequalities and a heuristic separation algorithm. Leung et al. [1989] study the polyhedral structure of the single-item lot-sizing problem with constant capacities. They propose a family of valid inequalities, along with a polynomial separation algorithm. They also provide computational results using the proposed inequalities on multi-item problems. Pochet and Wolsey [1993] propose facet-defining valid inequalities for the special case of constant batches and constant capacities, which can be also applied to constant capacity problem and are stronger than the previous inequalities. However, note that none of these studies could give the full convex hull description of the single-item constant-capacity problem on the original variable space, although the problem can be solved using dynamic programming in polynomial time, as we have seen in the previous section.

There are other insightful polyhedral studies about some special cases of the single-item problem. Pochet and Wolsey [1988] examine the uncapacitated problem with backlogging. In addition to integral reformulations of the problem, they define a family of valid inequalities and a heuristic separation algorithm, which is tested computationally. The recent study of Küçükyavuz and Pochet [2006] accomplishes the full description of the convex hull of single-item problem with backlogging. The general capacitated problem with start-up costs is studied by Constantino [1996]. The author proposes four families of valid inequalities, two of which are only valid for the constant-capacity case, and also describes polynomial separation algorithms. Constantino [1998] investigates the polyhedral characteristics of the lot-sizing problem with constant lower bounds on production quantities. He proposes different families of valid inequalities and polynomial separation algorithms, and also describes the convex hull relaxations of these problems.

The problem with sales instead of demands and lower bounds on inventory is studied by Loparic et al. [2001]. The main contribution of the paper is a family of valid inequalities that provides the full description of the convex hull of the problem. Van Vyve and Ortega [2004] accomplish a complete polyhedral analysis of the uncapacitated problem with fixed charges on inventories. The authors define a family of valid inequalities that is an extension of the (ℓ, S) inequalities, which are sufficient for the full description of the convex hull. Atamtürk and Küçükyavuz [2005] provide a detailed polyhedral study of the lot-sizing problem with inventory bounds and fixed costs on inventory, and conclude with facet-defining inequalities and polynomial separation algorithms to generate these inequalities. The authors also present a tight LP formulation for the special case of Wagner-Whitin costs.

The second group of exact methods use extended reformulations of the model, which basically add new variables to the problem to obtain a strong formulation in the new space. Although extended reformulations have the disadvantage of increasing the problem size because of additional dimensions and constraints, they can be very useful to solve the original problem if the size increase is computationally not challenging. Also, it is possible to generate valid inequalities for the original problem space using the characteristics of an extended reformulation and projection.

The first general extended reformulation for production planning problems is the "Facility Location (FL) Reformulation" of Krarup and Bilde [1977]. For simplicity, consider the single-item lot-sizing problem. The basic idea behind the reformulation is to decompose production in a period based on which period it is produced for, i.e., $x_t = \sum_{t'=t}^{NT} u_{t,t'}$, where $u_{t,t'}$ is the amount produced in period t to satisfy demand of period t'. This allows the problem to be seen as a network with NT possible "facilities"

to open, where a facility represents the setup decision in a period. This reformulation with $O(NT^2)$ variables and $O(NT^2)$ constraints has the significant property that its projection onto original space is integral for the uncapacitated single-item problem, i.e., it suffices to solve the LP relaxation of this reformulation. An improvement of this reformulation is achieved by the "Shortest Path (SP) Reformulation" of Eppen and Martin [1987]. The authors define variables $z_{t,t'}$ to represent the fraction of production produced in period t to satisfy demand from t to t'. This results in a "shortest path" problem with only $O(NT)$ constraints and the same number of variables as in the FL reformulation. This reformulation is equivalent to the FL reformulation in the sense that it provides integral solutions in the extended space. On the other hand, Rardin and Wolsey [1993] define the "Multi-Commodity (MC) Reformulation" for fixed charge network flow problems. The basic idea is similar to the FL reformulation, but this reformulation decomposes the production in addition with respect to which end-item an item is produced for. Although this is a stronger reformulation than the FL reformulation for the multi-level lot-sizing problem, the problem size is computationally inefficient. We will review and study these three reformulations in detail in Chapter 4 for our multi-level production planning problems.

Some polyhedral studies address extended reformulations for specific cases of lot-sizing problem. Pochet and Wolsey [1994] provide a detailed polyhedral study of different single-item lot-sizing problems with Wagner-Whitin costs. In particular, they propose integral extended reformulations for uncapacitated problem with backlogging and uncapacitated problem with start-up costs. They have also projected the reformulation for the backlogging problem into original space and defined valid inequalities with the original variables and a polynomial separation algorithm. Agra and Constantino [1999] is a

brief polyhedral study of the single-item lot-sizing problem with backlogging and fixed start-up costs. The authors investigate the properties of the optimal solutions, and they provide step-by-step extended reformulations that are integral, with some extensions of Pochet and Wolsey [1994]. A recent polyhedral study on the the multi-item discrete lot-sizing problem is accomplished by Miller and Wolsey [2003], where only one item can be produced in a period and production of an item is either 0 or a predefined constant amount for that item. The authors propose extended reformulations for some cases, particularly integral reformulations for the cases with backlogging and initial inventory variables. In addition to these studies, we also mention two recent studies, Belvaux and Wolsey [2001] and Wolsey [2002], for considering reformulation and modeling issues for solving practical-size production planning problems. Both papers demonstrate the use of MIP for tackling these problems, after efficient reformulation techniques for different problem structures with preprocessing are applied.

There have been also few polyhedral studies on multi-item problems. Pochet and Wolsey [1991] investigate both single-level and multi-level problems, and provide extensions of some previous results for single-item problems, such as the (ℓ, S) inequalities of Barany et al. [1984a], with extensive computational results. In a similar fashion, Belvaux and Wolsey [2000] extend some of the single-item problem results to big bucket problems and use them in a specialized Branch&Cut system. Although these extensions provide important insights, the current literature on strong formulations for big bucket problems is limited. An exception is the study of multi-item problems with shared resources and setup times in the companion papers Miller et al. [2000] and Miller et al. [2003]. The latter paper addresses a detailed polyhedral analysis of a single-period relaxation of the problem, with new families of facet-defining inequalities and characteristics of the

convex hull, and extends these to big bucket problems. The former paper investigates the practicality of these new inequalities, proposing separation heuristics and applying these in Branch&Cut to provide computational results.

Before concluding this section, we will mention two studies that provide some background and motivation for our work presented in later chapters. Jans and Degraeve [2004] study multi-item single-level problems and provide theoretical comparisons between lower bounds based on Lagrangian relaxation, where problems are decomposed either item-based or time-based. We will review some of these techniques in Chapter 4. Another recent result is the polyhedral study of Atamtürk and Muñoz [2004] on the single-item problem with general capacities. The authors approach the problem by reformulating it as a bottleneck flow problem and define cover inequalities for bottlenecks and lift them to obtain facet-defining inequalities. These inequalities are compared with the previous results in uncapacitated and constant capacity cases, and also a polynomial separation algorithm for the constant capacity problem is given. As we will address more in detail, this study is one of the major motivations for the work presented in Chapter 5.

2.2.3 Heuristics

Heuristics are techniques designed to provide reasonably good solutions in acceptable times for practical problems, without a solution guarantee. In general, heuristics do not provide any solution quality either. Heuristics are employed for a variety of challenging problems from artificial intelligence to transportation systems, and in the lack of exact methods, they are many times the only way to attain some problems.

Real-world MIP problems are often computationally very difficult, even with the recent advances in computing, hence researchers have often used heuristic approaches to tackle them. In this section, we will briefly review some general MIP heuristics, and then focus on production planning heuristics. We can classify heuristics as "Constructive Heuristics" (i.e., start with no solution and try to find one) and "Improvement Heuristics" (i.e., start with a solution and try to improve it).

Even though there are many problem-specific heuristics in the MIP literature, there are comparatively few general MIP heuristics. Two examples for simple constructive heuristics are "LP-and-fix", where integral values from the LP relaxation are fixed, and "Relax-and-Fix", which relaxes some integer variables to continuous and fixes the integer-defined variables in each iteration. We will provide more detail on these heuristics in Chapter 3, as they are key parts of the solution procedure proposed. Recent general constructive MIP heuristics include the polyhedra-based heuristic of Balas et al. [2001] and "Feasibility Pump" of Fischetti et al. [2005]. The former heuristic is designed for solving pure 0-1 problems, and it works by guessing a direction in the polyhedron and enumerating a number of facets in that direction. The latter heuristic provides a simple framework designed for finding initial feasible solutions for very hard problems, where a distance function between the LP relaxation solution of an MIP problem and its rounding is minimized. As recent improvement heuristics, we will remark "Local Branching" of Fischetti and Lodi [2003], which uses the idea of branching on the neighborhoods of the current MIP solution, and "Relaxation Induced Neighborhood Search" (RINS) of Danna et al. [2005], which searches the neighborhood between the LP relaxation solution and the current MIP solution. For a general review and details of MIP heuristics, including these mentioned, refer to Pochet and Wolsey [2006], pp. 107-113.

Problem-specific heuristic algorithms have been proposed and used for many production planning and lot-sizing problems, including complex and realistic systems such as the multi-level, capacitated problems studied in this thesis. One of the earliest and classical heuristics is the "Silver-Meal Heuristic" (Silver and Meal [1973]), which solves the uncapacitated single-item problem. The heuristic calculates per-period average cost for producing a sequence of periods and sets up production in a period which observes an increase in the average cost. Although exact solution of this problem was available with the dynamic programming algorithm of Wagner and Whitin [1958], limited computing abilities was the motivation for this work. For an excellent review of earlier lot-sizing heuristics with computational comparisons, see Maes and van Wassenhove [1986].

Many heuristic approaches use decomposition ideas, and can be grouped as follows:

1) *Lagrangian-based decomposition*: Trigeiro et al. [1989] study the multi-item problem with a single machine that is shared by all items. Using Lagrangian relaxation on the capacity constraint, the problem is decomposed into uncapacitated single-item problems, which are solved by dynamic programming. A smoothing heuristic is applied to generate feasible production plans for the original problem using the results of single-item problems. On the other hand, Tempelmeier and Derstroff [1996] decomposes a multi-level problem into single-level uncapacitated problems. Using the solutions of these subproblems, the workload on machines are calculated and the Lagrange multipliers are updated. The heuristic generates both lower and upper bounds for the original problem.

2) *Coefficient modification*: Katok et al. [1998] propose a two-stage heuristic for the multi-level big bucket production planning problems. Coefficient modification is applied by allocating setup times to variable times for finding an initial solution, and

restricted LP relaxations of the second stage try to improve the initial solution. Van Vyve and Pochet [2004] propose a coefficient modification based heuristic algorithm to be used within branch-and-cut, which is flexible enough to be extended to network design problems as well. The heuristic is based on modifying capacities to smooth the relation between the linear and binary variables in LP relaxation. Once integral values for all setup variables are obtained, the heuristic solves only an LP problem.

3) *Forward scheme and Relax-and-fix*: Afentakis and Gavish [1986] work on complicated BOM (Bills of Material) structures, by reformulating general structure problems as assembly structure problems. The authors use a forward scheme to obtain feasible solutions while using Lagrangian relaxation for lower bounds. Belvaux and Wolsey [2000] solve practical lot-sizing problems using a special branch-and-cut system that employs relax-and-fix heuristics for finding feasible solutions. The authors also provide extensive computational results. Federgruen and Tzur [1999] proposes a time-partitioning heuristic with non-overlapping subproblems for a lot-sizing problem with a distribution network where each subproblem is solved to optimality sequentially. Federgruen et al. [2002] use a relax-and-fix heuristic with both overlapping and non-overlapping subproblems, where the whole horizon is considered for all subproblems. In a similar fashion, Stadtler [2003] uses the idea of "time windows", which represent the subproblems in a relax-and-fix framework. The decisions for periods after a time window are not considered, although demand in future periods are taken care of using some cost modification scheme called as "bonuses". Suerie and Stadtler [2003] extend this framework to the big bucket lot-sizing problems with linked setups, i.e., the possibility of carrying the setup of one item from one period to the next. Federgruen and Meissner [2004] analyze these recent heuristics from the optimality and complexity perspective. As relax-and-fix is an

important part of the framework we propose in Chapter 3, we will provide more detail on some of these heuristics and comparisons with our framework.

4) *Local search*: Gopalakrishnan et al. [2001] investigate the capacitated big bucket problem with setup carryovers, which includes both classical lot-sizing problem and sequencing problem. Search strategies such as dynamic tabu list and adaptive memory are used to generate feasible solutions. Another good example is the neighborhood search heuristic of Simpson and Erenguc [2005], which finds an initial feasible solution and improves it using local search.

Even though we are investigating the production planning problem with a horizon of length NT, we will briefly remark some recent work on the so-called "end-effect", i.e., the possibility that an optimal plan for NT periods might have $s_{NT} = 0$ and it is not optimal or even feasible for T periods, where $T > NT$. The main reason why the problem with T periods is not solved directly instead of the problem with NT periods is that usually the demand forecast after a period NT is not reliable or available. The reason we make this remark in this section is that these methods are not exact methods and they fit heuristic approaches. Stadtler [2000] uses forecasted demand for periods outside the horizon NT to recalculate the cost coefficients of the problem, so that a setup decision is made close to the end of the horizon. This approach is also applied in his relax-and-fix approach mentioned before (Stadtler [2003]). On the other hand, Fisher et al. [2001] have the perspective of assigning a value for the end-inventory to tackle this problem. This study is mainly based on the EOQ model with constant-demand over the horizon. Finally, van den Heuvel and Wagelmans [2005] provide a computational analysis for the comparison of these two end-effect methods with their simple approach based on an extension of Wagner-Whitin algorithm.

2.2.4 General Review on MIP

This section is not intended to give a broad review of all MIP techniques, but in general to give a brief review of some techniques that are used in this thesis. First, we will provide for interested reader some general references related to MIP. As these fields are related closely to MIP, see Schrijver [2003] for a complete resource on combinatorial optimization and Ahuja et al. [1993] for network flow techniques. Finally, for the fundamentals of the MIP such as polyhedral analysis and numerous examples, refer to Nemhauser and Wolsey [1999], Wolsey [1998] and Schrijver [1986].

After these general references, we will briefly remark two areas that are extensively studied in polyhedral analysis and are related to lot-sizing problems as well as many other general MIP problems. The first one is fixed charge network problems, which lot-sizing problems are a special type of. The milestone study in this area is of Padberg et al. [1985], who investigate the polyhedral structure of these problems and propose the "flow cover inequalities" that have been proven useful in many MIP applications. Another area is continuous knapsack problems, which are commonly a basic structure involved in many MIP problems. A recent and significant work in this area is that of Marchand and Wolsey [1999], who investigated the case of binary variables and defined some facet-defining valid inequalities.

General valid inequalities for MIP problems have always been an attractive research area. The Chvátal-Gomory (CG) cuts (Gomory [1960], Chvátal [1973]) and the split cuts (Balas [1979]) are widely studied examples. Recently, there have been a big body of research and promising results on the "closures" of such general cutting planes and

some particular polyhedrons. "Closure" in this perspective can be defined as the smallest possible polyhedron that includes all the valid inequalities of a type. Clearly, an inequality necessary to define the closure is valid for the set.

Cook et al. [1990] and Nemhauser and Wolsey [1990] are earlier studies on Chvátal and split closures and how to generate inequalities for general MIP problems. Letchford [2001] provides theoretical reasoning on the success of disjunctive cuts. Caprara and Letchford [2003] provide complexity results on the separation problems of many elementary closures, including split and MIR (Mixed Integer Rounding) closures. Andersen et al. [2005] and Dash and Günlük [2006] study split cuts and MIR closure, respectively, and Bienstock and Zuckerberg [2006] investigate approximate closure of covering problems. Fischetti and Lodi [2005] and Balas and Saxena [2007] provide separation algorithms and promising computational results on optimizing over rank-1 Chvátal closure and rank-1 split closure, respectively. Cornuéjols and Li [2001] and Balas and Perregaard [2003] are excellent studies about the relations and equivalencies between different elementary closures, and Marchand et al. [1999] provides a detailed survey on cutting planes for general and special-structured MIP problems. With the motivation of recent results on closures, we study problem-specific convex hull closures in Chapter 5.

Chapter 3

Generating Feasible Solutions

As discussed in Chapter 2, there have been different approaches to solve production planning problems. The early MRP approach, while accounting for bill-of-material structures, follows decision rules that are too simplistic even to achieve feasible plans consistently. The more advanced systems of MRP-II and ERP fail to account accurately for capacity, even with their broader perspectives on manufacturing companies as a whole. Some special cases of production planning problems can be solved using dynamic programming exactly and in polynomial time. However, knowing that even the capacitated version of the single-item lot-sizing problem is $\mathcal{NP}$-hard, we should not expect to be able to solve big bucket problems using only exact methods.

On the other hand, mathematical programming results have been in general limited for the realistic production planning problems that involve big bucket capacities, although they have provided some valuable insights into the problem structure and dynamics. The advantage of these approaches is that they run on increasingly powerful computers with advanced commercial and academic solvers that can handle very big problems. Therefore using exact methods is becoming more attractive. However, even though some problems with hundreds of thousands of variables and constraints can now be solved optimally using exact methods, there are many MIP problems in the literature that are much smaller in size and still result in large duality gaps, e.g. see MIPLIB

[2003] for some hard test problems. For some types of MIP problems, including production planning problems, heuristics have in general been useful for practical purposes to generate high quality solutions in acceptable times. However, these methods mainly lack a solution guarantee and can end up in a low quality solution or no solution at all. A good example is given by Van Vyve and Pochet [2004] for the failure of simple MIP heuristics that use rounding.

In general, the tradeoff in all these different methodologies is between acceptable solution times and solution quality. Using an exact methodology, such as Branch&Bound on an MIP problem, gives in theory the optimal solution, but running time might be too long. On the other hand, as we mentioned, a heuristic can solve a practical problem in a short time, without a guarantee on the quality. This tradeoff suggests that a combination of different methods might be efficient both in time and solution quality.

In this chapter, we propose a simple heuristic framework for production planning problems in general. However, note that this is not simply a pure heuristic approach, as exact MIP methods are also employed in the process. More specifically, the proposed framework combines different mathematical programming-based heuristics, namely LP-and-fix and relax-and-fix, after obtaining a strengthened formulation of the problem. One of the main advantages of the framework is that it has the ability to generate multiple solutions during the whole process, which gives flexibility to a decision maker. In addition, using mathematical programming-based methodology allows it to provide lower bounds at least as strong as other known bounds, and in many computational cases, significantly stronger lower bounds. One of the significant aspects of the proposed framework is its simplicity, which also allows it to be implemented in a mathematical modeling language, namely Mosel of Xpress-MP. Finally, the approach has the generality

advantage that it can be used for different types of production planning problems.

In the next section, we will briefly discuss how to obtain an initial strong formulation for better results. Then, we will briefly review MIP heuristics of interest that are employed in our framework. Section 3.3 is devoted to the description of the framework in detail. Extensive computational results and concluding remarks will follow.

3.1 Strengthening the Formulation

In order to start the heuristic with the best possible formulation, we investigate the possibility of extending the basic formulation of Section 1.4. When doing so, it is important to keep in mind the tradeoff between solution quality and solution times we mentioned in the introduction of this chapter. More specifically, we are interested in a formulation that is both strong, probably not the strongest, and computationally tractable. We have mentioned briefly in Section 2.2.2 some methods to obtain strong formulations for production planning problems, which we will investigate in detail in the next chapter for lower bound aspect of these problems. In some of the arguments here, we will refer to next chapter for details.

We particularly consider several ways to strengthen formulations. First, we consider the (ℓ, S) inequalities (2.1) of Barany et al. [1984a] we previously discussed. These inequalities are generalized by Pochet and Wolsey [1991] to multi-level problems using echelon demands and stocks, as follows:

$$\sum_{t \in S} x_t^i \leq \sum_{t \in S} D_{t,\ell}^i y_t^i + E_\ell^i \qquad \ell \in [1, NT], i \in [1, NI], S \subseteq [1, \ell] \tag{3.1}$$

Note that because of multiple levels, (3.1) require the use of echelon quantities. The validity of these inequalities follow the same logic as the original single-item inequalities.

Although there are an exponential number of (ℓ, S) inequalities for each item, as it is the case for single-item case, a simple extension of the polynomial separation algorithm of Algorithm 1 exists, see Algorithm 2. Recall that these inequalities define the convex hull of the uncapacitated single-item problem (Barany et al. [1984b]).

Algorithm 2: (ℓ, S) separation (multi-item)
Input: LP relaxation solution (x^*, y^*, E^*)
Output: Violated (ℓ, S) inequalities
for i=1 **to** NI
  for $\ell = 1$ **to** NT
    Initialize $S \leftarrow \{\,\}$
    for t=1 **to** ℓ
      if ${x^*}^i_t > D^i_{t,\ell} {y^*}^i_t$
        $S \leftarrow S \cup \{t\}$
    if $\sum_{t \in S} {x^*}^i_t > \sum_{t \in S} D^i_{t,\ell} {y^*}^i_t + {E^*}^i_\ell$
Add the violated (ℓ, S) inequality

In a cutting plane approach, this separation algorithm will be used to generate violated inequalities after each time solver finds a new LP relaxation solution. This allows a practicality that inequalities can be deleted if they are not active anymore, and thus preventing computational problems related to a big formulation.

One important remark is that (ℓ, S) inequalities are valid for most of the extensions that can be applied to the basic formulation of Section 1.4. For example, capacities do not affect the validity of them. Similarly, in case of overtime, the original inequalities are still valid. In these cases, violated (ℓ, S) inequalities will be generated using the same separation algorithm. An interesting case is when backordering is allowed. In that case, a slightly modified family of (ℓ, S) inequalities can be defined, as follows:

$$\sum_{t \in S} x^i_t \leq \sum_{t \in S} (D^i_{t,\ell} y^i_t + \sum_{j \in endp} r^{ij} b^j_{t-1}) + E^i_\ell \qquad \ell \in [1, NT], i \in [1, NI], S \subseteq [1, \ell]$$

The backordering variables are represented with b_t^i for each period t and each end-item i. Here, one can observe that the separation algorithm for these inequalities has the same logic as Algorithm 2.

Next, we discuss the facility location reformulation of Krarup and Bilde [1977], mentioned before in Section 2.2.2 and originally proposed for single-item problems. The reformulation uses new variables $u_{t,t'}^i$, which indicate the amount of item i that is produced in period t to satisfy demand in period t'. Again, since we are dealing with multi-level problems, echelon demands and stocks will be used to apply this reformulation efficiently. The following constraints need to be added into the basic formulation:

$$u_{t,t'}^i \leq D_{t'}^i y_t^i \qquad t \in [1, NT], t' \in [t, NT], i \in [1, NI] \tag{3.2}$$

$$\sum_{t=1}^{t'} u_{t,t'}^i = D_{t'}^i \qquad t' \in [1, NT], i \in [1, NI] \tag{3.3}$$

$$x_{t'}^i \geq \sum_{t=t'}^{NT} u_{t',t}^i \qquad t' \in [1, NT], i \in [1, NI] \tag{3.4}$$

$$u \geq 0 \tag{3.5}$$

(3.2) provides a tighter relation between the new variables and the binary setup variables, compared to the relation between the original production variables and binary variables provided by (1.10). Obviously, this formulation has $O(NT^2)$ variables, and $O(NT^2NI)$ constraints are added to the problem, hence the size gets bigger. Recall that the projection of the facility location reformulation onto the space of original variables gives the convex hull of the uncapacitated single-item lot-sizing problem (Krarup and Bilde [1977]).

Although it is a known fact that both of these methods are equivalent and provide integral formulations in the uncapacitated single-item case, we rather need a relation

for the general multi-level problem. In the next chapter, we investigate this relation, and in Proposition 4.2, we show that these are also equivalent in the multi-level case, i.e., the LP relaxations of both formulations are equal to each other. Hence, from the strength perspective, we are indifferent between these methods. On the other hand, clearly facility location reformulation increases the problem size significantly, whereas, as we will see in the next chapter in more detail, (ℓ, S) inequalities are computationally efficient. Hence, the choice between these two methods will be (ℓ, S) inequalities.

There exist stronger inequalities and reformulations such as the multi-commodity reformulation of Rardin and Wolsey [1993], or valid inequalities generated for the multi-item single-period sub-models by Miller et al. [2000] and Miller et al. [2003]. However, as it will be presented in the next chapter, we have found that using (ℓ, S) inequalities alone seems to be the most effective way to strengthen the formulation without making the problem computationally inefficient. Therefore, we will use (ℓ, S) inequalities for the initial strong formulation in this heuristic framework.

3.2 MIP Heuristics

Before explaining our framework in detail in the next section, we discuss here two heuristic techniques that we will use in our proposed framework that have been briefly mentioned and explained in Section 2.2.3. For more details on general MIP heuristics such as "rounding" and "diving", please refer to Pochet and Wolsey [2006].

3.2.1 LP-and-Fix

LP-and-fix is a simple technique, closely related to diving, and works as follows: First, we solve the LP relaxation (LPR) of an MIP problem. Then, we check all the integer variables in the LPR solution and fix those having integral values. Finally, the restricted MIP with fixed variables is re-solved, with the hope of finding a solution to the original problem quickly. This technique is employed in our heuristic framework both to provide an initial solution to be used as a cutoff value, and also throughout the main algorithm to generate multiple production plans and to improve the best solution and hence the cutoff value.

3.2.2 Relax-and-Fix

Relax-and-fix is a heuristic method for problems with a special structure. More specifically, suppose all integer variables in an MIP problem can be partitioned into n sets $\mathcal{S}_i$, $i = 1, ..., n$, where the decisions on the integer variables in $\mathcal{S}_i$ are more important than the decisions on the integer variables in $\mathcal{S}_j$ for any $i < j$. Then, the partial LP relaxation of the MIP problem is solved, where only the integer variables in $\mathcal{S}_1$ are enforced to be integer, and all or some of the variables in $\mathcal{S}_1$ are fixed once this problem is solved. The scheme works the same way for all $i \in [2, n]$ in the increasing order of i. Lot-sizing is well-suited for such a method since decisions made earlier in the horizon are more important then later ones since early decisions affect later periods. In this section, we discuss two previously developed heuristic methods that employ the "relax-and-fix" idea.

The first production planning tool we will mention that uses relax-and-fix is $bc-$

prod, proposed by Belvaux and Wolsey [2000], which is a specialized branch-and-cut system for lot-sizing problems. The system first generates cutting planes, both default Xpress cuts and problem specific cuts, and then the relax-and-fix idea is applied using "time windows", horizons under consideration, for which the length of the window is predefined. The basic description of the "time windows" idea is as follows: Except for the variables in the periods of the predefined time window, relax all the binary variables to be continuous, solve the problem, and using the solution obtained, fix the binary variables in the window. The next window is then processed in the same manner.

Another heuristic algorithm based on relax-and-fix is proposed by Stadtler [2003], which we will refer to as "SH" (Stadtler's Heuristic) from now on. This approach uses time windows that overlap for better quality results. In such an approach, fixing variables in a window will occur only for the periods that do not overlap with the next window. Also, SH allows a period to be relaxed to continuous inside of the window, as well. Different than the previous approach, this scheme is not solving a problem considering decisions on later periods in each window because of using "bonuses" according to Stadtler [2000]. This bonus approach makes it more attractive to produce later in a time window to prevent stockouts. In this framework, each subproblem is formulated and strengthened separately. Figure 3.1 summarizes the idea of "time windows" in SH. A recent paper of Federgruen et al. [2002] also uses a relax-and-fix approach in a similar fashion to the time windows.

The main advantage of $bc - prod$ is that strong formulations provide good lower bounds for the problem, hence solution quality can be proven without extra computation. On the other hand, $bc - prod$ does not have any overlapping windows, therefore it can miss considering the effect of future periods' setup decisions. Also, it can require long

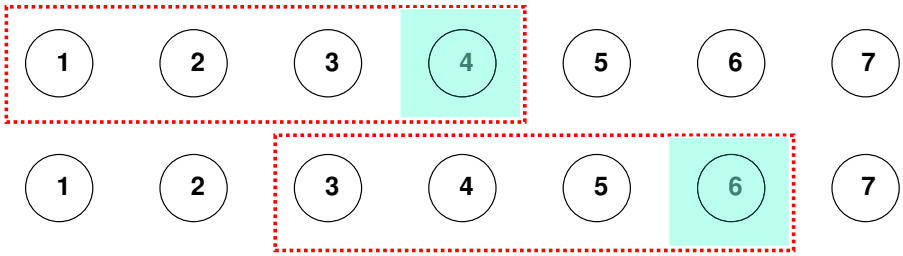

Figure 3.1: The first 2 "time windows" of SH, with window length of 4 periods, 2 periods overlapping and 1 period relaxed to be continuous

computational times because of long windows, large numbers of inequalities added to the formulation and the fact that it solves the subproblem of each window to optimality. Even though SH provides comparatively good results for hard test instances, it should be noted that the implementation is not straightforward because of complex calculations, such as the bonuses. Moreover, as in many other heuristic frameworks, it does not generate lower bounds. In order to determine solution quality, lower bounds need to be generated separately.

3.3 The Heuristic Framework

As in the last section, we use "time windows", or simply "windows", to refer to the time interval in which binary variables are forced to take binary values. For later periods, they are relaxed to be continuous. The window has a "length", or number of periods. The first part of the window which does not overlap with the next window is called the "fixing interval", since this is where the binary variables will be fixed once the subproblem is solved. Note that in our approach the formulation is strengthened for the *entire* horizon, and not just within the window, as in SH.

We also use LP-and-fix idea to find feasible solutions for the original problem throughout the framework in order to provide multiple solutions and upper bounds (cutoff values) for later windows. Our computational experience is that using cutoff values significantly improves the solution process of many windows. Another difference from SH is that we use the objective function defined in Section 2 in each window. This is simpler to implement, and also helps us to gauge the effect of future setup decisions

```
Input: Lot-sizing problem
Output: Multiple feasible solutions and lower bound for the problem
(ℓ, S) separation (Algorithm 2)
Initialize upper bound and cutoff bound using LP-and-fix
for i=1 to numwin
   Relax the y variables after the window to continuous
   Solve the sub-problem
   Fix the y variables in the fixing interval
   if extra time
      Enforce binary restriction on y variables after the window
      Fix all y variables having an integral value
      Solve the partially fixed MIP
      If solution found, reset cutoff value
      Unfix all y variables after the window
 if extra time at the end
   Improve the solution
```

Figure 3.2: The heuristic framework

The pseudocode of the general framework can be seen in Figure 3.2, and a complete sample file of this heuristic for one of the test sets, written in Mosel, is included in Appendix A. We can describe the framework in words as follows: After generating violated (ℓ, S) inequalities and deleting inactive ones, the LP relaxation solution of the original problem is used to start LP-and-fix to find a first feasible solution for the original problem. This solution will be used to initialize the cutoff value. After initialization,

we start with the first window, i.e., solve the problem where all the binary variables are relaxed to continuous for any period beyond the scope of the window. If extra time is available, i.e., relax-and-fix is complete before the pre-set time limit, LP-and-fix is used in order to generate a feasible solution for the original problem. At the end of LP-and-fix, all the binary variables fixed beyond the window will be unfixed before processing the next window. The same procedure will be repeated for future windows.

Here, note that the heuristic provides a lower bound to the original problem in the first window. More specifically, we are solving a partial LP relaxation of the original formulation already strengthened with (ℓ, S) inequalities, where only the integer variables in the first window are enforced to be integer. The computational results in the next section show that this lower bound is often competitive with the best known methods.

In the interest of speed, we seek to avoid running a window for a long time to find the optimal solution but obtain a reasonable solution in limited time and move on to the next window. Therefore we set a maximum time for each window. This issue is discussed in more detail later.

One important question to ask for both relax-and-fix and LP-and-fix is "which variables to fix". One basic concern for relax-and-fix is that the more we fix in a window, the higher probability we have for infeasibility in later windows. Hence, in our early tests, we tried to fix only 0's and only 1's, but no significant difference from fixing both 0's and 1's has been observed. On the other hand, for LP-and-fix, fixing all 0's and 1's appears to create problems occasionally, such as infeasibility and poor-quality solutions, hence only 1's are fixed in that part of the framework.

The heuristic framework depends on many parameters and options. These include the length of a window, whether we have to overlap consecutive windows, and how long

the overlap should be. It is obvious that the shorter the length of the window, the easier it is to solve the related subproblem. However, this can deteriorate the solution quality since decisions become more myopic and since the number of windows grows and hence time allocated to a window decreases as well. It is best to have a window length which neither takes too much time nor finds too poor solutions. When consecutive windows do not overlap, we deal with fewer windows, but this does not facilitate the consideration of setup decisions from later periods and therefore can result in bad solutions. After experimentation, we use the window length of 3, with an overlap of 1 period between windows.

We have found it advisable to allocate more time to earlier windows than to later windows because the problems are bigger in size and hence harder to solve, and also to employ LP-and-fix as often as possible in earlier periods in order to generate good solutions. After assigning a total time for a problem, this allocation process is achieved inside of the algorithm. We divided windows into four sets of the same size.Thus, if we have a total of 12 windows, the first three windows are in the first set, the next three windows are in the second set and so forth. Then we assign 1.75, 1.25, 0.75 and 0.25 times of the average time per window to those groups respectively. Thus, if we have a total of 180 seconds to process 12 windows, we allocate $\frac{180}{12} * 1.75 = 26.25$ seconds to each of the first three windows, $\frac{180}{12} * 1.25 = 18.75$ seconds to each of the next three windows, and so forth.

We also set a relative gap parameter so that if we obtain a solution with a duality gap less than this preset amount, the heuristic stops trying to complete the time assigned to this window and moves on either to the next window or starts LP-and-fix with the extra time if it is sufficient for that purpose. Our computational experience suggests

that choosing gap values that are too small neither guarantees a better solution at the end nor allows LP-and-fix sufficient time to generate good solutions. In addition to these major parameters, there are some minor parameters used in the heuristic framework in order to increase its efficiency, such as minimum and maximum times to employ LP-and-fix. Also, note that we start employing LP-and-fix in the windows not after the first window but after the second window, because computational experience suggests that early LP-and-fix procedures take much more time to find a solution than later LP-and-fix procedures.

3.4 Computational Results

In order to provide diversified results and test the generality of our heuristic approach, we used the following various test sets from the literature for our computations:

- Test instances generated by Tempelmeier and Derstroff [1996] and Stadtler [2003]: These include overtime variables. Sets A+ and B+ involve problems with 10 items, 24 periods and 3 machines, and sets C and D involve problems with 40 items, 16 periods and 6 machines. Sets B+ and D include setup times. We chose the hardest instances of each data set for our computations, i.e., for each data set, we picked 10 assembly and 10 general instances with the highest duality gaps according to the results of Stadtler [2003]. This test set will be referred as "TDS" in the remainder of the thesis.

- Multi-level LOTSIZELIB [1999] library instances: These include single-machine problems with big bucket capacities. Backlogging is allowed. The problems vary

between 40 item, single end-item problems and 15 item, 3 end-item problems, with both assembly and general BOM structures. All problems have 12 periods.

- Multi-LSB instances: We have generated 4 sets of test problems based on the problem described in the paper of Simpson and Erenguc [2005], each set having 30 instances with low, medium and high variability of demand. From now on, we will call these sets SET1, SET2, SET3, and SET4. These instances are different from the previous problems in that they take component commonality into consideration and hence consider setup variables for each family so that setup times are defined for each family of items instead of for each item. While keeping the original data such as BOM structures and holding costs, we removed the setup costs and added backlogging variables into the problem to obtain problems with a different nature from the other problems. The backlogging costs are set to the double of inventory holding costs for the first two sets, and 10 times the inventory holding costs for the last two sets. Except the problems in SET2, which considers a horizon of 24 periods, all the instances have 16 periods. The main difference between these three sets is that they have different resource utilization factors, and the rest of the data remain the same. All instances consider 78 items and have an assembly structure, and all instances allow backlogging in the last period. For more details about the instances, see Multi-LSB [2006] and Simpson and Erenguc [2005].

All the test instances were run on a PC with an Intel Pentium 4 2.53 GB processor and 1 GB of RAM. All the formulations and the heuristic algorithm were implemented using only Mosel, which is a high-level algebraic modeling language that is intuitive to use. In all the computational experiments, the Xpress-MP 2004C package and Mosel

version 1.4.1 are used.

Different test sets will be discussed separately since we are using different benchmarks for each set, and different characteristics of each set make it more interesting to analyze each separately. We assigned a total time of 180 seconds for each instance of A+, B+, LOTSIZELIB, SET1 and SET2, and 500 seconds for each C, D, SET3 and SET4 instance because of the problem complexity. Also, a 0.5% duality gap in a window is successful for A+, B+, LOTSIZELIB, SET1 and SET2 instances, duality gaps of 2.5% for later windows and 4% for earlier windows for the C, D and SE4 instances, and finally 10% duality gap for SE3 instances. These percentages are set intuitively from the problem difficulties, e.g., if the default problem solved by Xpress augmented with (ℓ, S) inequalities results with a 10% gap and if there are 10 windows, 1% will be assigned to each window.

Table 3.1: Summary of results for TDS instances

Test Set	Best Solution found by			Average duality gap		
	SH	Xpress	Heuristic	SH	Xpress	Heuristic
A+	4	-	16	29.28%	25.28%	24.47%
B+	11	1	7	29.27%	34.21%	29.86%
C	11	-	9	30.69%	35.40%	28.33%
D	7	-	3	215.96%	364.57%	198.54%

For the TDS instances, we ran an executable of SH on our computer to provide fair comparisons. Another benchmark used for these instances is default Xpress augmented with the generation of (ℓ, S) inequalities. Inactive inequalities are deleted at the root node. The main results are summarized in Table 3.1: The first three columns show for how many instances of a particular test set any of these three methods found the best solution, and the last three columns show the average duality gaps of each method at each set. Note that since SH does not generate lower bounds, the (ℓ, S) lower bound

obtained at the root node is used for that purpose.

Table 3.2: Pairwise comparisons of the heuristics with benchmarks for TDS instances

Test Set	SH vs. Heuristic			Xpress vs. Heuristic			
	# S	# H	UB	# X	# H	UB	LB
A+	4	16	2.26%	-	20	5.77%	5.15%
B+	12	7	-1.63%	4	15	5.73%	4.08%
C	11	9	-0.22%	-	20	6.91%	1.26%
D	7	3	2.45%	-	10	74.94%	1.12%

Table 3.2 summarizes pairwise comparisons between our heuristic and the two benchmarks. The first two columns show how many times SH and our heuristic provide a better solution, respectively, and the next column shows the average difference between our heuristic's and SH's upper bounds, respectively, calculated as (SH Bound - Heuristic Bound)/ Heuristic Bound. The last four columns are prepared in the same fashion, for the comparison between our heuristic and the default Xpress, and the last two columns are calculated as (Xpress Bound - Heuristic Bound)/ Heuristic Bound. It is easy to observe that our heuristic generates good solutions compared to SH, with a good lower bound guarantee. On the other hand, as expected, default Xpress generates better lower bounds than our heuristic, but it is also notable that the harder the problem is, the better the lower bounds the heuristic generates compare to those generated by Xpress. Also note that the proposed heuristic method improves the lower bounds obtained at the root node with (ℓ, S) inequalities on average by 1.11%. For detailed results for all the instances of these test sets, please refer to Appendix B.

For the LOTSIZELIB instances, on the other hand, default Xpress with (ℓ, S) inequalities has been the primary benchmark. The reason that we did not test SH on these problems is that it is not designed for problems with backlogging and significant

Table 3.3: Computational results on LOTSIZELIB instances

Instance	Optimal Solution	Xpress			Heuristic		
		UB	LB	Time	UB	LB	Time
LLIB B	3,965	3,965	3,957	180	3,973	3,915	23
LLIB C*	2,083	2,125	2,046	180	2,113	2,067	73
LLIB D*	6,482	6,852	4,723	180	7,002	4,714	180
LLIB E*	2,801	3,099	2,537	180	2,952	2,416	152
LLIB F*	2,429	2,458	2,180	180	2,429	2,099	99

* indicates instances that could not be solved to optimality by default Xpress in 900 seconds.

modification would be necessary. Also note that these instances already have known optimal solutions, hence the solution quality is more transparent. Table 3.3 shows results of the LOTSIZELIB instances: The first column indicates the optimal solution, then the next three columns show respectively upper and lower bounds obtained by default Xpress and in how many seconds, and finally the last three columns present results of our heuristic. As expected, default Xpress generates better lower bounds generally, however, the heuristic provides comparably good solutions, better solutions in 3 out of 5 instances, and in less time.

Finally, we analyze the 4 sets of Multi-LSB. These instances vary from easy to very hard problems, as can be seen in detailed results in Appendix B. Here, we summarize the results in Table 4 separately for each different set and also separately for different problem difficulties, as follows: Problems with a duality gap less than 10% after the default time with Xpress augmented with (ℓ, S) inequalities are classified as "easy", problems with a duality gap more than 50% are called "hard", and the rest of the problems are "moderate".

Table 3.4 is organized as follows: For "Upper Bounds" and "Lower Bounds", "#X" and "#H" indicate how many times in that set Xpress and the proposed heuristic found

Table 3.4: Summary of results for Multi-LSB instances

Test Set	Class	Upper Bounds		Lower Bounds		Ave. Difference	
		# X	# H	# X	# H	UB	LB
SET1	Easy	2	9	10	2	0.62%	2.43%
	Moderate	0	18	5	13	7.33%	-0.16%
SET2	Easy	3	13	14	3	0.94%	0.90%
	Moderate	0	13	4	8	5.81%	-0.15%
SET3	Hard	0	30	15	15	28.48%	-0.10%
SET4	Easy	2	1	3	0	-1.02%	6.34%
	Moderate	1	9	9	1	2.78%	5.84%
	Hard	0	17	4	13	9.35%	-3.88%

*"Easy" classes of SET1 and SET2, and "hard" class of SET3 has each one instance where both methods found the exactly same solution, and one moderate instance of SET2 had equal lower bounds for both methods, which are not included in the numbers above.

a better bound, respectively. Note that cases in which both methods generate exactly the same bound are not included in these numbers. Finally, the last two columns indicate the average difference between the bounds in percentage, and these are calculated by the expression (Xpress Bound - Heuristic Bound)/ Heuristic Bound. As the summary indicates, the heuristic finds generally better solutions for the problem than Xpress does. Similar to previous results, Xpress default lower bounds are often better than the heuristic lower bounds, although it is interesting to observe that in SET3, i.e., the hardest test instances of all our computations, and in the hard instances of SET4, the heuristic generates lower bounds that are competitive with those of Xpress. This seems to be due to the fact that, for the hardest problems, the problems' difficulty prevents Xpress from making more than minuscule improvements to the lower bound during the branch-and-bound process.

To summarize this section, the proposed heuristic generates good solutions for different kinds of lot-sizing problems when compared to the default Xpress solver augmented with (ℓ, S) inequalities; Xpress generated a better solution than the heuristic only in 14

of 194 instances, and these were close to the heuristic's solutions. The comparative efficiency of the heuristic is especially noticeable for harder problems, such as set D, SET3 and SET4, where the heuristic improves default Xpress solutions significantly. On the other hand, the proposed heuristic generates results competitive with those generated by SH for the test sets A+, B+, C and D. The advantages of the proposed heuristic over SH are its flexibility in being applicable to problems with different characteristics, its ability to generate lower bounds, its ease of implementation, and the fact that it generates multiple solutions.

3.5 Concluding Remarks

We have presented a heuristic framework that is designed for easy implementation and is flexible enough to handle a variety of production planning problems. The heuristic finds both good solutions and competitive lower bounds. Moreover, computational results indicate that the heuristic is particularly effective on the most difficult problems, where effectiveness is measured by comparison with our benchmarks.

We have also provided further evidence that using (ℓ, S) inequalities is a good method to strengthen the formulation of big bucket problems. Computational experience suggests that they should be used within any default solver. Moreover, when combined with an effective heuristic framework, they can help to find good solutions as well.

Even though mathematical programming provides tools for exact solutions, extended reformulations generally result in big problems that cannot be solved efficiently. A recent study of Van Vyve and Wolsey [2005] suggests that partial reformulations can be applied in order to have high quality lower bounds while preventing the problem size

from growing too much. Similar approaches may be effective for hard lot-sizing problems as well, and this is left for future research.

While the proposed framework for finding good solutions to lot-sizing problems is comparably efficient, many of the instances discussed in the computational results remain challenging. As we discuss in detail in the next chapter, big bucket lot-sizing problems still need deeper analysis and better understanding of the polyhedral structure to improve lower bounds and provide better solutions; polyhedral analysis focused on their particular structure is an important future research area. We will address this issue in Chapter 5.

Another interesting question to be answered is whether this simple approach can be applied to other challenging MIP problems. We know that production planning problems have a special structure in which early decisions affect later decisions, but similar structures may be applicable to other problems as well, e.g. scheduling and facility location problems.

Chapter 4

Finding Lower Bounds

We have studied different solution approaches for production planning problems primarily in the last chapter and proposed a simple and generic framework for obtaining good feasible solutions. Although being able to find exact optimal solutions would be the ultimate goal, we cannot have much expectation in that direction due to problem complexity. On the other hand, even though we might create some useful heuristics to find near-optimal solutions, these methods can fail anytime and more importantly, they often do not provide any insight that could be extended for different approaches.

As we have reviewed in Chapter 2, mathematical programming results on the production planning problems have usually focused on the special cases such as single-item problems and they have been limited for big bucket capacities. However, the actual advantage of these studies is that they help us understand the underlying difficulties solving these problems so that we can develop more efficient techniques. Also it is a common technique in mathematical programming to develop methodologies for the substructures that exist in a problem and then apply these to the main problem. For example, knapsack polytopes are studied widely not due to the fact that they are very interesting problems but because of their existence in many MIP problems.

The primary goals of this chapter are to evaluate different mathematical programming techniques and to investigate why big bucket production planning problems are

hard to solve in practice. More specifically, it is not only aimed to extend the single-item results to general production planning problems, but also we try to discover substructures and relationships between different lower bounds. When doing so, it is important to note that some of these techniques can be computationally inefficient for finding even feasible solutions for these problems.

In the next section, we describe in detail different lower bounding techniques for production planning problems and make some brief remarks on their advantages and disadvantages. The following section is devoted to theoretical comparisons of these techniques. Section 4.3 provides computational results and then we conclude this chapter.

4.1 Overview of Methodologies

This section is divided into three subsections to effectively address related techniques, which include valid inequalities, extended reformulations and Lagrangian relaxations. We will describe these techniques in detail and define problems and feasible regions related to these.

4.1.1 Valid Inequalities

First, recall the (ℓ, S) inequalities (3.1) for the multi-level problems using echelon stocks, which we discussed in detail in Chapter 3:

$$\sum_{t \in S} x_t^i \leq \sum_{t \in S} D_{t,\ell}^i y_t^i + E_\ell^i \qquad \ell \in [1, NT], i \in [1, NI], S \subseteq [1, \ell] \tag{4.1}$$

Recall that although there is an exponential number of these inequalities, a simple polynomial separation algorithm exists, as we have presented in Algorithm 2. (ℓ, S) inequalities are valid for most of the extensions that can be incorporated into the basic formulation of Section 1.4, such as overtime variables. In these cases, violated inequalities can be generated using the same separation algorithm. In case of backlogging, a slightly modified family of (ℓ, S) inequalities (recall (3.2)) can be defined, which use a similar separation algorithm.

We can define the feasible region for the formulation with added (ℓ, S) inequalities as $X_{LS} = \{(x, y, E)|(1.9) - (1.12), (1.18) - (1.20), (4.1)\}$, and the problem can be defined as $Z_{LS} = \min\{(1.21)|\ (x, y, E) \in X_{LS}\}$.

There exist other inequalities than (ℓ, S) inequalities that provide higher lower bounds, but these have good practical use, especially when considering big-size problems. Also note that deleting inactive inequalities of earlier iterations at every new iteration will increase computational efficiency.

Another interesting set of valid inequalities, called as "dicut collection inequalities", are defined originally for fixed-charge network flow problems by Rardin and Wolsey [1993] and can be simply extended to production planning problems. In a network with source and sink nodes S and T, the authors define a "dicut" as a set of arcs whose removal from the network will block the flow from S to T. Strong inequalities can often be constructed using the minimal dicuts. As Rardin and Wolsey [1993] noted, there exist an exponential number of dicut collection inequalities and there is no known polynomial separation algorithm to generate all of these inequalities. More detail on the definition of these inequalities is skipped here to allow the continuity of the thesis and the interested reader is referred to the mentioned paper. The discussion here will be focused on how to

define a polynomial separation algorithm for an exponential subset of these inequalities for production planning problems.

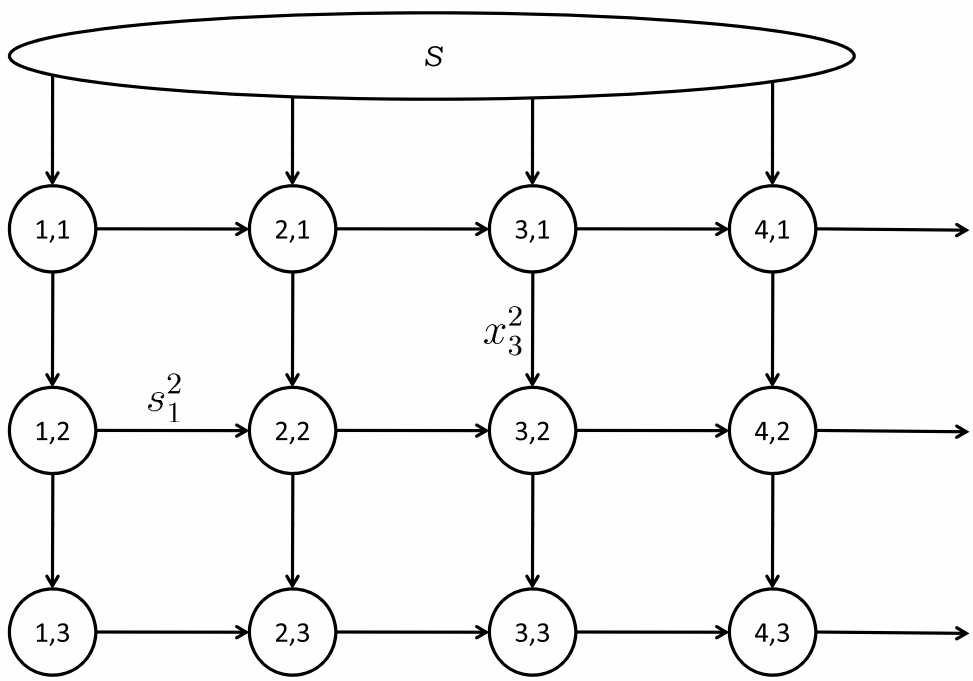

Figure 4.1: Sample graph representation of a serial BOM structure with $NI = 3$

Consider a serial (i.e., each item either has exactly one successor item or is an end-item) production planning problem, and for simplicity, let the parameter r^{ij} be 1 for all immediate successor item pairs i, j. See Figure 4.1 for a network view of the problem with three items, where each node is named after a two-tuple (t, i) representing period and item respectively, horizontal arcs represent inventories carried and vertical arcs representing production variables (the figure gives two examples for these arcs). First, fix the time period t' and create a copy of the network, with an additional node T

as a sink node and additional arcs flowing from the demand nodes to the sink node (see Figure 4.2 for the new network). Using the LP relaxation solution (x^*, y^*, s^*), set capacities on this network to the following values:

- s^*_{t-1} for all horizontal arcs entering node (t, i)
- $\min(x^*_t, d^{NI}_{t,t'} y^*_t)$ for all arcs entering node (t, i)
- d^{NI}_t for all arcs exiting node (t, NI) and entering node T

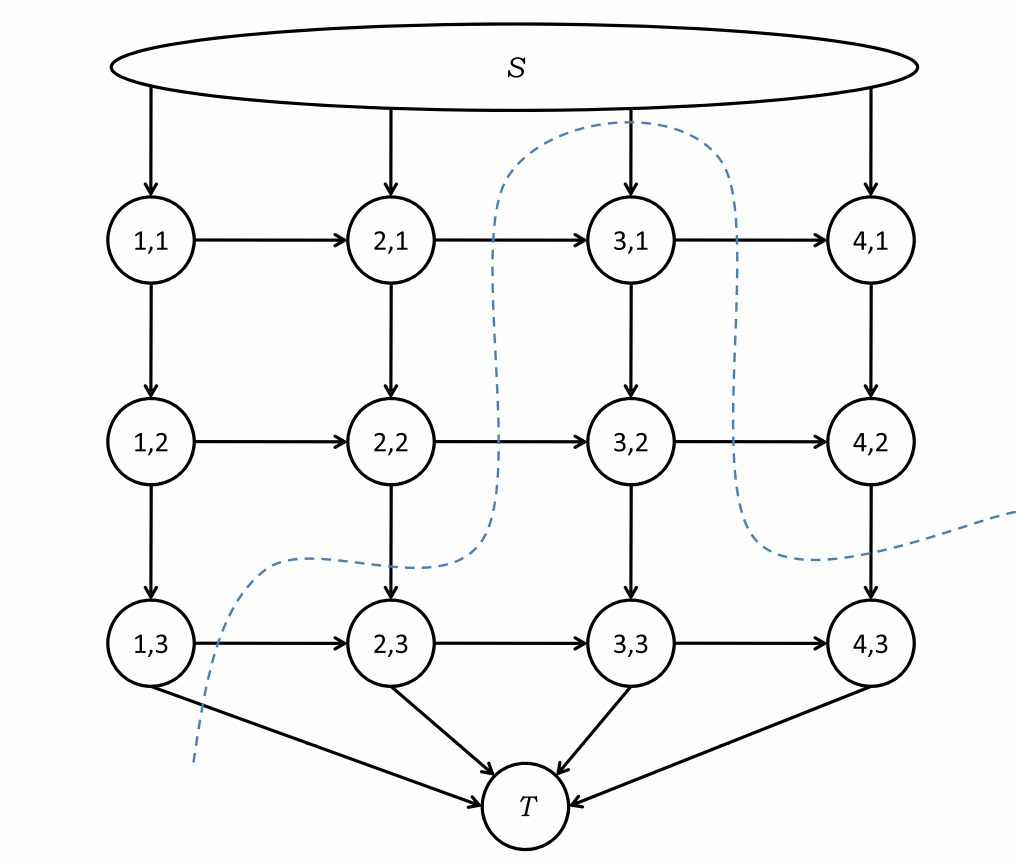

Figure 4.2: Max-flow problem and a sample cut

Once this network is obtained, a maximum flow problem is solved. If the max-flow is strictly less than $d^{NI}_{1,t'}$, then the min-cut defines a valid inequality of the form

$$\sum_{(t,i)\in C_H} s^i_{t-1} + \sum_{(t,i)\in C_V} \min(x^i_t, d^i_{t,t'} y^i_t) + \sum_{t\in C_T} d^{NI}_t \geq d^{NI}_{1,t'} \tag{4.2}$$

Here, C_H and C_V represent horizontal and vertical min-cut arcs entering node (t, i) respectively, C_T represent min-cut arcs exiting from node (t, NI), and all these arcs are directed from S to T. As one can simply observe, (4.2) is violated for (x^*, y^*, s^*). Since for any LP relaxation solution only NT maximum flow problems have to be solved, we have a polynomial separation algorithm.

Example 4.1 *Assume we have a min-cut as drawn in Figure 4.2 for an LPR solution* (x^*, y^*, s^*)*, where* $((x^3_2)^* < d^3_{24}(y^3_2)^*)$*,* $((x^1_3)^* < d^3_{34}(y^1_3)^*)$ *and* $((x^3_4)^* > d^3_4(y^3_4)^*)$*. Then, the valid inequality for this cut can be written as*

$$d^3_1 + s^3_1 + x^3_2 + s^1_2 + s^2_2 + x^1_3 + d^3_4 y^3_4 \geq d^3_{14}$$

This is a valid inequality, one of many for this choice of nodes. For the characteristics of the solution in the question, it happens to be the most violated cut of the subset under consideration.

An important note is that this separation procedure generates not all but only a subset of dicut collection inequalities. However, this set has some important characteristics. One can observe that if $NI = 1$, this procedure is nothing but the separation algorithm of the (ℓ, S) inequalities we previously discussed, and therefore, (ℓ, S) inequalities are already included in the cuts generated through this separation algorithm. Moreover, the path inequalities of Van Roy and Wolsey [1987] are also a subset of the dicut collection inequalities we generate.

We discussed a serial production system to explain our separation algorithm. However, the problems we can run this separation routine are not restricted to serial BOM structures. For example, consider an assembly BOM. In such a system, each beginning item has a serial path to the end item, and therefore, we can create as many serial problems as the number of beginning items and apply our separation algorithm to generate the violated dicut collection inequalities. General BOM structures can be handled in a similar way as well.

One final note on these inequalities is that the projection of the multi-commodity reformulation we will discuss in the next section is equivalent to the region provided by all of the dicut collection inequalities (see Rardin and Wolsey [1993] for the complete proof). In computational results, we will discuss the efficiency of this separation algorithm.

4.1.2 Extended Reformulations

The next technique is the facility location reformulation, originally defined by Krarup and Bilde [1977] for the single-item problem. This reformulation divides production according to which period it is intended for. This requires first defining new variables $u^i_{t,t'}$, which indicate the production of item i in period t to satisfy the demand of period t', where $t' \geq t$. The following constraints should be added into the basic formulation to finalize the reformulation:

$$u^i_{t,t'} \leq D^i_{t'} y^i_t \qquad t \in [1, NT], t' \in [t, NT], i \in [1, NI] \tag{4.3}$$

$$\sum_{t=1}^{t'} u^i_{t,t'} = D^i_{t'} \qquad t' \in [1, NT], i \in [1, NI] \tag{4.4}$$

$$x^i_{t'} = \sum_{t=t'}^{NT} u^i_{t',t} \qquad t' \in [1, NT], i \in [1, NI] \tag{4.5}$$

$$u \geq 0 \tag{4.6}$$

This formulation adds $O(NT^2NI)$ variables and $O(NT^2NI)$ constraints to the problem.

One advantage of using the new variables $u^i_{t,t'}$ is that we can rewrite the capacity constraint (1.9) as follows:

$$\sum_{i=1}^{NI}(a^i_k(\sum_{t'=t}^{NT} u^i_{t,t'}) + ST^i_k y^i_t) \leq C^k_t \qquad t \in [1, NT], k \in [1, NK] \tag{4.7}$$

This, along with constraints (4.3), can considerably help a state-of-the-art MIP solver generate knapsack cover cuts. Specifically, note that by adding $\sum_{i=1}^{NI} a^i_k D^i_{t,NT} y^i_t$ on both sides and after rearranging the terms, (4.7) can be rewritten as

$$\sum_{i=1}^{NI}(a^i_k D^i_{t,NT} + ST^i_k) y^i_t \leq C^k_t + \left(\sum_{i=1}^{NI}\sum_{t'=t}^{NT} a^i_k (D^i_{t'} y^i_t - u^i_{t,t'})\right) \tag{4.8}$$

For each fixed pair of (t, k), and for any subsets $\mathcal{I} \subseteq \{1, ..., NI\}$ and $\mathcal{T} \subseteq \{t, ..., NT\}$, we may generate cover cuts for each of the following continuous 0-1 knapsack constraints:

$$\sum_{i \in \mathcal{I}}(a^i_k(\sum_{t' \in \mathcal{T}} D^i_{t'}) + ST^i_k) y^i_t \leq C^k_t + \left(\sum_{i \in \mathcal{I}}\sum_{t' \in \mathcal{T}} a^i_k (D^i_{t'} y^i_t - u^i_{t,t'})\right) \tag{4.9}$$

Note that because of (4.3), the expression in the parenthesis on the right-hand side of (4.8) or (4.9) can be considered as a single nonnegative continuous variable. Binary knapsack constraints with a single nonnegative continuous variable were studied by Marchand and Wolsey [1999, 2001] (see also Richard et al. [2003a,b]). Commercial solvers use the kinds of results they present to efficiently find subsets $\mathcal{I}$ and $\mathcal{T}$ and generate cover cuts that will approximate $conv(X_{KN}^{(t,k)})$, where $X_{KN}^{(t,k)} = \{(x, y, E, u)|(1.11), (4.3), (4.6), (4.7)\}$ is the feasible region of the intersection of these continuous 0-1 knapsack problems for a fixed (t, k) pair. Of course, this approach will increase the problem size and it might easily become so large that it cannot be solved to optimality in an acceptable time. However, using this approach for the purpose of generating lower bounds can yield insights into the structure of our problems. This idea was initially suggested for single-level, single-machine problems by Van Vyve [2003]. To the best of our knowledge, this approach has not been tested for multi-level problems before.

The feasible region associated with the facility location reformulation can be defined as $X_{FL} = \{(x, y, E, u)|(1.10) - (1.12), (1.18) - (1.20), (4.3) - (4.7)\}$, and the associated problem as $Z_{FL} = \min\{(1.21)|(x, y, E, u) \in X_{FL}\}$. On the other hand, generating all cover cuts approximates $\bigcap_{t=1}^{NT} \bigcap_{k=1}^{NK} conv(X_{KN}^{(t,k)})$, which is an approximation for $conv(\bigcap_{t=1}^{NT} \bigcap_{k=1}^{NK} X_{KN}^{(t,k)})$. This leads us to define the polyhedron $X_{FL}^{KN} = \{(x, y, E, u)|(1.10), (1.12), (1.18) - (1.20), (4.4), (4.5)\} \cap conv(\bigcap_{t=1}^{NT} \bigcap_{k=1}^{NK} X_{KN}^{(t,k)})$ and the associated problem $Z_{FL}^{KN} = \min\{(1.21)|(x, y, E, u) \in X_{FL}^{KN}\}$.

Next, we discuss the single-period relaxation of Miller et al. [2000, 2003], called as *PI* (*Preceding Inventory*). To describe the single-period formulation, for a given machine $k \in [1, NK]$ and a given time period $t \in [1, NT]$, we choose a time period $t(i) \geq t$ for

each $i \in [1, NI]$. Then we define

$$E^i = E^i_{t-1} + \sum_{\hat{t}=t+1}^{t(i)} D^i_{\hat{t}t(i)} y^i_{\hat{t}} \qquad i \in [1, NI]$$

$$D^i = D^i_{tt(i)} \qquad i \in [1, NI]$$

Then, the single-period formulation can be written as follows:

$$x^i_t + E^i \geq D^i \qquad i \in [1, NI] \tag{4.10}$$

$$x^i_t \leq M^i_t y^i_t \qquad i \in [1, NI] \tag{4.11}$$

$$\sum_{i=1}^{NI} (a^i_k x^i_t + ST^i_k y^i_t) \leq C^k_t \tag{4.12}$$

$$x^i_t, E^i \geq 0 \qquad i \in [1, NI] \tag{4.13}$$

$$y^i_t \in \{0, 1\} \qquad i \in [1, NI] \tag{4.14}$$

We can define $X_{PI}^{(t,k,\{t(i)\})} = \{(x, y, E)|(4.10)-(4.14)\}$ as the feasible region associated with a set of $t(i)$ values, and $X_{PI}^{(t,k)} = \bigcap_{\{t(i)\}} X_{PI}^{(t,k,\{t(i)\})}$ represents the feasible region for a given (t, k) pair. Note the similarity between this feasible region and $X_{KN}^{(t,k)}$ we discussed earlier. Miller et al. [2000, 2003] define valid inequalities (namely cover and reverse cover inequalities) for *PI*, which are naturally valid for the original problem as well, and these inequalities can be seen as an approximation for $conv(X_{PI}^{(t,k)})$.

Next, we define the shortest path reformulation of Eppen and Martin [1987]. In this formulation, which was originally defined for single-item uncapacitated models, we define new variables $z^i_{t,t'}$, which are 1 if production of i in period t satisfies all the demand for i in periods $t, ..., t'$, and 0 otherwise. Note the relationship between the new and original variables:

$$x^i_t = \sum_{t'=t}^{NT} D^i_{t,t'} z^i_{t,t'} \qquad t \in [1, NT], i \in [1, NI] \tag{4.15}$$

For the multi-level capacitated problem, we let the z variables take fractional values. Also, using the echelon inventory holding costs H_t^i, we define total inventory costs $c_{t,t'}^i = D_{t,t'}^i \sum_{j=t}^{NT} H_j^i$. Then the formulation is

$$\min \sum_{t=1}^{NT} \sum_{i=1}^{NI} f_t^i y_t^i + \sum_{t=1}^{NT} \sum_{t'=t}^{NT} \sum_{i=1}^{NI} c_{t,t'}^i z_{t,t'}^i \tag{4.16}$$

$$\text{s.t. } 1 = \sum_{t=1}^{NT} z_{1,t}^i \qquad i \in [1, NI] \tag{4.17}$$

$$\sum_{t=1}^{t'-1} z_{t,t'-1}^i = \sum_{t=t'}^{NT} z_{t',t}^i \qquad t' \in [2, NT], i \in [1, NI] \tag{4.18}$$

$$\sum_{t'=t}^{NT} z_{t,t'}^i \leq y_t^i \qquad t \in [1, NT], i \in [1, NI] \tag{4.19}$$

$$\sum_{i=1}^{NI} (ST_k^i y_t^i + a_k^i \sum_{t'=t}^{NT} D_{t,t'}^i z_{t,t'}^i) \leq C_t^k \qquad t \in [1, NT], k \in [1, NK] \tag{4.20}$$

$$\sum_{t=1}^{t'} \sum_{\hat{t}=t}^{NT} (D_{t,\hat{t}}^i z_{t,\hat{t}}^i - \sum_{j \in \delta(i)} r^{ij} D_{t,\hat{t}}^j z_{t,\hat{t}}^j) \geq d_{1,t'}^i \qquad t' \in [1, NT], i \in [1, NI] \tag{4.21}$$

$$z \geq 0 \tag{4.22}$$

$$y \in \{0, 1\}^{NTxNI} \tag{4.23}$$

The constraints (4.17) and (4.18) are the flow balance constraints, (4.19) provide the relationship between the linear and binary variables, (4.20) is the capacity constraint, (4.21) ensures the relationship between different levels, and finally (4.22) and (4.23) provide the nonnegativity and integrality constraints. Note that for our multi-level problem, we derive the constraint (4.21) as follows: Using (1.18) and (1.19), and the assumption of zero initial inventory, we obtain

$$\sum_{t=1}^{t'}(x_t^i - D_t^i) \geq \sum_{t=1}^{t'} \sum_{j \in \delta(i)} r^{ij}(x_t^j - D_t^j) \tag{4.24}$$

Substituting (4.15) into (4.24) and rewriting results in (4.21). Note that this formulation adds as many variables as the facility location reformulation, but number of constraints is only $O(NT\text{x}NI)$. However, this formulation is not necessarily easier to solve, in part because the new constraints are comparatively dense and the coefficients on the new variables comparatively large.

The feasible region associated with this formulation can be defined as $X_{SP} = \{(y, z)|$ $(4.17) - (4.23)\}$, and the problem can be defined as $Z_{SP} = \min\{(4.16)|(y, z) \in X_{SP}\}$. Part of our motivation for completely substituting the x and E variables out of the formulation is that relaxing the constraints (4.17), (4.18), and (4.21) decomposes the problem into NT distinct subproblems, one for each time period (an analogous observation was first made for single-level problems by Jans and Degraeve [2004]). We will discuss this property in more detail later.

Next, we consider the multi-commodity reformulation proposed by Rardin and Wolsey [1993]. This approach is originally described for fixed-charge network flow problems. Like the facility location reformulation, it divides production using destination information, but since we have multiple levels, it also includes information about which end-item in the BOM it is produced for. Stock variables are also divided in a similar fashion. Thus, the new variables $w_{t,t'}^{i,j}$ indicate production of item i in period t to satisfy the demand of end item j in period t', $t' \geq t$, and the new variables $v_{t,t'}^{i,j}$ indicate the inventory of item i held over at the end of period t to satisfy demand of end item j in period t', $t' > t$. The following constraints should be added to the basic formulation to finalize

the reformulation:

$$x^i_{t'} = \sum_{t=t'}^{NT} \sum_{j \in endp} w^{i,j}_{t',t} \qquad t' \in [1, NT], i \in [1, NI] \tag{4.25}$$

$$w^{i,j}_{t,t'} \leq r^{ij} d^j_{t'} y^i_t \qquad t \in [1, NT], t' \in [t, NT], \quad i \in [1, NI], j \in endp \tag{4.26}$$

$$v^{i,i}_{t-1,t} + w^{i,i}_{t,t} = d^i_t \qquad t \in [1, NT], i \in endp \tag{4.27}$$

$$v^{i,i}_{t-1,t'} + w^{i,i}_{t,t'} = v^{i,i}_{t,t'} \qquad t \in [1, NT-1], t' \in [t+1, NT], \quad i \in endp \tag{4.28}$$

$$v^{i,q}_{t-1,t} + w^{i,q}_{t,t} = \sum_{j \in \delta(i)} r^{ij} w^{j,q}_{t,t} \qquad t \in [1, NT], i \in [1, NI] \backslash endp, \quad q \in endp \tag{4.29}$$

$$v^{i,q}_{t-1,t'} + w^{i,q}_{t,t'} = v^{i,q}_{t,t'} + \sum_{j \in \delta(i)} r^{ij} w^{j,q}_{t,t'} \qquad t \in [1, NT-1], t' \in [t+1, NT], \quad i \in [1, NI] \backslash endp, q \in endp \tag{4.30}$$

$$w, v \geq 0 \tag{4.31}$$

This reformulation introduces $O(NT^2 NI^2)$ additional variables and $O(NT^2\ NI^2)$ additional constraints. This is the main disadvantage of this reformulation, which can become computationally intractable as the problem size grows. However, it is the tightest compact, i.e., polynomial size, reformulation that we know for the problems in question.

The feasible region associated with this formulation can be defined as $X_{MC} = \{(x, y, E, w, v) | (1.9) - (1.12), (1.18) - (1.20), (4.25) - (4.31)\}$, and the problem can be defined as $Z_{MC} = \min\{(1.21) |\ (x, y, E, w, v) \in X_{MC}\}$.

4.1.3 Lagrangian Relaxation

Next, we discuss three approaches that employ Lagrangian relaxation to obtain structured subproblems and from those lower bounds for the original problem. The first approach is to relax the capacity constraints (1.9), and obtain

$$\begin{aligned} LR_1(\lambda) =& \min \sum_{t=1}^{NT}\sum_{i=1}^{NI} f_t^i y_t^i + \sum_{t=1}^{NT}\sum_{i=1}^{NI} H_t^i E_t^i \\ & - \sum_{t=1}^{NT}\sum_{k=1}^{NK} \lambda_t^k \left(C_t^k - (\sum_{i=1}^{NI} a_k^i x_t^i + ST_k^i y_t^i) \right) \\ & \text{subject to } (x, y, E) \in X_{LR1} \end{aligned} \tag{4.32}$$

where $X_{LR1} = \{(x, y, E)|(1.10)-(1.12), (1.18)-(1.20)\}$. Thus, the Lagrangian subproblem is a multi-item, multi-level uncapacitated lot-sizing problem. The Lagrangian dual problem is

$$LD_1 = \max_{\lambda \geq 0} LR_1(\lambda)$$

The next Lagrangian relaxation approach relaxes the constraints linking separate levels, i.e. constraints (1.19), to obtain

$$\begin{aligned} LR_2(\mu) =& \min \sum_{t=1}^{NT}\sum_{i=1}^{NI} f^i y_t^i + \sum_{t=1}^{NT}\sum_{i=1}^{NI} H_t^i E_t^i \\ & - \sum_{t=1}^{NT}\sum_{i=1}^{NI} \mu_t^i \left(E_t^i - \sum_{j \in \delta(i)} r^{ij} E_t^j \right) \\ & \text{subject to } (x, y, E) \in X_{LR2} \end{aligned} \tag{4.33}$$

where $X_{LR2} = \{(x, y, E)|\ (1.9) - (1.12), (1.18), (1.20)\}$. The Lagrangian subproblem therefore decomposes into NK disjoint multi-item, big bucket single-machine problems, one for each machine. The Lagrangian dual problem becomes

$$LD_2 = \max_{\mu \geq 0} LR_2(\mu)$$

Finally, the last Lagrangian approach extends the work of Jans and Degraeve [2004] for single-level problems, which itself uses the shortest path reformulation of Eppen and Martin [1987]. Their approach simply relaxed the constraints linking time periods, yielding disjoint single-period subproblems. However, the problem in the multi-level case is that the constraints linking levels also involve multiple periods. Therefore, decomposing the problem into disjoint subproblems for each period is not possible, unless all constraints linking levels are also dualized. We dualize the constraints (4.17), (4.18) and (4.21) in the shortest path reformulation to obtain

$$\begin{aligned} LR_3(\beta, \gamma) = & \min \sum_{t=1}^{NT} \sum_{i=1}^{NI} f_t^i y_t^i + \sum_{t=1}^{NT} \sum_{t'=t}^{NT} \sum_{i=1}^{NI} c_{t,t'}^i z_{t,t'}^i - \sum_{i=1}^{NT} \beta_1^i \left(1 - \sum_{t=1}^{NT} z_{1,t}^i \right) \\ & - \sum_{i=1}^{NI} \sum_{t'=2}^{NT} \beta_{t'}^i \left(\sum_{t=1}^{t'-1} z_{t,t'-1}^i - \sum_{t=t'}^{NT} z_{t',t}^i \right) \\ & - \sum_{i=1}^{NI} \sum_{t'=1}^{NT} \gamma_{t'}^i \left(\sum_{t=1}^{t'} \sum_{\hat{t}=t}^{NT} (D_{t,\hat{t}}^i z_{t,\hat{t}}^i - \sum_{j \in \delta(i)} r^{ij} D_{t,\hat{t}}^j z_{t,\hat{t}}^j) - d_{1,t'}^i \right) \\ & \text{s.t. } (y, z) \in X_{LR3} \end{aligned} \tag{4.34}$$

where $X_{LR3} = \{(y, z)|\ (4.19), (4.20), (4.22), (4.23)\}$. The Lagrangian subproblem decomposes into NKxNT disjoint capacitated multi-item, single-machine, single-period

problems, and the Lagrangian dual problem is

$$LD_3 = \max_{\gamma \geq 0, \beta} LR_3(\beta, \gamma)$$

4.2 Comparative Relationships between Methodologies

Let the superscript LP indicate the LP relaxation of a problem, i.e., the binary variables y relaxed to be continuous with the bounds $0 \leq y \leq 1$. For example, Z_{LS}^{LP} is the problem Z_{LS} with the integrality requirements for y variables relaxed. Similarly, X_{LS}^{LP} is the polyhedron of the LP relaxation of X_{LS}.

Proposition 4.2 $Z_{LS}^{LP} = Z_{FL}^{LP}$.

In words, adding (ℓ, S) inequalities to the original formulation and using the facility location reformulation provide the same lower bound for the multi-level lot-sizing problem.

Proof. It is known that both of these formulations give the convex hull for the uncapacitated single-item problem (Barany et al. [1984b], Krarup and Bilde [1977], Pochet and Wolsey [2006]). This fact will be used to conclude the proof.

Let $X'_{IP} = \{(x, y, E) | (1.10), (1.11), (1.12), (1.18), E \geq 0\}$. Now define the following Lagrangian function $\widehat{LR}_1(\lambda, \mu)$ by relaxing capacity constraint (1.9) and multi-level constraint (1.19):

$$\begin{aligned} \min \sum_{t=1}^{NT} \sum_{i=1}^{NI} f_t^i y_t^i + \sum_{t=1}^{NT} \sum_{i=1}^{NI} H_t^i E_t^i - \sum_{t=1}^{NT} \sum_{k=1}^{NK} \lambda_t^k (C_t^k - \sum_{i=1}^{NI} (a_k^i x_t^i + ST_k^i y_t^i)) \\ - \sum_{t=1}^{NT} \sum_{i=1}^{NI} \mu_t^i (E_t^i - \sum_{j \in \delta(i)} r^{ij} E_t^j) \end{aligned} \tag{4.35}$$

st. $(x, y, E) \in X'_{IP}$

Let $X'_{LS} = \{(x, y, E)|(1.10), (1.12), (1.18), (4.1), E \geq 0, 0 \leq y \leq 1\}$ and $X'_{FL} = \{(x, y, E, u)|(1.10), (1.12), (1.18), (4.3) - (4.5), E \geq 0, 0 \leq y \leq 1\}$, i.e., X^{LP}_{LS} and X^{LP}_{FL} without capacity and multilevel constraints, respectively. Now, define the following Lagrangian functions:

$$\widehat{LR}_2(\lambda, \mu) = \min\{(4.35)|(x, y, E) \in X'_{LS}\}$$

$$\widehat{LR}_3(\lambda, \mu) = \min\{(4.35)|(x, y, E, \bar{x}) \in X'_{FL}\}$$

Observe that X'_{IP} is the intersection of NI convex sets, each of which is equivalent to the set of the single-item uncapacitated lot-sizing problem for an item i. That is, by relaxing the capacity constraints and constraints linking levels, we obtain uncapacitated single-item problems.

Now, define the Lagrangian dual $\widehat{LD}_i$ for each Lagrangian relaxation problem i, $i = 1, 2, 3$, as follows:

$$\widehat{LD}_i = \max_{\mu, \lambda \geq 0} LR_i(\lambda, \mu)$$

Finally, by the Proposition 1.14:

$$\widehat{LD}_1 = \min\{(1.21)|(x, y, E) \in X'\}$$

$$\widehat{LD}_2 = \min\{(1.21)|(x, y, E) \in X''\}$$

$$\widehat{LD}_3 = \min\{(1.21)|(x, y, E, u) \in X'''\}$$

where $X' = \{(x, y, E) \in conv(X'_{IP})|(1.9), (1.19)\}$, $X'' = \{(x, y, E) \in X'_{LS}|(1.9), (1.19)\}$ and $X''' = \{(x, y, E) \in X'_{FL}|(1.9), (1.19)\}$. Since $conv(X'_{IP}) = X'_{LS}$ and $conv(X'_{IP}) = proj_{x,y,E}(X'_{FL})$, $\widehat{LD}_2 = \widehat{LD}_3$ holds. Hence, $Z^{LP}_{LS} = Z^{LP}_{FL}$. $\square$

Corollary 4.3 $Z_{LS}^{LP} = Z_{FL}^{LP} = Z_{SP}^{LP}$, *i.e.,* (ℓ, S) *inequalities, facility location reformulation and shortest path reformulation all provide the same lower bound for the original problem.*

Using the fact that the projections of both the shortest path and the facility location reformulations onto original variable space define the convex hull of the single-item uncapacitated problem, this corollary follows the proposition directly, as it can be proven with the same technique used for proving the Proposition 4.2. Even though all three methods discussed above provide the same lower bounds, the facility location and shortest path reformulations have the disadvantage of making the problem size much larger than the (ℓ, S) inequalities, which are dynamically added to the formulation and deleted if inactive. Our experience shows that, for this reason, (ℓ, S) inequalities allow for more efficient B&B and identifying of feasible solutions. See Krarup and Bilde [1977], Barany et al. [1984b] and Eppen and Martin [1987] for the convex hull and integrality proofs in the single-item case.

Theorem 4.4 (Rardin and Wolsey [1993]) $Z_{MC}^{LP} \geq Z_{FL}^{LP}$, *i.e., the multi-commodity reformulation provides at least as strong a lower bound as the facility location reformulation. If the problem consists of a single level, then* $Z_{MC}^{LP} = Z_{FL}^{LP}$.

Although this result has been known by at least some researchers since the publication of Rardin and Wolsey [1993], we include a proof for the sake of completeness.

Proof. We will prove this by showing that $proj_{x,y,E}(X_{MC}^{LP}) \subseteq proj_{x,y,E}(X_{FL}^{LP})$ for the multi-level case. Let $(v^*, w^*, x^*, y^*, E^*) \in X_{MC}^{LP}$. First, observe that we can eliminate v^*

and rewrite (4.27)-(4.30) in terms of w^*, as follows:

$$\sum_{t=1}^{t=t'} w^{*i,j}_{t,t'} = r^{ij} d^j_{t'} \qquad t' \in [1, NT], i \in [1, NI], j \in endp \tag{4.36}$$

Now, let

$$u^{*i}_{tt'} = \sum_{j \in endp} w^{*ij}_{tt'} \tag{4.37}$$

Obviously $u^* \geq 0$ since $w^* \geq 0$. Since w^* satisfies (4.25), $x^{*i}_t = \sum_{t'=t}^{NT} u^{*i}_{tt'}$. Similarly, summing (4.36) over $j \in endp$, we obtain $\sum_{t=1}^{t'} u^{*i}_{tt'} = \sum_{j \in endp} r^{ij} d^j_{t'} = D^i_{t'}$, where the second equation follows from the definition of echelon demand (1.16). Finally, using (4.26) and (4.37), we obtain $u^{*i}_{tt'} = \sum_{j \in endp} w^{*ij}_{tt'} \leq (\sum_{j \in endp} r^{ij} d^j_{t'}) y^{*i}_t = D^i_{t'} y^{*i}_t$. This shows that $(u^*, x^*, y^*, E^*) \in X^{LP}_{FL}$. Hence, $proj_{x,y,E}(X^{LP}_{MC}) \subseteq proj_{x,y,E}(X^{LP}_{FL})$. $\square$

The second part of the theorem can also be shown using the same technique as in the first proof, i.e., using Lagrangian duality and the fact that the multi-commodity reformulation and the facility location reformulation provide equivalent lower bounds in the single-item case (see Barany et al. [1984b] and Eppen and Martin [1987]).

This theorem shows us theoretically that the multi-commodity reformulation is stronger than the formulation defined by adding (ℓ, S) inequalities, the facility location reformulation, and the shortest path reformulation. In the next section, we will computationally address the question of "how much stronger" for a variety of test problems.

So far we have made comparisons of different polyhedral approaches. Also interesting are the relationships between the Lagrangian approaches and these reformulations, as we investigate in the following results.

Theorem 4.5 $Z^{LP}_{MC} \leq LD_1$.

In words, the lower bound obtained by the Lagrangian that relaxes the capacity constraints is at least as strong as the lower bound obtained by multi-commodity reformulation.

Proof. By Proposition 1.14,

$$LD_1 = \min\{(1.21)|(x,y,E) \in (1.9) \cap conv((1.10)-(1.12),(1.18)-(1.20))\}$$

On the other hand,

$$Z_{MC}^{LP} = \min\{(1.21)|(x,y,E,w,v) \in (1.9) \cap \{((1.10),(1.12),(1.18)-(1.20),(4.25)-(4.31)) \cap \ conv((1.11))\}\}$$

Observe that

$$\{(x,y,E) \in conv((1.10)-(1.12),(1.18)-(1.20))\} \subseteq proj_{(x,y,E)}\{(x,y,E,w,v) \in \{((1.10),(1.12),(1.18)-(1.20),(4.25)-(4.31)) \cap \ conv((1.11))\}\}$$

This follows because $conv((1.10)-(1.12),(1.18)-(1.20))$ has integer extreme points because the polyhedron is the convex hull of an integer feasible region. On the other hand, $\{((1.10),(1.12),(1.18)-(1.20),(4.25)-(4.31)) \cap \ conv((1.11))\}$ does not necessarily have integral extreme points. Therefore, $Z_{MC}^{LP} \leq LD_1$. $\square$

Theorem 4.6 $Z_{FL}^{LP} \leq Z_{FL}^{KN} \leq LD_2$.

In words, the lower bound obtained by the Lagrangian that relaxes the level linking constraints is at least as strong as the lower bound obtained by the facility location reformulation strengthened to approximate the knapsack convex hulls.

Proof. The first relationship follows from the fact that Z_{FL}^{KN} is obtained by strengthening Z_{FL}^{LP} with additional constraints. For the second relationship, first observe that (using Proposition 1.14)

$$LD_2 \quad = \quad \min\{(1.21)|(x,y,E) \in (1.19) \cap conv((1.9)-(1.12),(1.18),(1.20))\}$$

Observe also that

$$conv((1.9)-(1.12),(1.18),(1.20)) \subseteq proj_{x,y,E}$$
$$\left\{\{(x,y,E,u)|(1.10),(1.12),(1.18),(1.20),(4.4),(4.5)\} \cap conv(\bigcap_{t=1}^{NT}\bigcap_{k=1}^{NK} X_{KN}^{(t,k)})\right\}$$

This shows that Z_{FL}^{KN} is not as strong as LD_2. $\square$

As mentioned before, generating cover cuts from (4.9) only approximates the knapsack polyhedron and hence Z_{FL}^{KN} is the best possible bound that can be obtained by adding cover cuts to the LP relaxation of the facility location reformulation.

Theorem 4.7 $Z_{FL}^{KN} = LD_3$.

We need the following result for the proof of the theorem.

Lemma 4.8 (Pochet and Wolsey [1988]) *All optimal solutions of the single-item uncapacitated problem formulated using the facility location reformulation have the following property:*

$$\frac{u_{tt'}}{D_{t'}} \geq \frac{u_{tt'+1}}{D_{t'+1}} \qquad t \in [1,NT], t' \geq t$$

Before starting the proof of the theorem, let $S_1 = \bigcap_{t=1}^{NT}\bigcap_{k=1}^{NK} X_{KN}^{(t,k)} = \{(y,u)|(1.11),(4.3),(4.6),(4.7)\}$ and $S_2 = \{(y,z)|(4.19),(4.20),(4.22),(4.23)\}$. Also let $T_1 = \{(x,y,E,u)|$

$((1.18) - (1.20), (4.5)) \cap conv(S_1)\}$ and $T_2 = \{(x, y, E, z)|((1.18) - (1.20), (4.15)) \cap conv(S_2)\}$. Note that S_1 and S_2 are integer feasible regions whereas T_1 and T_2 are both polyhedra.

Proof. We will prove this by showing $proj_{x,y,E}(T_1) = proj_{x,y,E}(T_2)$.

First, let $(x^*, y^*, E^*, u^*) \in T_1$ and hence $(x^*, y^*, E^*) \in proj_{x,y,E}(T_1)$. Therefore, $\exists p^j = (x^j, y^j, E^j, u^j) \in S_1$, $j \in [1, J]$, such that $(x^*, y^*, E^*, u^*) = \sum_{j=1}^{J} \lambda_j p^j$ for some $\lambda \geq 0$, $\sum_{j=1}^{J} \lambda_j = 1$.

For all $j \in [1, J]$, let $\{z^i_{tNT}\}^j = \frac{\{u^i_{tNT}\}^j}{D^i_{NT}}$, where $t \in [1, NT]$ and $i \in [1, NI]$. Then, define recursively $\{z^i_{tt'}\}^j = \frac{\{u^i_{tt'}\}^j}{D^i_{NT}} - \sum_{\bar{t}=t'+1}^{NT} \{z^i_{t\bar{t}}\}^j$, for all $t \in [1, NT]$, $t' = NT - 1, ..., t$ and $i \in [1, NI]$. Since $\sum_{t'=t}^{NT} D^i_{tt'} \{z^i_{tt'}\}^j = \sum_{t'=t}^{NT} \{u^i_{tt'}\}^j$ and u^j satisfies (4.7), z^j satisfies (4.20). Next, note that

$$\sum_{t'=t}^{NT} \{z^i_{tt'}\}^j = \frac{\{u^i_{tt}\}^j}{D^i_t} \leq \{y^i_t\}^j$$

where the last inequality is essentially (4.3). Finally, using Lemma 4.8, observe that

$$\{z^i_{tt'}\}^j = \frac{\{u^i_{tt'}\}^j}{D^i_{t'}} - \frac{\{u^{*i}_{tt'+1}\}^j}{D^i_{t'+1}} \geq 0$$

Therefore, $\hat{p}^j = (x^j, y^j, E^j, z^j) \in S_2$, and using the same λ as before, $(x^*, y^*, E^*, z^*) = \sum_{j=1}^{J} \lambda_j \hat{p}^j \in T_2$. Hence, $(x^*, y^*, E^*) \in proj_{x,y,E}(T_2)$. We conclude therefore that $proj_{x,y,E}(T_1) \subseteq proj_{x,y,E}(T_2)$.

Now, let $(x^*, y^*, E^*, z^*) \in T_2$ and hence $(x^*, y^*, E^*) \in proj_{x,y,E}(T_2)$. Therefore, $\exists q^k = (x^k, y^k, E^k, z^k) \in S_2$, $k \in [1, K]$, such that $(x^*, y^*, E^*, z^*) = \sum_{k=1}^{K} \mu_k q^k$ for some $\mu \geq 0$, $\sum_{k=1}^{K} \mu_k = 1$.

For all $k \in [1, K]$, let $\{u^i_{tt'}\}^k = D^i_{t'} \sum_{\bar{t}=t'}^{NT} \{z^i_{t\bar{t}}\}^k$, where $t \in [1, NT]$, $t' \in [t, NT]$, and

$i \in [1, NI]$. Obviously, u^k satisfies (4.6) since z^k satisfies (4.22). Since $\sum_{t'=t}^{NT} \{u^i_{tt'}\}^k = \sum_{t'=t}^{NT} D^i_{tt'} \{z^i_{tt'}\}^k$ and z^k satisfies (4.20), u^k satisfies (4.7). Finally, note that

$$\{u^i_{tt'}\}^k = D^i_{t'} \sum_{\bar{t}=t'}^{NT} \{z^i_{t\bar{t}}\}^k \leq D^i_{t'} \sum_{\bar{t}=t}^{NT} \{z^i_{t\bar{t}}\}^k \leq D^i_{t'} \{y^i_t\}^k$$

where the last inequality follows from (4.19).

Therefore, $\hat{q}^k = (x^k, y^k, E^k, u^k) \in S_1$, and using the same μ as before, $(x^*, y^*, E^*, u^*) = \sum_{k=1}^{K} \mu_k \hat{q}^k \in T_1$. Hence, $(x^*, y^*, E^*) \in proj_{x,y,E}(T_1)$. Therefore, $proj_{x,y,E}(T_2) \subseteq proj_{x,y,E}(T_1)$. This concludes the proof. □

Corollary 4.9 $LD_3 \leq LD_2$.

The proof for this corollary follows immediately from the Theorems 4.6 and 4.7. This result is our main motivation for skipping LD_3 in the computational tests discussed in the next section.

Proposition 4.10 $conv(X^{(t,k)}_{PI}) = proj_{x,y,E}(conv(X^{(t,k)}_{KN}))$.

This result, combined with Corollary 4.9, is our main motivation for omitting computationally testing the cover and reverse cover inequalities of Miller et al. [2000, 2003] in the next section.

4.3 Computational Results

The main goal of this section is to computationally test the results we have theoretically proven and to observe how these strength relationships work in practice. This not only provides us with information about how strong the lower bounds actually are but also helps us to understand what prevents us from improving them. All the test instances

are run on a PC with an Intel Pentium 4 2.53 GB processor and 1 GB of RAM. All the formulations are implemented using Mosel (Xpress-MP 2004C package and Mosel version 1.4.1). The test instances introduced and used in the previous chapter are used for the purposes of this section.

Table 4.1 summarizes some results on the dicut collection inequalities generated through the separation algorithm we discussed in Section 4.1.1. The table presents lower bounds generated using the (ℓ, S) inequalities, the subset of dicut collection inequalities generated through the separation procedure, and multi-commodity reformulation.

Table 4.1: Dicut collection inequalities separation results

Instance	(ℓ, S)	DCI	MC	Instance	(ℓ, S)	DCI	MC
AK501131	96,968	96,975	96,983	BK511141	125,307	125,457	125,494
AK501132	101,699	101,719	101,781	BK512131	90,733	90,735	90,787
AK501141	134,805	134,940	134,943	BK512132	90,814	90,842	90,858
AK501142	134,880	134,984	135,006	BK521131	92,350	92,361	92,382
AK501432	92,533	92,572	92,605	BK521132	94,257	94,287	94,317
AK502130	102,222	102,242	102,245	BK521142	124,988	125,232	125,257
AK502131	93,369	93,369	93,423	BK522131	90,532	90,536	90,588
AK502132	96,312	96,316	96,396	BK522142	119,559	119,721	119,739
AK502142	127,792	127,964	127,977	CK501120	141,900	142,021	142,034
AK502432	88,980	89,037	89,088	CK501221	101,028	101,068	101,108
BK511131	92,602	92,620	92,640	CK501121	131,993	132,173	132,185
BK511132	95,323	95,334	95,355	CK501122	153,861	153,873	154,358

As these results indicate, dicut collection inequalities provide no significant improvement over the (ℓ, S) inequalities. As we will see later in more detail, the multi-commodity reformulation itself does not provide a much stronger lower bound than (ℓ, S) inequalities, which is the motivation of excluding discussion of dicut collection inequalities from the rest of the thesis. Next, we will make some remarks on the computations and comparisons we accomplish in the rest of the chapter.

In evaluating Lagrangians, we do not calculate any of the Lagrangian duals exactly using a subgradient optimization procedure. Instead, we use the approach of starting with a strengthened LP formulation, i.e., (ℓ, S) inequalities, and obtain the optimal values for the duals of the constraints relaxed to use as Lagrange multipliers. We approximate LD_1 and LD_2 by solving the Lagrangian subproblems evaluated at these multipliers. One important note is that the lower bounds obtained from these approximations are at least as strong as the lower bound provided by our base formulation, i.e., that strengthened with (ℓ, S) inequalities. Similarly, as we previously discussed, generating cover cuts on top of the facility location reformulation provides only an approximation to Z_{FL}^{KN}. Hence, the computational comparisons we provide for these relationships are all based on approximations. However, it seems that the approximations are often close. This gives us the chance to compare empirical results in addition to theoretically proven relationships.

The detailed results for TDS instances are provided in Appendix C.1. The first three columns indicate the lower bound at the root node solution of the Branch&Bound tree for (ℓ, S) inequalities, multi-commodity reformulation (MC) and facility location reformulation (FL), respectively. Only the bound for FL contains the cuts generated by the solver (only cover cuts added) and the other two bounds do not involve any additional default cuts. The next two columns indicate the lower bounds obtained by running the Lagrangian relaxations that relax the capacity and level-linking constraints, respectively, after default times of 180 seconds for A+ and B+ instances, and 500 seconds for C and D instances. The next two columns of the results show the upper bounds for the Lagrangian relaxations after the default times. The main reason for including the upper bounds is that if the Lagrangian relaxation subproblem is not solved to optimality

for the specified set of multipliers, then we only know that the lower bound provided by these Lagrangian duals lies between the lower and upper bounds presented in this table. Also recall the computational results of previous chapter, presented in Appendix B, which we will refer for some comparisons as well. Note that we do not include the lower bounds obtained by the shortest path reformulation, as they are equivalent to the bounds provided by (ℓ, S) inequalities and the facility location reformulation (without any additional cuts).

In general, adding (ℓ, S) inequalities and running default Xpress seems to be the most computationally tractable method for strengthening the formulation, if our goal is to solve problems quickly rather than acquire information. Moreover, as discussed in the previous chapter, this also often generates acceptable feasible solutions. This gives us reason to conclude that (ℓ, S) inequalities are computationally efficient for general production planning problems. On the other hand, our heuristic provide competitive lower bounds on hard problems. However, these conclusions do not provide much insight on problem structure, particularly considering that the final duality gaps remain quite high.

We review these results now in pairwise comparisons, which are summarized in Table 4.2. One interesting computational comparison is the relationship we have proven in Theorem 4.4. As we can see from the detailed results, MC improves the (ℓ, S) bound slightly, in general less than %1. The average improvements from the (ℓ, S) inequalities bound to the MC bound, calculated as (MC bound - ℓ, S bound)/(ℓ, S bound) for each test instance, are provided in the column "MC vs. ℓ, S", and these values are around 0.20%. Considering the enormous size of the MC reformulation, these improvements are simply not worth the computational effort. The Lagrangian relaxation that relaxes the

capacity constraints (1st LR) provides in general another slight improvement over the lower bounds of the MC reformulation, as can be seen in the second column of the same table (Column LB under "1st LR vs. MC"), which is calculated in a similar fashion, i.e., (1st LR bound - MC bound)/(MC bound). Note that we also provide averages calculated in the same way using the 1st LR's upper bounds instead of its lower bounds (Column UB under "1st LR vs. MC"). An interesting observation from the problems in set D, where 1st LR problems for all instances are solved to optimality, is that although in general 1st LR improves the MC bound, it is an approximation of LD_1 and it might result in a bound not as strong as the MC bound. However, as these results indicate, these two bounds are in general very close to each other.

Table 4.2: Pairwise comparisons of lower bounds and LR gaps for TDS instances

Test Set	MC vs. ℓ, S	1st LR vs. MC		FL vs. ℓ, S	2nd LR vs. FL		Gaps	
		LB	UB		LB	UB	1st LR	2nd LR
A+	0.29%	0.80%	2.99%	1.81%	-0.05%	7.44%	2.09%	6.87%
B+	0.28%	0.59%	3.06%	1.37%	-0.35%	6.23%	2.38%	6.18%
C	0.14%	0.20%	1.67%	0.86%	-0.32%	6.25%	1.44%	6.14%
D	0.21%	-0.06%	-0.06%	0.45%	-0.43%	19.88%	0%	15.85%

On the other hand, as the "FL vs. ℓ, S" column of Table 4.2 indicates, the facility location reformulation with cover cuts added (FL) improves in general the (ℓ, S) bound more significantly compared to previous methods. These average percentages are calculated by (FL bound - ℓ, S bound)/(ℓ, S bound). Similar to our previous comparisons, we also provide the average improvements of the Lagrangian relaxation that relaxes level-linking constraints (2nd LR) over the FL bound in the column "2nd LR vs. FL", calculated by (2nd LR bound - FL bound)/(FL bound). Although one would expect the 2nd LR, the approximation of LD_2, to improve the FL lower bounds, at first sight this does not seem to be the case for many problem instances, particularly due to negative

averages in the LB column of Table 4.2. However, as can be seen from the UB column of the table, these problems are not close to optimality, particularly the bigger instances of test sets C and D, and the challenge here is that these problems need much more time than the assigned default times (or any reasonable amount of time) for optimality or even for an acceptable gap. For testing whether this is the case here, we experimented with a few of the small A+ and B+ instances that did not achieve the FL bounds earlier and ran them either until the lower bound was at least as strong as the FL bound or to optimality. For some instances, however, the experiment failed due to memory problems, particularly for sets C and D.

Finally, the last two columns of Table 4.2 should also be addressed briefly. These columns indicate the duality gaps for the two Lagrangian problems, and as we mentioned before, the 1st LR problem is in general comparatively easier to solve than the 2nd LR problem. We had a total of 11 instances where the 1st LR could achieve the optimal solution in the assigned default times, compared to none for the 2nd LR.

Next, we present results for LOTSIZELIB instances in Table 4.3, which is organized in a similar fashion to the detailed results of TDS instances in Appendix C.1. MC provides significant improvement over the (ℓ, S) bound for some of these instances, whereas FL provides negligible improvement over MC. The 1st LR is comparatively more efficient on these instances than the 2nd LR. Note that 1st LR and 2nd LR do not necessarily improve MC and FL bounds respectively, similarly to the results for some TDS instances, since these are approximations for LD_1 and LD_2. Also, note that all 2nd LR problems are at optimality or near, whereas 1st LR did not result in optimality in quite a few instances after the default time of 180 seconds. This indicates that these instances have the bottleneck not in capacity constraints but in the multi-level structure. This seems

to be due in part to the fact that there is a single machine, and the capacity in these problems is comparatively loose.

Table 4.3: LOTSIZELIB results

	Lower Bounds					**Upper Bounds**	
	ℓ, S	MC	FL	1st LR (Cap)	2nd LR (Lev)	1st LR (Cap)	2nd LR (Lev)
LLIB B	3,888	3,890	3,892	3,888	3,888	3,888	3,888
LLIB C	1,904	1,993	1,998	1,904	1,904	1,904	1,905
LLIB D	4,534	4,794	4,795	4,766	4,534	6,095	4,535
LLIB E	2,341	2,361	2,361	2,462	2,341	3,136	2,341
LLIB F	2,075	2,098	2,111	2,237	2,079	2,459	2,079

The detailed results on Multi-LSB instances are presented in Appendix C.2, which is organized in the same fashion as Appendix C.1, and the pairwise comparisons are summarized in Table 4.4, which is organized in the same fashion as Table 4.2. The default times for the first two sets are 180 seconds, and for the last two sets 500 seconds. First of all, note that MC improves the (ℓ, S) bound poorly in most of the instances. Also note that the 1st LR is solved to optimality for all these test problems, and as the table indicates, this approximation of LD_1 does not often provide an improvement over MC. This is due in part to poor multipliers generated from the (ℓ, S) formulation.

Table 4.4: Pairwise comparisons of lower bounds and LR gaps for Multi-LSB instances

Test	MC vs.	1st LR vs. MC		FL vs.	2nd LR vs. FL		Gaps	
Set	ℓ, S	LB	UB	ℓ, S	LB	UB	1st LR	2nd LR
SET1	0.02%	-0.02%	-0.02%	0.85%	-0.29%	-0.28%	0.00%	0.01%
SET2	0.06%	-0.06%	-0.06%	0.28%	-0.11%	-0.05%	0.00%	0.06%
SET3	6.28%	-4.27%	-4.27%	6.11%	-5.14%	24.83%	0.00%	21.92%
SET4	1.23%	-1.14%	-1.14%	3.40%	-0.99%	4.34%	0.00%	4.76%

On the other hand, FL improves in general the (ℓ, S) bound more significantly than MC, although the improvements are still minuscule. Note that 2nd LR does not solve

to optimality for many test instances, particularly for the hard problems. Similar to the 1st LR, the 2nd LR does not provide necessarily an improvement over FL bound, due to poor multipliers. Compared to previous test problems, Multi-LSB instances are parallel to TDS problems, where the bottleneck lies in the capacities rather than the multi-level structure of these problems.

4.4 Concluding Remarks

In this chapter, we have discussed different methodologies for obtaining lower bounds and presented both theoretical and computational comparisons of them.

One of our main goals in this chapter was to understand the structure of production planning problems and the underlying difficulties that make these problems very hard. In general, the Lagrangian relaxations we tested are helpful for this. First of all, recall that in general the Lagrangian relaxation that relaxes capacity constraints provides only slight improvement over the (ℓ, S) bound. This bound is an approximation for the uncapacitated problem polyhedron, which indicates that removing capacities makes the problem much easier. This can also be observed by recalling that the final gaps after the default times were quite small for this Lagrangian relaxation in general.

On the other hand, the facility location reformulation with cover cuts and the Lagrangian that relaxes the level-linking constraints improve the lower bounds much more significantly. Recall that the cover cuts approximate the intersection of all knapsack sets included in the problem, and 2nd LR is an approximation for a single-level capacitated problem. Having higher duality gaps compared to the 1st LR, this Lagrangian relaxation problem is in general much harder to solve, indicating that the level-linking constraints

are not the bottleneck of these problems. A similar comparison is achieved by Jans and Degraeve [2004] for single-level problems, where their Lagrangian relaxation relaxing only period-linking constraints is a harder problem than the one that relaxes capacities. Recall that we did not report computational results on LD_3, due to the result presented in Corollary 4.9.

In summary, it seems that the multi-level structure by itself makes some of our problems challenging to solve. However, for most instances, and in particular for the most challenging, the single-level, capacitated substructures are clearly a much greater contributor to problem difficulty. It is this substructure for which the tools currently at our disposal are evidently not sufficient.

These observations indicate that the main bottleneck with these problems lies in the fact that there is no efficient polyhedral approximation of the multi-item, multi-period, single-level, single-machine capacitated problems. It seems that if we could solve these problems well or even adequately, our ability to solve multi-level big bucket problems would increase dramatically. While initial efforts to find strong formulations for these problems have been made (e.g. see Miller et al. [2000]), this is a fundamental area in which it is crucial for the research community to improve the current state of the art. This is the main motivation for the proposed methodology in the next chapter that aims to achieve an adequate approximation of these problems.

Chapter 5

Improving Lower Bounds

Previous computational results in the literature and in this thesis have indicated high duality gaps for big bucket production planning problems, even though some strategies can be partially efficient for generating lower bounds and feasible solutions. The study accomplished in the previous chapter has provided us important feedback on why these problems are still hard to solve after decades of studies and different approaches. More specifically, we have concluded that better approximations for the convex hull of the single-machine single-level capacitated problems are necessary to accomplish better results on general production problems. In this chapter, we will investigate how we can use this information for improving lower bounds, particularly in a way that does not require reformulating the problems, which in general result in either inefficient problem sizes or limitation for a general model. Moreover, we are not aware of strong inequalities or reformulations for these problems. Therefore, we also have the motivation to use the proposed framework of this chapter to obtain these characteristics.

The recent study of Atamtürk and Muñoz [2004] approaches the single-item capacitated lot-sizing problem by formulating it as a bottleneck flow network problem. This allows the authors to define facet-defining inequalities, which could be used to cut off all the fractional extreme points of the LP relaxation of this particular formulation. This specific formulation, generalized to multi-level structure, is in our interest and it is one

of the main motivations for the simple formulation we will use in our framework. More details will follow.

Another motivation for the proposed framework in this chapter is that 2-period problems are computationally easy to solve. This is particularly our experience from the relax-and-fix heuristic we proposed in Chapter 3. However, note that our main aim here is not to provide a computationally efficient and fast method, but it is rather to investigate further production planning problems and to accomplish a better understanding of these problems.

As we have noted before in Section 2.2.4, the recent literature on closures provides quite promising results. Even partially achieving some elementary closures have helped many researchers to be able to close duality gaps very efficiently and solve some problems that were never solved before (see e.g. Balas and Saxena [2007]). Different than these approaches that approximate the closures related to some sets of valid inequalities, our interest is into convex hull closures for some subproblems of the production planning models. In particular, we generate all the valid inequalities for the convex hulls of two period subproblems. Hence, we call the methodology we will discuss in this chapter as "2-Period Convex Hull Closure".

In the next section, we will give a detailed overview of the 2-period convex hull closure methodology, starting from the basic idea and developing the crucial theorems that build the basics of the framework. We will discuss both the Manhattan distance and the Euclidean distance approaches, for which the general framework will be the same. In Section 5.2, we will discuss in detail how to define 2-period submodels of a more complicated, larger problem, which is a crucial part of applying the framework to instances of our test set of big bucket problems. Then, we will present our computational

results concerning some test problems, starting with simple problems and continuing with more general test problems. We will conclude the chapter with a discussion of how the methodology can be improved and how it can be extended.

5.1 Separation Over the 2-Period Convex Hull

In this section, we will first give the basic idea of the proposed framework and how it relates to the problem we are trying to approximate for better lower bounds. Then, a detailed description, along with theoretical results that support the validity of the framework, will follow. We will give special emphasis on the column generation as it is a crucial component of the framework.

5.1.1 Basic Idea

The idea of the framework can be summarized briefly as the separation of a fractional solution obtained from the LP relaxation over the convex hull of the 2-period problem. Column generation will be used to generate the favorable extreme points of each 2-period single-machine production planning problem until no such point is found or until the LPR solution is proven to be in the convex hull of that 2-period problem. In the rest of this thesis, we will refer to the feasible region of the 2-period problem as X^{2PL}, which can be defined as follows:

$$x^i_{t'} \leq \widetilde{M}^i_{t'} y^i_{t'} \qquad i = [1, ..., NI], t' = 1, 2 \tag{5.1}$$

$$x^i_{t'} \leq \widetilde{d}^i_{t'} y^i_{t'} + s^i \qquad i = [1, ..., NI], t' = 1, 2 \tag{5.2}$$

$$x^i_1 + x^i_2 \leq \widetilde{d}^i_1 y^i_1 + \widetilde{d}^i_2 y^i_2 + s^i \qquad i = [1, ..., NI] \tag{5.3}$$

$$x_1^i + x_2^i \leq \widetilde{d}_1^i + s^i \qquad i = [1, ..., NI] \tag{5.4}$$

$$\sum_{i=1}^{NI} (a^i x_{t'}^i + ST^i y_{t'}^i) \leq \widetilde{C}_{t'} \qquad t' = 1, 2 \tag{5.5}$$

$$x, s \geq 0, y \in \{0, 1\}^{2\text{x}NI} \tag{5.6}$$

This formulation is a multi-item extension of the bottleneck flow formulation studied by Atamtürk and Muñoz [2004] when $NT = 2$. It also extends the single-period study of Miller et al. [2000, 2003]. Note that since we are looking at a single-machine problem, we omitted the sub- and superscripts k representing machines for the sake of the simplicity of the formulation. Also, since we consider only one stock variable for each item, a time subscript t is not necessary for these variables. The parameters are defined in a similar fashion to the original parameters of the original model, although it is important to note that the parameter $\widetilde{d}$ represents the remaining cumulative demand, e.g., $\widetilde{d}_1^i$ is the demand for i in periods 1 and 2. One can easily observe the similarity between the constraints of X^{2PL} and of the original production planning problem, with noting that constraints (5.2) and (5.3) are simply the (ℓ, S) inequalities we previously discussed. Further discussion of how exactly to define submodels of this form within a more complicated problem and its parameters will be accomplished in Section 5.2 with some other related issues.

Column generation will be used to generate the favorable extreme points of $conv(X^{2PL})$, since the number of all extreme points can be extremely large even for small problems, as we will see later from some examples. Once we generate all the favorable extreme points, we check whether the LPR solution can be written as a convex combination of these or not. If not, we will generate a valid inequality using theory based on duality, particularly on Farkas' Lemma, that cuts off the fractional solution.

One remark is that this framework is not based on defining a family of valid inequalities, which is its advantage. An inequality will be generated in all cases when the LPR solution is not in the convex hull of a 2-period problem. This is also the justification for our expectation that this framework will provide an adequate approximation of the bottleneck in the production planning problems, as this is focused on the capacitated single-machine problems with an approach providing exact solutions for the subproblems.

5.1.2 Overview of the Method

To describe the methodology, let $(\bar{x}, \bar{y}, \bar{s})$ be a solution obtained from the LPR of X^{2PL}. Then, we can define the "Manhattan distance" of this solution to the convex hull of X^{2PL} as follows:

$$\min\ z = \sum_i [(\Delta_s^-)^i + \sum_{t'=1}^{2} (\Delta_x^+)^i_{t'} + (\Delta_x^-)^i_{t'} + (\Delta_y^+)^i_{t'} + (\Delta_y^-)^i_{t'}] \tag{5.7}$$

$$\text{s.t. } \bar{x}^i_{t'} = \sum_k \lambda_k (x_k)^i_{t'} + (\Delta_x^+)^i_{t'} - (\Delta_x^-)^i_{t'} \qquad \forall i, t' = 1, 2 \qquad (\alpha^i_{t'}) \tag{5.8}$$

$$\bar{y}^i_{t'} = \sum_k \lambda_k (y_k)^i_{t'} + (\Delta_y^+)^i_{t'} - (\Delta_y^-)^i_{t'} \qquad \forall i, t' = 1, 2 \qquad (\beta^i_{t'}) \tag{5.9}$$

$$\bar{s}^i \geq \sum_k \lambda_k (s_k)^i - (\Delta_s^-)^i \qquad \forall i \qquad (\gamma^i) \tag{5.10}$$

$$\sum_k \lambda_k \leq 1 \qquad (\eta) \tag{5.11}$$

$$\lambda_k \geq 0,\ \Delta \geq 0 \tag{5.12}$$

Here, note that (x_k, y_k, s_k) is the vector representing the k^{tk} extreme point of X^{2PL}, and these will be generated using column generation, as explained in the next section.

Variables Δ represent the distance, where the superscript indicates positive (+) or negative (-) distance, and the subscript indicates which original variable (x, y, or s) the distance is defined for. λ variables are multipliers used for the extreme points, and (5.11) ensures convex combinations of the extreme points. Also note that the formulation above has the associated dual variables written next to all constraints in the parentheses, as these play a crucial role in the framework as we will discuss later.

Note that we do not define Δ_s^+ variables, thanks to the following property:

Proposition 5.1 *X^{2PL} has NI extreme rays, each of which has the form:* $s^i = 1, s^j = 0, x = 0, y = 0$ *for each* $i \in [1, ..., NI], j \neq i$.

If there is a positive distance on any s^i, this will be covered by the associated extreme ray and since we are minimizing the sum of all distance variables, Δ_s^+ can be simply omitted. On the other hand, these positive extreme rays provide us the simplicity of writing inequalities (5.10) instead of equations of the form

$$\bar{s}^i = \sum_k \lambda_k (s_k)^i - (\Delta_s^-)^i + \mu^i$$

where μ^i represents the nonnegative multiplier of the extreme ray i.

As one can observe, this problem is always feasible, since we can assign a value of 0 for all the λ variables and assign the values of the associated variables from the LPR solution to the Δ^+ variables. On the other hand, the problem is bounded, as $\Delta \geq 0$ always holds. Therefore, we will always have an optimal solution for this problem, which also holds for the dual of the problem because of the strong duality theorem.

If the optimal solution of this problem for any $(\bar{x}, \bar{y}, \bar{s})$ has an objective function value $z = 0$, then we know that $(\bar{x}, \bar{y}, \bar{s}) \in conv(X^{2PL})$, since all the Δ values have

to be 0 and hence this point can be written as a convex combination of the extreme points and nonnegative amounts of the extreme rays. On the other hand, if $z > 0$, then $(\bar{x}, \bar{y}, \bar{s}) \notin conv(X^{2PL})$, and this allows us to generate a valid inequality to cut off this fractional point, as will be discussed next.

Next, we present the dual of the distance problem:

$$\max \sum_{i=1}^{NI} \sum_{t'=1}^{2} (\bar{x}^i_{t'} \alpha^i_{t'} + \bar{y}^i_{t'} \beta^i_{t'}) + \sum_{i=1}^{NI} \bar{s_k}^i \gamma^i + \eta \tag{5.13}$$

$$\text{s.t.} \sum_{i=1}^{NI} \sum_{t'=1}^{2} ((x_k)^i_{t'} \alpha^i_{t'} + (y_k)^i_{t'} \beta^i_{t'}) + \sum_{i=1}^{NI} (s_k)^i \gamma^i + \eta \leq 0 \qquad \forall k \tag{5.14}$$

$$-1 \leq \alpha^i_{t'} \leq 1 \qquad \forall i, t' = 1, 2 \tag{5.15}$$

$$-1 \leq \beta^i_{t'} \leq 1 \qquad \forall i, t' = 1, 2 \tag{5.16}$$

$$-1 \leq \gamma^i \leq 0 \qquad \forall i \tag{5.17}$$

$$\eta \leq 0 \tag{5.18}$$

The following theorem is the main result that addresses the issue of generating inequalities if $(\bar{x}, \bar{y}, \bar{s}) \notin conv(X^{2PL})$.

Theorem 5.2 *Let $z > 0$ for $(\bar{x}, \bar{y}, \bar{s})$, and let $(\bar{\alpha}, \bar{\beta}, \bar{\gamma}, \bar{\eta})$ be the optimal dual values. Then,*

$$\sum_{i=1}^{NI} \sum_{t'=1}^{2} (\bar{\alpha}^i_{t'} x^i_{t'} + \bar{\beta}^i_{t'} y^i_{t'}) + \sum_{i} \bar{\gamma}^i s^i \leq -\bar{\eta} \tag{5.19}$$

is a valid inequality for $conv(X^{2PL})$ that cuts off $(\bar{x}, \bar{y}, \bar{s})$.

Proof. From (5.14), we know that

$$\sum_{i=1}^{NI} \sum_{t'=1}^{2} (\bar{\alpha}^i_{t'} (x_k)^i_{t'} + \bar{\beta}^i_{t'} (y_k)^i_{t'}) + \sum_{i=1}^{NI} \bar{\gamma}^i (s_k)^i + \bar{\eta} \leq 0 \tag{5.20}$$

is true for all extreme points k of $conv(X^{2PL})$. On the other hand, for any point $(x^*, y^*, s^*) \in conv(X^{2PL})$, we should investigate

$$\sum_{i=1}^{NI}\sum_{t'=1}^{2}(\bar{\alpha}^i_{t'}(x^*)^i_{t'} + \bar{\beta}^i_{t'}(y^*)^i_{t'}) + \sum_i \bar{\gamma}^i(s^*)^i + \bar{\eta} \tag{5.21}$$

We know that (x^*, y^*, s^*) can be written as a convex combination of extreme points and nonnegative amounts of extreme rays. Therefore, we can rewrite the expression in (5.21) as follows (after rearranging terms):

$$\sum_k \lambda_k[\sum_{i=1}^{NI}\sum_{t'=1}^{2}((x_k)^i_{t'}\bar{\alpha}^i_{t'} + (y_k)^i_{t'}\bar{\beta}^i_{t'}) + \sum_{i=1}^{NI}(s_k)^i\bar{\gamma}^i + \bar{\eta}] + \sum_{i=1}^{NI}\mu_i\bar{\gamma}^i \tag{5.22}$$

Since $\bar{\gamma} \leq 0$ and $\mu \geq 0$, the expression in (5.22) is less than or equal to

$$\sum_k \lambda_k[\sum_{i=1}^{NI}\sum_{t'=1}^{2}((x_k)^i_{t'}\bar{\alpha}^i_{t'} + (y_k)^i_{t'}\bar{\beta}^i_{t'}) + \sum_{i=1}^{NI}(s_k)^i\bar{\gamma}^i + \bar{\eta}] \tag{5.23}$$

Since $\lambda \geq 0$ and (5.20) hold, this expression is less than or equal to 0. This shows that (5.19) is a valid inequality for $conv(X^{2PL})$. Finally, using (5.13) and strong duality, we know that

$$\sum_{i=1}^{NI}\sum_{t'=1}^{2}(\bar{x}^i_{t'}\bar{\alpha}^i_{t'} + \bar{y}^i_{t'}\bar{\beta}^i_{t'}) + \sum_{i=1}^{NI}\bar{s_k}^i\bar{\gamma}^i + \bar{\eta} = z > 0$$

Hence, (5.19) cuts off $(\bar{x}, \bar{y}, \bar{s})$. This concludes the proof. □

Here, we handled the proof as if we were aware of the all extreme points of $conv(X^{2PL})$. As we will show in the next section, we will only generate a small subset of the extreme points to provide as much computational efficiency as possible. The general framework of the separation procedure over 2-period convex hull can be summarized as in Figure 5.1, where z_P refers to the objective function of the column generation problem, which will be explained in detail in the next section.

repeat
  Solve the distance problem for $conv(X^{2PL})$
  if z = 0 **then** break
    else Solve column generation problem
    if $z_P \leq 0$ **then** break
      else Add new extreme point

until z = 0 **or** $z_P \leq 0$
if z=0 **then** $(\bar{x}, \bar{y}, \bar{s}) \in conv(X^{2PL})$
  else Add the violated cut (5.39)

Figure 5.1: Separation of a point $(\bar{x}, \bar{y}, \bar{s})$ over the 2-period convex hull

5.1.3 Column Generation

Column generation is a method particularly useful for problems where the number of variables is much more than the number of constraints. The basic idea of the methodology is to generate "columns", i.e., the variables, one by one, in order to prevent the problem becoming too big for the solver to handle. The main logic behind the method is to generate all the possible "favorable" columns, which are expected to improve the objective function value of the problem in consideration. When no more favorable columns exist, the current solution is optimal, hence resulting in a much smaller problem than the original problem. Deciding on whether a column is favorable or not is done by calculating the reduced costs of variables. Naturally, one does not calculate reduced costs for all the variables, but instead simply finds the maximum or minimum of those (depending whether it is a max or min problem we are trying to solve) and observes if this satisfies the condition of being either bigger or smaller than zero (again dependent on the original problem). For more general information on column generation, please refer to Wolsey [1998].

In our case, we have the Manhattan distance minimization problem, which has an exponential number of columns, namely the extreme points of $conv(X^{2PL})$. As we will see from some of the examples presented in the next section, even very small problems might have an enormous number of extreme points. Hence, we do not want to deal with all these extreme points since it might easily exceed the limitations of any solver, and column generation will be used to generate the favorable extreme points, as explained below.

First of all, we want to have a closer look at the reduced cost of each extreme point. When we solve the problem with the currently available extreme points, we obtain a dual optimal solution $(\bar{\alpha}, \bar{\beta}, \bar{\gamma}, \bar{\eta})$. Then, for any extreme point (x_k, y_k, s_k), the reduced cost can be calculated as

$$\sum_{i=1}^{NI} \sum_{t'=1}^{2} (\bar{\alpha}^i_{t'} (x_k)^i_{t'} + \bar{\beta}^i_{t'} (y_k)^i_{t'}) + \sum_i \bar{\gamma}^i (s_k)^i + \bar{\eta} \tag{5.24}$$

Recall from the proof of Theorem 5.2 that for all the currently available extreme points, (5.24) is less than or equal to 0. This condition clearly holds since we are solving exactly the same primal distance problem, only with the currently available extreme points instead of all extreme points. One can simply observe that if this condition holds for all the other extreme points not yet included in the problem, than we do not have any favorable extreme point that will improve the solution of the previous problem and hence that is optimal. Therefore, we are searching for extreme points that have a strictly positive value for (5.24), and we will be maximizing this value over all the extreme points since that is only in our interests: If the maximum value over all extreme points is not strictly positive, then this indicates that all other extreme points already satisfy this condition; if not, then we will generate only this column with the highest

positive objective function value since it is the most favorable column. This basically results in the following pricing problem:

$$\max z_P = \sum_{i=1}^{NI} \sum_{t'=1}^{2} (\bar{\alpha}^i_{t'} x^i_{t'} + \bar{\beta}^i_{t'} y^i_{t'}) + \sum_i \bar{\gamma}^i s^i + \bar{\eta}$$

$$\text{s.t. } (x, y, s) \in X^{2PL}$$

Corollary 5.3 *If the optimal value $z_P \leq 0$, then the solution of the distance problem is optimal. Otherwise, the optimal (x, y, s) values should be added as a new column to the distance problem.*

Note that this is an integer problem. However, as previously discussed, these problems are generally very easy to solve. On the other hand, since we cannot know beforehand how many iterations of column generation will be needed for a problem, this is the major drawback for this framework to become a computationally efficient scheme. However, as noted before, we are studying this framework for approximating the single-machine multi-item capacitated problem convex hulls, and hence this is more of a theoretical approach than practical. Finally, note that we will discuss some possibilities to improve this part of the framework in the last section of this chapter.

5.1.4 Euclidean Distance

We have so far discussed the modeling of the distance of the LP relaxation solution to the convex hull of X^{2PL} in the linear fashion using the Manhattan distance. Now we will discuss how to model it using Euclidean distance. The main motivation for the Euclidean distance approach is that it has in general a faster convergence compared to the Manhattan distance approach (Robinson [2007]). The general framework and

approach is the same as the one discussed in Figure 5.1; however we have to establish some theoretical results and make new statements that fit the nature of the Euclidean distance problem.

As in the previous sections, let $(\bar{x}, \bar{y}, \bar{s})$ be the solution obtained from the LPR of the original problem. Then, we define the "Euclidean distance" problem as follows:

$$\min_{\Delta,\lambda} \; z = \sum_i \left[[(\Delta_s)^i]^2 + \sum_{t'=1}^{2} \left([(\Delta_x)^i_{t'}]^2 + [(\Delta_y)^i_{t'}]^2 \right) \right] \tag{5.25}$$

$$\text{s.t. } \bar{x}^i_{t'} = \sum_k \lambda_k (x_k)^i_{t'} + (\Delta_x)^i_{t'} \qquad \forall i, t' = 1, 2 \qquad (\alpha^i_{t'}) \tag{5.26}$$

$$\bar{y}^i_{t'} = \sum_k \lambda_k (y_k)^i_{t'} + (\Delta_y)^i_{t'} \qquad \forall i, t' = 1, 2 \qquad (\beta^i_{t'}) \tag{5.27}$$

$$\bar{s}^i \geq \sum_k \lambda_k (s_k)^i - (\Delta_s)^i \qquad \forall i \qquad (\gamma^i) \tag{5.28}$$

$$\sum_k \lambda_k \leq 1 \qquad (\eta) \tag{5.29}$$

$$\lambda_k \geq 0, \; \Delta_s \geq 0, \; \Delta_x, \Delta_y \text{ free} \tag{5.30}$$

The variables and constraints of the Euclidean distance problem are almost identical to the Manhattan distance and hence we will only mention the differences here. First of all, note that the distance variables Δ_x and Δ_y are defined as free variables for the simplicity of the model and the superscripts + and - are skipped. This does not affect the objective function value in this case since all the distance variables are squared. Other than that, note that the Proposition 5.1 is also valid in this case. The associated dual variables are written next to all constraints in the parentheses.

Similar to the Manhattan distance problem, this problem is also always feasible, since we can assign a value of 0 for all the λ variables and assign the values of the associated

variables from the LPR solution to the Δ variables. On the other hand, the problem is bounded, since the objective function value can attain 0 as lowest value when all $\Delta = 0$. Hence, we will always have an optimal solution for this problem. Note that this is a simple quadratic programming (QP) problem with linear constraints, i.e., a convex set, and the objective function has quadratic terms with positive coefficients only, i.e., if we write z in the form $\frac{1}{2}x^T Qx$ (x indicating the vector with all the variables), then the matrix Q is positive semidefinite. Thanks to these nice properties, we will next make use of some basic nonlinear programming theory to establish the necessary results of this section.

Here, we present the dual of the Euclidean distance problem (see e.g. Mangasarian [1994] for QP duality):

$$\max_{\Delta,\alpha,\beta,\gamma} z_D = -\sum_i \left[[(\Delta_s)^i]^2 + \sum_{t'=1}^{2} [(\Delta_x)^i_{t'}]^2 + [(\Delta_y)^i_{t'}]^2 \right]$$
$$- \left(\sum_{i=1}^{NI} \sum_{t'=1}^{2} (\bar{x}^i_{t'} \alpha^i_{t'} + \bar{y}^i_{t'} \beta^i_{t'}) + \sum_{i=1}^{NI} \bar{s}^i \gamma^i + \eta \right) \tag{5.31}$$

$$\text{s.t.} \quad \sum_{i=1}^{NI} \sum_{t'=1}^{2} ((x_k)^i_{t'} \alpha^i_{t'} + (y_k)^i_{t'} \beta^i_{t'}) + \sum_{i=1}^{NI} (s_k)^i \gamma^i + \eta \geq 0 \qquad \forall k \tag{5.32}$$

$$\alpha^i_{t'} = -2(\Delta_x)^i_{t'} \qquad \forall i, t' \tag{5.33}$$

$$\beta^i_{t'} = -2(\Delta_y)^i_{t'} \qquad \forall i, t' \tag{5.34}$$

$$-\gamma^i \geq -2(\Delta_s)^i \qquad \forall i \tag{5.35}$$

$$\gamma \geq 0, \ \eta \geq 0, \ \Delta_s \geq 0, \ \alpha, \beta, \Delta_x, \Delta_y \text{ free} \tag{5.36}$$

Before stating the main result of this section, we will state the following result that is particularly adapted for our dual problems from the strong duality result for quadratic programming with linear constraints, with the fact that Q is positive semidefinite. Let

(PP) and (DP) represent the Euclidean distance problem (primal problem) and its dual, respectively.

Proposition 5.4 *Let $(\bar{\Delta}_x, \bar{\Delta}_y, \bar{\Delta}_s, \bar{\lambda})$ be an optimal solution to PP. Then, $(\bar{\Delta}_x, \bar{\Delta}_y, \bar{\Delta}_s)$ and some $(\bar{\alpha}, \bar{\beta}, \bar{\gamma}, \bar{\eta})$ solve DP, and $z = z_D$ for these solutions.*

The following theorem is the main result associated with the Euclidean distance problem and how to generate inequalities if $(\bar{x}, \bar{y}, \bar{s}) \notin conv(X^{2PL})$.

Theorem 5.5 *Let $z > 0$ for $(\bar{x}, \bar{y}, \bar{s})$, with optimal primal values $(\bar{\Delta}_x, \bar{\Delta}_y, \bar{\Delta}_s, \bar{\lambda})$, and $(\bar{\alpha}, \bar{\beta}, \bar{\gamma}, \bar{\eta})$ be the associated optimal dual values. Then,*

$$\sum_i \sum_{t'=1}^{2} (\bar{\alpha}^i_{t'} x^i_{t'} + \bar{\beta}^i_{t'} y^i_{t'}) + \sum_i \bar{\gamma}^i s^i + \bar{\eta} \geq 0 \tag{5.37}$$

is a valid inequality for $conv(X^{2PL})$ that cuts off $(\bar{x}, \bar{y}, \bar{s})$.

Proof. First, for any point $(x^*, y^*, s^*) \in conv(X^{2PL})$, we know that it can be written as a convex combination of extreme points and nonnegative amounts of extreme rays (recall the proof of Theorem 5.2 for detailed expression). Since (5.37) is a valid inequality for all the extreme points of $conv(X^{2PL})$ and since $\bar{\gamma} \geq 0$, we can conclude the validity of the inequality for $conv(X^{2PL})$.

For the violation proof, note that

$$\left(\sum_{i=1}^{NI} \sum_{t'=1}^{2} (\bar{x}^i_{t'} \bar{\alpha}^i_{t'} + \bar{y}^i_{t'} \bar{\beta}^i_{t'}) + \sum_{i=1}^{NI} \bar{s}^i \bar{\gamma}^i + \bar{\eta} \right) = -2 \left[(\bar{\Delta}_s)^2 + (\bar{\Delta}_x)^2 + (\bar{\Delta}_y)^2 \right] < 0$$

where the equality follows from $z = z_D$ as stated in the Proposition 5.4 and inequality follows $z > 0$. Hence, (5.37) cuts off $(\bar{x}, \bar{y}, \bar{s})$. This concludes the proof. □

To avoid dealing with an exponential number of extreme points, we will again use column generation to generate only a small subset of useful extreme points, as we did in

the previous section (see e.g. Pang [1981] for column generation in QP). To define the pricing problem, first note that any extreme point that already satisfies (5.37) is not in our interest. Hence, the pricing problem can be stated as follows:

$$\min\ z_P = \sum_{i=1}^{NI}\sum_{t'=1}^{2}(\bar{\alpha}^i_{t'}x^i_{t'} + \bar{\beta}^i_{t'}y^i_{t'}) + \sum_i \bar{\gamma}^i s^i + \bar{\eta}$$

$$\text{s.t. } (x, y, s) \in X^{2PL}$$

This leads us to a similar conclusion as in the previous section, which is presented next.

Corollary 5.6 *If the optimal value $z_P \geq 0$, then the column generation is finished and the solution of the distance problem is optimal. Otherwise, the optimal (x, y, s) values of the pricing problem should be added as a new column to the distance problem.*

Note that the framework presented in Figure 5.1 is valid for using the Euclidean approach as well, with only the exception of changing the $z_P \leq 0$ condition to $z_P \geq 0$.

We will provide computational results in Section 5.3, in particular the comparisons between the two distance problems, after discussing in the next section how to define the 2-period problems.

5.2 Defining 2-Period Subproblems

We have defined the feasible region of the generic 2-period problem as X^{2PL}, however we did not discuss exactly how to define these problems, including its parameters. Considering the production planning problems we have investigated in this thesis with multiple levels, periods, items and machines, the first question one can ask is at which two periods to run the separation algorithm. For a problem with NT periods, we can look at

all the 2-period problems, i.e., we can create $NT-1$ 2-period problems and run our separation routine we discussed in the previous sections. Next, recall that we have only one stock variable for each item, namely s^i. This leads to the question "which period's stock is represented by s^i?". Therefore, for each item i, we will define a horizon parameter $\phi(i)$, as it is not necessary for us to use the same horizon for all items, where $\phi(i) \in [t+1, .., NT]$.

As one can easily note, the obvious choice for the horizon would be $t+1$, i.e., $s^i = s^i_{t+1}$. In this case, the definition of the parameter $\widetilde{M}^i_{t'}$ of (5.1) is the same as of $M^i_{t+t'-1}$ from (1.14), for all i and $t' = 1, 2$. Similarly, capacity parameter $\widetilde{C}^i_{t'}$ of (5.5) is the same as $C^i_{t+t'-1}$, for all i and $t' = 1, 2$. Cumulative demand parameter $\widetilde{d}^i_{t'}$ represents simply $d^i_{t+t'-1,\, t+1}$, for all i and $t' = 1, 2$, i.e., $\widetilde{d}^i_1 = d^i_{12}$ and $\widetilde{d}^i_2 = d^i_2$.

The main disadvantage of assigning $\phi(i) = t+1$ for all i is that later periods are not taken into consideration. For example, consider a case where there does not occur any production in periods $t+2, ..., t'$. Then, the variable s^i will be associated with a big number and the formulation will be weak, whereas it could be associated with a smaller number if we choose the horizon to be t'. By definition, we can select different values for horizon parameters to allow consideration of inventory carried to later periods. The horizon parameters play an important role particularly on the (ℓ, S) inequalities of X^{2PL} (recall (5.2) and (5.3)) and we want to make sure these inequalities are as efficient as possible. As Miller et al. [2000] noted for their single-period relaxations, one key observation is that if a number of periods have no setups following the period $t+1$, their demands should be incorporated to obtain the smallest amount of inventory carried from period $t+1$ without weakening the (ℓ, S) inequalities. Another observation is that if a setup occurs in a period after $t+1$, the (ℓ, S) inequalities will be weakened if that period

is included in the horizon and hence it should be avoided. Following Miller et al. [2000] we define

$$\phi(i) = \max\{t' | t' \geq t+1, \sum_{t''=t+1}^{t'} y^i_{t''} \leq y^i_{t+1} + \Theta\} \tag{5.38}$$

where Θ is a random number between 0 and 1. As Miller et al. [2000] discuss, this assignment works efficiently in case of their single period relaxation.

Note that X^{2PL} is used to define the feasible region of the generic 2-period problem, and in case we want to refer to the feasible region of the specific 2-period problem from t to $t+1$, we will use a subscript and call it as X^{2PL}_t. One final note on the notation is that since different horizon possibilities for different items would result in a large number of different combinations of horizons, X^{2PL}_t will be used to refer to $X^{2PL}_t(\phi(1), \phi(2), ..., \phi(NI))$, where $\phi(i)$ values will be set as previously discussed. Every time the separation procedure is called, horizon parameters will be assigned for all the items and hence X^{2PL}_t redefined.

Solve LPR of the original problem
$\rightarrow (\bar{x}, \bar{y}, \bar{s})$
for t=1 **to** NT-1
  Define X^{2PL}_t
  Apply 2-period convex hull separation algorithm

Figure 5.2: 2-period convex hull closure framework

Finally, note that the definition of parameters are the same as when $t+1$ is assigned as horizon, except that cumulative demand parameter $\widetilde{d}^i_{t'}$ now represents $d^i_{t+t'-1,\ \phi(i)}$, for all i and $t' = 1, 2$.

5.3 Computational Results

In this section, we will present our computational experience regarding the 2-period convex hull closure framework. While doing so, our focus is to provide detailed results on a representative set of problems rather than summary results on large data sets. The particular reason for this is that we are trying to understand the structure of these difficult problems and to investigate the possibility of defining valid inequalities for the big-bucket structure.

The separation algorithm and the 2-period convex hull closure framework are both implemented using the Mosel language of Xpress-MP (Mosel version 2.0.0, Xpress 2007 package). Both Manhattan distance and Euclidean distance approaches are implemented. We first present results for 2-period problems in the following section, and then the results for some multi-period problems from the literature follow, and we discuss these results in detail.

5.3.1 2-Period Problems

As previously discussed, for a production planning problem with only two periods, we seek to solve a separation problem for the LP relaxation solution. In this section, we will focus on the cuts generated by the separation algorithm, some computational challenges and issues will be briefly addressed as well.

In order to be able to provide a thorough investigation, we generated 20 problems with two periods only and with two to six items. The detailed data of the instances of this set, called 2PCLS (2-Period Capacitated Lot-Sizing) is provided in Appendix D, where the last column indicates whether or not we let the final inventory variables take

any values other than zero (In case of zeros, these variables can be simply eliminated from the problems). One of the advantages of having such toy problems is that we might actually obtain the full description of the convex hull. Another important remark is on the number of items: The more items share a resource, the more the structure tends to resemble that of an uncapacitated problem. This is another motivation not to generate problems with too many items, in addition to keeping simplicity.

A very useful tool for obtaining the full description of the convex hull is the cdd package of Fukuda [2005]. This is an implementation of the double description method of Motzkin et al. [1953], which can calculate all the extreme points and rays of a polyhedron given by inequalities or all the facets of a polyhedron given by extreme points and rays. We tested this tool for most of the 2-period problems we generated, except for problems with four or more items, due to impractical running times and memory problems experienced for these problems, as the running time grows exponentially. For example, for the instance 2pcls13, roughly the first 5000 extreme points were processed in 4 hours CPU time, whereas processing of the next 30 extreme points took more than 24 hours CPU time. Sample cdd input and output files for two problems are provided in Appendix E for the interested reader. The main use of cdd for our purposes is as follows: We first generate all the extreme points and extreme rays of the LP relaxation of an instance. Next, we eliminate all the fractional extreme points and therefore we only have only all the integral extreme points. One important note is that since all our integer variables are binary, eliminating fractional points actually gives us all the extreme points of the integral polyhedron. Finally, using those extreme points, we can generate all the facets of this integral polyhedron. This will give us the advantage of comparing these facets to the cuts generated by our separation procedure. Example 5.7

illustrates a detailed case where we did such a comparison.

Instance	Total	Integer
2pcls01	13,984	2,408
2pcls04	591	180
2pcls08	520	199
2pcls09	136	90
2pcls10	1,777	819
2pcls11	1,448	815
2pcls12	1,496	824
2pcls13	17,440	8,014

Table 5.1: Number of extreme points for some 2PCLS instances

Table 5.1 presents the number of extreme points for some of these test problems, where the column "Total" shows the total number of extreme points obtained from the LP relaxation of the problems, and the column "Integer" indicates the number of extreme points once the fractional points are eliminated. It is interesting that even these small problems have such enormous numbers of extreme points. As we noted before, there are only as many extreme rays as the number of items. Also note that assigning zeros to the final inventory values eliminate a significant number of extreme points, see e.g. 2pcls08 vs. 2pcls09.

Example 5.7 *As we will see later in the computational results, only one inequality generated by our separation framework with Manhattan approach sufficed to close the duality gap for the test problem "2pcls10". This inequality can be written as*

$$\frac{1}{21}x_1^1 + \frac{1}{18}x_1^2 + \frac{1}{19}x_1^3 + \frac{1}{21}x_2^1 + \frac{1}{18}x_2^2 + \frac{1}{19}x_2^3 - y_1^1 - y_1^2 - y_1^3 - s^1 - s^2 - s^3 \leq 3.36424$$

One interesting property about the coefficients of the x variables is that the denominators are the demand amounts in the first period, i.e., d_1^i. Also note that if we produce exactly the demand for all items in the second period, i.e., $x_2^i = d_2^i$, the quantity

$\frac{1}{21}x_2^1 + \frac{1}{18}x_2^2 + \frac{1}{19}x_2^3$ *will be exactly 3.36424.*

Next, we are interested in comparing this inequality with the actual facets of the integral polyhedron. Final inventories are assigned values of zero in this instance, and the output of cdd indicates a total of 90 facets, which we added one by one to the LP relaxation in Xpress. This procedure indicates to us that 8 of these facets can close the duality gap by their own, i.e., adding any one of these facets closes the duality gap on its own, which is our next focus. One should also note that some of the trivial facets of the polyhedron, already present in the LP relaxation, should also be active at the particular extreme points, as only 8 facets cannot provide an extreme point in 12-dimensional space.

Looking at these particular facets, one can simply observe that our inequality is not one of them. One of the facets that has some similarity to our inequality is as follows:

$$x_1^1 + x_1^2 + x_1^3 + x_2^1 + x_2^2 + x_2^3 - 20y_1^1 - 17y_1^2 - 39y_1^3 - y_2^3 \leq 46$$

Here, we can observe that our inequality is a tilted version of this facet, i.e., the coefficients of x variables is the same for all items in the same period on this facet whereas those coefficients are slightly different in our inequality. To observe this, multiply all the coefficients in our inequality with 18, and obtain

$$\frac{18}{21}x_1^1 + x_1^2 + \frac{18}{19}x_1^3 + \frac{18}{21}x_2^1 + x_2^2 + \frac{18}{19}x_2^3 - 18y_1^1 - 18y_1^2 - 18y_1^3 - 18s^1 - 18s^2 - 18s^3$$
$$\leq 60.55632$$

Now, rearrange the terms in the facet as follows:

$$x_1^1 + x_1^2 + x_1^3 + x_2^1 + x_2^2 + x_2^3 - 18y_1^1 - 18y_1^2 - 18y_1^3 \leq 46 + 2y_1^1 - y_1^2 + 21y_1^3 + y_2^3$$
$$\leq 46 + 2y_1^1 - y_1^2 + 21y_1^3 + y_2^3 + 18(s^1 + s^2 + s^3) \leq 69 + 18(s^1 + s^2 + s^3)$$

Note that the second inequality above follows the fact that $s^i \geq 0$, and the last inequality is the result of the second inequality with the following trivial facets:

$$y_1^1 \leq 1, y_1^2 \leq 1, y_1^3 \leq 1, y_2^3 \leq 1$$

Note that the coefficients of the x variables are different, and taking the intersection of this inequality with some trivial facets regarding x variables in a similar fashion provides us similar inequalities. This makes us conclude that our inequality does provide a face of the integral polyhedron. $\square$

Next, we present the summary of computational results with the Manhattan approach for the 2PCLS instances in Table 5.2. The column "i" indicates the number of items for each instance. Columns "XLP" and "IP" indicate strengthened the LP relaxation solution using (ℓ, S) inequalities and the optimal integer solution, respectively. The column "2SEP" indicates the lower bound obtained after our 2-period convex hull separation is terminated. The next column indicates the total number of cuts generated during the process, and finally, the last column indicates number of "Reduced Cost" cuts, i.e., inequalities generated using the reduced cost information. We will address this issue after discussing these results.

As these results clearly indicate, the 2-period convex hull separation algorithm using the Manhattan distance function successfully closed duality gaps for all these problems. In general, this is achieved with a relatively small number of cuts. Also, looking carefully at the cuts generated, there are often similarities between the cuts, as illustrated in Example 5.9. Also, note that problem size is not affecting number of cuts by itself (see e.g. "2pcls02" vs. "2pcls19"), but problem data is quite significant (see e.g. "2pcls19" vs. "2pcls20").

Instance	i	XLP	IP	2SEP	Total # cuts	# RC cuts
2pcls01	3	17.033	25	25	11	-
2pcls02	3	12.6253	19	19	13	-
2pcls03	3	76.5345	104	104	5	-
2pcls04	2	14.7674	19	19	4	-
2pcls05	3	38.39	52	52	8	-
2pcls06	3	117.375	173	173	5	-
2pcls07	2	36.5	43	43	2	-
2pcls08	2	21.45	26	26	7	-
2pcls09	2	129	153	153	2	-
2pcls10	3	17.6539	24	24	1	-
2pcls11	3	71.7209	102	102	4	-
2pcls12	3	46.68	69	69	4	-
2pcls13	4	85.6256	113	113	7	1
2pcls14	4	70.2961	81	81	6	-
2pcls15	4	54.1848	74	74	6	-
2pcls16	4	34.0844	39	39	6	-
2pcls17	5	164.858	211	211	39	6
2pcls18	5	57.0825	97	97	34	3
2pcls19	6	115.131	150	150	11	-
2pcls20	6	59.2412	89	89	34	11

Table 5.2: Separation of 2PCLS instances using Manhattan approach

From our computational experience, one issue has been to determine how many extreme points of a 2-period problem to generate using column generation, which we will refer to as "*lim*" for simplicity in the remainder of this chapter. Assigning a big number for *lim* has the advantage of solving more distance problems, since it is less probable to terminate without a cut. On the other hand, a big *lim* has the disadvantage of making the overall process longer and it is in some cases not even useful due to highly fractional numbers. On the other hand, we would like to generate cuts when we generate *lim* extreme points and still not be finished with the separation, in order to prevent the waste of computational effort. The following corollary (adapted from Theorem 5.2) provides an important result for this purpose for the Manhattan distance

problem. Recall that z represents the value of the distance function, and z_P the value of the pricing problem.

Corollary 5.8 *Let $z > 0$ for $(\bar{x}, \bar{y}, \bar{s})$, $(\bar{\alpha}, \bar{\beta}, \bar{\gamma}, \bar{\eta})$ be the optimal dual values, and $z_P > 0$. Then,*

$$\sum_{i=1}^{NI} \sum_{t'=1}^{2} (\bar{\alpha}^i_{t'} x^i_{t'} + \bar{\beta}^i_{t'} y^i_{t'}) + \sum_i \bar{\gamma}^i s^i \leq -\bar{\eta} + z_P \tag{5.39}$$

is a valid inequality for $conv(X^{2PL})$.

A detailed proof is omitted as the validity of this inequality can be simply proven by using the reduced cost information, particularly since $\sum_k \lambda_k \leq 1$, the distance function value can be at most reduced by z_P. Although this inequality is weaker than the original inequality, it provides us with the advantage of generating cuts even if we terminate the separation due to hitting *lim*. Since these inequalities are using reduced cost information, we refer to those inequalities as "Reduced Cost Cuts" or simply as "RC cuts".

Next, we present another detailed example from our results to highlight our observations on the generated inequalities.

Example 5.9 *In the instance 2pcls04, there are 4 cuts generated. The first cut is*

$$\frac{1}{21}x^1_1 + \frac{1}{19}x^2_1 + \frac{1}{21}x^1_2 + \frac{1}{19}x^2_2 - y^1_1 - y^2_1 - \frac{1}{21}y^1_2 - s^1 - s^2 \leq 1.81704$$

The second cut is

$$\frac{1}{21}x^1_1 + \frac{1}{19}x^2_1 + \frac{1}{21}x^1_2 + \frac{1}{19}x^2_2 - y^1_1 - y^2_1 - \frac{1}{21}y^1_2 - \frac{1}{21}s^1 - s^2 \leq 1.81704$$

Note that this inequality dominates the first inequality because of the coefficient of the s^1 variable. Next, the third cut is

$$\frac{1}{21}x^1_1 + \frac{1}{19}x^2_1 + \frac{1}{21}x^1_2 + \frac{1}{19}x^2_2 - y^1_1 - y^2_1 - \frac{1}{21}y^1_2 + y^2_2 - s^1 - \frac{1}{19}s^2 \leq 2.81704$$

This inequality can be obtained by lifting the y_2^2 into the first inequality. Finally, the last cut is

$$\frac{1}{21}x_1^1 + \frac{1}{19}x_1^2 + \frac{1}{21}x_2^1 + \frac{1}{19}x_2^2 - y_1^1 - y_1^2 - \frac{1}{21}y_2^1 - \frac{1}{21}s^1 - \frac{1}{19}s^2 \leq 1.81704$$

This inequality can be seen as a mixing inequality obtained using the second and third inequalities. Also note that this last inequality suffices to close the duality gap on its own when added manually. □

A similar pattern described in Example 5.9 can be seen in some other problems as well, e.g. the first 4 cuts generated for the 2pcls05 instance are related to each other by lifting and mixing as well. Also note that the denominators of the coefficients of the x variables are demand values of the first period, i.e., d_1^i, similar to Example 5.7, which we observed as a characteristic of the cuts generated using this Manhattan distance approach. A final note on the coefficients of the cuts is that the x variables have relatively smaller coefficients than the y variables, which we did expect due to the significance of these variables.

One computational remark has to be made on the tolerance value, i.e., what number to use as "zero" distance to assume the LPR solution is in the convex hull of the 2-period problem, since zero might cause computational problems due to rounding. In the framework with the Manhattan distance, we used the default zero of Xpress. On the other hand, we assigned 0.000001 as the value of this tolerance parameter in the Euclidean case. Without this tolerance, we observed that some of the instances did not terminate as the framework was trying to generate inequalities to cut off the fractional solution with an infinitesmall distance.

Next, we present and discuss results of the 2PCLS instances that are obtained

through the separation algorithm employing Euclidean distance. Table 5.3 summarizes these results in a similar fashion to the previous table, where the column "2SEP" indicates again the lower bound obtained after the separation is terminated.

Instance	XLP	IP	2SEP	# cuts
2pcls01	17.033	25	25	19
2pcls02	12.6253	19	19	7
2pcls03	76.5345	104	104	1
2pcls04	14.7674	19	19	1
2pcls05	38.39	52	52	4
2pcls06	117.375	173	173	5
2pcls07	36.5	43	43	1
2pcls08	21.45	26	26	2
2pcls09	129	153	153	3
2pcls10	17.6539	24	24	1
2pcls11	71.7209	102	102	1
2pcls12	46.68	69	69	2
2pcls13	85.6256	113	113	9
2pcls14	70.2961	81	81	5
2pcls15	54.1848	74	74	1
2pcls16	34.0844	39	39	4
2pcls17	164.858	211	211	14
2pcls18	57.0825	97	97	6
2pcls19	115.131	150	150	1
2pcls20	59.2412	89	89	5

Table 5.3: Separation of 2PCLS instances using Euclidean approach

As the results indicate, the separation algorithm successfully closed the duality gaps for all these instances, as was the case for the Manhattan distance approach. However, as number of cuts indicate, the Euclidean approach is a more efficient scheme than the Manhattan approach as expected, due to the reduced number of cuts generated, which is particularly true for bigger instances. Although we did not report on the computational times, mainly because of the relative insignificance of the time aspect due to our desire for theoretical understanding, our observation is that the framework with the Euclidean

distance approach is significantly faster than the framework with the Manhattan distance approach.

Example 5.10 *Similar to the Manhattan distance approach, only one inequality generated by the separation framework with Euclidean approach sufficed to close the duality gap for the test problem "2pcls10". This inequality can be written as*

$$0.0494472x_1^1 + 0.057943x_1^2 + 0.0560172x_1^3 + 0.0494472x_2^1 + 0.057943x_2^2$$
$$+0.0560172x_2^3 - 1.03839y_1^1 - 1.04297y_1^2 - 1.06433y_1^3 + 0.0000568y_2^1 + 0.00010624y_2^2$$
$$+0.00011438y_2^3 - 0.0494472s^1 - 0.057943s^2 - 0.0560172s^3 \leq 3.5288$$

Recall from Example 5.7 that only 8 facets can close the duality gap by their own. Looking at these facets, we simply observe that our inequality is not one of them. More interesting is the similarity between this inequality and the one generated by the separation using Manhattan. Although these two inequalities are not exactly same, the proportions of x variables' coefficients to y variables' coefficients is exactly the same for the first period. More specifically, if we multiply all the coefficients of this inequality with $\frac{18}{1.04297}$, we obtain the following equivalent inequality:

$$0.853379867x_1^1 + 1.000003835x_1^2 + 0.966767596x_1^3 + 0.853379867x_2^1 + 1.000003835x_2^2$$
$$+0.966767596x_2^3 - 17.9209565y_1^1 - 18y_1^2 - 18.36863956y_1^3 + 0.000980899y_2^1$$
$$+0.001833533y_2^2 + 0.001974017y_2^3 - 0.853379867s^1 - 1.000003835s^2 - 0.966767596s^3$$
$$\leq 60.90146409$$

Recall the cut generated by the Manhattan approach:

$$\frac{18}{21}x_1^1 + x_1^2 + \frac{18}{19}x_1^3 + \frac{18}{21}x_2^1 + x_2^2 + \frac{18}{19}x_2^3 - 18y_1^1 - 18y_1^2 - 18y_1^3 - 18s^1 - 18s^2 - 18s^3$$
$$\leq 60.55632$$

The main difference between these cuts is the coefficients of s variables, which are significantly smaller in the cut generated by the Euclidean approach. Other coefficients indicate that these inequalities are slightly tilted versions of each other, although it is possible that the Euclidean cut is stronger than the Manhattan cut due to smaller s variable coefficients and inclusion of the y variables from the second period. □

One note is that in case multiple inequalities are generated with the Euclidean approach for a particular instance, those inequalities did not have similarities as Example 5.9 illustrated. Moreover, from our computational experience, there are in general very few cuts that increase the lower bound significantly, to a value very close to IP solution, and many other weak cuts are added until the IP bound is reached. This suggest the use of lifting to obtain stronger inequalities.

5.3.2 Multi-Period Problems

Computational results in the previous section indicated a significant potential for improving the lower bounds on capacitated production planning problems. In this section, we will look into more realistic models with multiple periods that were previously discussed in the literature. Since we are dealing with bigger problems, computational challenges will be addressed in this section.

Computational tests are accomplished mainly on some Trigeiro instances (Trigeiro et al. [1989]), which are single-level problems and have a single machine that is shared by multiple items, where no backlogging is allowed. Compared to the test problems we have used for our computations in previous chapters, these problems are rather easy and can be solved to optimality in acceptable times using the state-of-the-art solvers.

However, as we have noted earlier, our main incentive here is to see how much we can approximate the convex hull of the single-level single-machine capacitated multi-item problems, and therefore these problems will provide a reasonable starting point for our computational tests. Once the framework is successfully tested on these problems, our tests can be extended to more complicated problems.

First of all, note that Appendix F presents the whole Mosel code for the tr6-15 instance (6 items, 15 periods). This is the implementation of the 2-period convex hull closure framework with Manhattan distance approach for a big problem, with the separation algorithm as a subroutine, and horizons are implemented as discussed in Section 5.2. This implementation follows the pseudocode presented in Figure 5.2, i.e., an LPR solution is separated over $NT-1$ 2-period problems. We did not provide the implementation of the Euclidean approach in the thesis due its similarity to the code provided. We will address next some computational issues that can be seen in this code.

Iter	lim=100	lim=150	RCC	Iter	lim=100	lim=150	RCC
1	37203	37203	37203	16	37220.1	37253.7	37257.6
2	37206.1	37205.8	37206.1	17	37223	37270.1	37258.3
3	37207.7	37206.3	37207.6	18	37223.1	37270.2	37259
4	37207.9	37207.7	37207.8	19	37223.3	37272.5	37261
5	37208	37208.1	37214.7	20	37224.7	37275	37261.2
6	37208.2	37213.1	37214.9	21	37231.4	37277.7	37262.6
7	37208.7	37213.7	37215.4	22	37231.6	37280.1	37265.4
8	37214.2	37218.3	37215.7	23	37231.8	37284.9	37270.8
9	37215.2	37218.5	37216.3	24	37232.4	37289	37279.1
10	37215.9	37219.6	37216.9	25	37233.8	37291.9	37283.9
11	37216.2	37224.1	37223	26	37234.3	37292.6	37286.1
12	37216.7	37230.8	37223.7	27	37234.3	37296	37286.1
13	37219	37231.1	37223.9	28	37234.5	37296.3	37288.1
14	37219.2	37240.1	37228.3	29	37234.6	37296.9	37288.4
15	37219.5	37252.4	37257.2	30	37234.6	37298.8	37290.1

Table 5.4: Different runs on the tr6-15 instance with Manhattan approach

Table 5.4 presents some results on the computational issues discussed by testing the tr6-15 instance by the Manhattan approach. The column "Iter" indicates the iteration number of the framework, where at most 14 cuts will be added each time. The columns "lim=100" and "lim=150" indicate lower bounds with generating at most 100 and 150 extreme points for each 2-period problem, respectively. Finally, the column "RCC" indicates lower bounds obtained with the implementation of RC cuts, where *lim* is set to 150.

As Table 5.4 indicates, increasing *lim* from 100 to 150 increased the efficiency of the framework significantly, since number of cuts generated at every iteration is increased. On the other hand, one would expect RC cuts to improve the lower bounds even higher, although it does not seem to be the case. The main reason is the "luck" factor, since horizons are randomly selected, as previously explained. Also, our computational experience indicates to us that the RC cuts in general do not improve the lower bounds as efficiently as our separation cuts. However, they are useful as they make use of the reduced cost information instead of no action, and as one might recall from the previous section, they did help the separation framework to close duality gaps completely in couple of 2PCLS instances.

Next, we present the results using the Euclidean distance approach. Table 5.5 presents three different runs on the Trigeiro instance tr6-15 in a similar fashion to the previous table. The columns "lim=100" and "lim=150" indicate runs with setting the horizon to $t + 1$ for every 2-period problem, and with generating at most 100 and 150 extreme points for each 2-period problem respectively. The column "Horizon" indicates results of a run with assigning $\phi(i)$ values as discussed in Section 5.2 and with $lim = 150$. The reason we have different characteristics for runs here is to present different

Iter	lim=100	lim=150	Horizon	Iter	lim=100	lim=150	Horizon
1	37218.6	37218.6	37213	16	37294.9	37294.4	37286
2	37222.8	37222.8	37214.6	17	37299	37295.5	37288.7
3	37226	37226	37218.1	18	37299.6	37297.1	37297.1
4	37230.6	37230.6	37222.2	19	37300.4	37300.4	37300.1
5	37239.9	37239.9	37227.6	20	37300.5	37302.4	37303.7
6	37255.4	37255.3	37241.1	21	37301.8	37304.6	37310
7	37261.3	37260.8	37244.1	22	37302.7	37304.6	37310.8
8	37265.1	37262.9	37247.5	23	37302.9	37304.6	37311.9
9	37274.7	37267.1	37249.9	24	37304.6	37304.7	37323.6
10	37277.8	37268.5	37255.3	25	37304.6	37304.7	37324.4
11	37281.1	37270.7	37256.7	26	37305.7	37305.9	37324.4
12	37285.5	37273.9	37263.9	27	37306.6	37306.5	37325.9
13	37287.5	37277.8	37272	28	37306.7	37306.7	37328.5
14	37288.1	37288.9	37275.7	29	37306.7	37306.8	37328.7
15	37289.8	37290.9	37280.9	30	37306.7	37306.8	37331.2

Table 5.5: Different runs on the tr6-15 instance with Euclidean approach

aspects of computations and how they affect results.

There are some interesting conclusions that can be drawn from Table 5.5. First of all, note that increasing *lim* in case of 2-period problems with $\phi(i) = t + 1$ affects the lower bound generated only insignificantly. Also note that the lower bound improvement is becoming less significant with the number of iterations in case of $\phi(i) = t + 1$, as we did expect in our discussion of Section 5.2. On the other hand, assigning $\phi(i)$ values using the scheme described in Section 5.2 allows more significant lower bound improvements even in later iterations.

Another important aspect of these results is the comparison of Euclidean distance approach to Manhattan distance approach. Note that even in case of $\phi(i) = t + 1$, Euclidean approach improved the lower bounds much more efficiently than the Manhattan approach could before, as we did also observe in case of 2-period problems. Also, it is important to note that, even though we did not report computational times here,

the Euclidean approach is 2 to 3 times faster than the Manhattan approach (per cut generated), which makes us to conclude that it is the choice of approach.

5.4 Concluding Remarks

In this chapter, we have presented a new approach that can improve the traditional lower bounds for the production planning problems significantly by tackling the subproblems. One important advantage of the framework is its independence from defining families of valid inequalities or reformulations a priori, and to our knowledge, this is an original approach in that manner in the production planning literature. Although we do not define families of valid inequalities in the framework, this is expected to be a significant output of the framework that we can define some sets of valid inequalities a priori using our experience from this framework and hence obtain a better understanding of these problems. This was the major motivation for the detailed discussions in the computational results. This is left as future work.

Both Manhattan and Euclidean approaches have proven to be useful to generate cuts and improve lower bounds significantly, particularly for small problems of the test set 2PCLS. From the aspect of computational efficiency, the Euclidean approach is significantly better than the Manhattan approach. This has been observed both on the efficiency of cuts and on the number of extreme points generated in column generation before termination. There is only limited computational experience with big and realistic models, so that is a near future goal to test the framework on a few other test problems.

Computational efficiency of the framework will also be an important future work to accomplish. One improvement strategy is to have a better understanding of the extreme

points of the 2-period problems, and use this knowledge to predefine some commonly useful extreme points. In the current version of the framework, for the sake of the implementation, we only predefine the origin as the first extreme point before starting the column generation. Another improvement possibility is for later iterations of the framework. It is possible that the later iterations use the same or similar extreme points due to similar solutions obtained from the LPR, and hence the extreme points generated in an iteration can then be used directly in the next iterations. However, since we possibly define different horizons for multi-period problems, this strategy will be not applicable in these problems. Also, another strategy would be to remove the extreme points from the distance problem for simplicity if these extreme points have λ coefficients of 0 after some iterations.

Although this approach is using the special structure of the production planning problems that can look at the 2-period subproblem convex hulls separately, we believe that this concept can be extended for many other MIP problems and, therefore, provide a more general framework. An interesting ongoing study that can be linked to our work is the study of Cook et al. [2006] that studies generating local cuts instead of general cuts for MIP problems. This also suggests to us that the accomplishments we achieve may have a substantial impact on the MIP literature.

Chapter 6

Conclusions and Future Research

In this thesis, we have carefully investigated different aspects of big bucket production planning problems. First, we have presented an MIP based heuristic framework for generating high quality feasible solutions for practical problems. We have accomplished a detailed mathematical and computational analysis of these problems that has not only shown the relations between different lower bounds and relaxations but also indicated to us the main difficulty in the problem structure. Finally, we have studied this bottleneck and proposed a new methodology that aims to provide better lower bounds. We can use this framework even if we do not know the inequalities we need, and it can help us discover these inequalities and it can also show us how valuable they are.

In Chapter 3, we have proposed a heuristic framework that combines different mathematical programming techniques, namely relax-and-fix and LP-and-fix. The major advantages of the proposed framework are easy implementation, flexibility to solve a variety of production planning problems and the capability to generate multiple solutions allowing decision makers to have different options if needed. In addition, the heuristic provides lower bounds, which is not a typical characteristic of heuristics. Computational results have indicated that our heuristic can generate high quality solutions and competitive lower bounds, particularly for the most difficult problems, when compared with different benchmarks. Although the framework has provided good solutions, many test

problems have remained challenging with high duality gaps, providing motivation for the analysis of the next chapter.

In Chapter 4, we have presented different methodologies for obtaining lower bounds, which include both traditional and new techniques. Mathematical analysis with the goal of understanding the underlying difficulties of the production planning problems has been accomplished with a variety of theoretical results. Computational tests have provided more insight into these theoretical results, especially for the approximations of some methodologies. In particular, Lagrangian relaxations have indicated that relaxing capacity constraints results in significantly easier problems compared to the problems obtained by relaxing level-linking constraints. The main conclusion of the chapter is that the bottleneck of production planning problems lies in the current absence of an adequate polyhedral approximation of the convex hull of the single-level single-machine capacitated problems. This has provided the main motivation for the proposed methodology in the following chapter.

In Chapter 5, we have proposed a new approach to significantly improve traditional lower bounds. The proposed technique uses information obtained from subproblems of the original problem to generate valid inequalities. More specifically, an LPR solution is examined to determine whether it is in the convex hull of the 2-period subproblem or not, and in the latter case, a cutting plane is generated using a crucial theory derived from duality. This separation algorithm can be accomplished using both a linear approach (Manhattan distance) and a quadratic approach (Euclidean distance). We refer to the general framework as "2-period convex hull closure" since for a given multi-period problem, it generates all the valid inequalities for the convex hulls of the 2-period subproblems and hence defining their closures. Subproblems are defined due to LPR

solution's characteristics to achieve the most efficient cuts. The main advantage of the framework is that it does not require any predefined valid inequalities or reformulations as in the traditional MIP approach, although it should be noted that one of our main goals in this chapter was to define new facet-defining valid inequalities using the results of the framework and therefore to improve our understanding about these problems. We are not aware of any similar approach in the production planning literature, and we believe that this has significant potential to impact a broad area of MIP by applying it to substructures of other problems that increase problem complexity. It is also important to note that Euclidean distance provides a significantly more efficient approach than the Manhattan distance, which can be motivation for dealing with other MIP issues, as we will discuss next in future research.

To summarize, we have not only studied different aspects of big bucket production planning problems but also employed different methodologies ranging from heuristic methods to polyhedral analysis, in order to prevent myopic conclusions. We believe our contribution to these problems, which currently have limited results, can provide significant insights and motivations for other researchers in production planning and even in the general field of MIP.

Although theoretical and computational results presented in this thesis have proven to be useful for production planning problems, there are still a number of research goals to be accomplished in the future to address questions that still remain unresolved.

One important and immediate future goal is to identify new general families of facet defining inequalities for production planning problems based on the results obtained from the 2-period convex hull closure framework. Although we had a detailed discussion of some of the cuts generated by the framework in Chapter 5, we have not generalized

any of these inequalities yet. Our main focus will be on the results obtained from the 2-period problems, as these do not only provide an easier overview but also we do have actual facets for some of these instances through cdd. Once we can define any new family or families of facet-defining inequalities, investigating the possibility of polynomial separation algorithms will be the subsequent goal, although this is unlikely due to the fact that even the simpler single-period relaxation of Miller et al. [2000] cannot achieve a polynomial separation. Achieving these goals will give us not only a chance to test the computational strength of these inequalities but also provide a better understanding of these problems.

Although the results we presented in Section 5.3 are computationally promising, the 2-period convex hull closure framework is not thoroughly tested on realistic size production planning problems, such as TDS instances used in previous chapters. Our preliminary results indicate that inequalities defining 2-period convex hull closures close duality gaps of the production planning problems significantly, and the possibility of generalizing this conclusion will strengthen our results. A crucial step of extending these results might be to improve the computational efficiency of the framework, such as eliminating inactive extreme points to keep problem sizes as small as possible, and therefore, we will also focus on the computational efficiency of the framework while testing. Achieving more computational results is also an immediate future goal.

As general MIP techniques can be applied to production planning problems, some characteristics from production planning problems can be applied to MIP problems as well. This is particularly true if we look at similar problems, such as fixed charge network problems and knapsack problems. This is our motivation to investigate the possibility of extending the 2-period convex hull closure framework to other MIP problems.

This might require us to define interesting structured subproblems such as the 2-period problems, however, it is also possible to generalize it in a way that these subproblems are automatically selected. This will not only generate valid inequalities without using any problem specific inequalities but also improve our understanding about specific or general MIP problems. This study is left as future work.

Another future research direction will be to study the theoretical strength of the 2-period convex hull closures. It will be particularly interesting to compare it with the elementary closures mentioned before, assuming it is analytically possible. As of now we do not have sufficient knowledge to determine whether it is possible or not. However, if any results can be obtained, that will be significantly important for the MIP community.

Recall that we emphasized the advantage of using Euclidean distance rather than Manhattan distance in Chapter 5. This has significant potentials in the general MIP literature, as Manhattan distance has been the traditional approach. One particular area of interest is the "cut generation LP" procedure of lift-and-project cuts (Balas et al. [1993]). To be more specific, consider an MIP problem $\min\{cx|Ax \geq 0, x_i \in \{0,1\}\forall i = 1,...,n\}$. The convex hull of the union of disjunctions over the first binary variable can be defined as

$$
\begin{array}{llll}
Ax^1 \geq by^1 & (u) & Ax^2 \geq by^2 & (v) \\
x_1^1 = 0y^1 & (u_0) & x_1^2 = 1y^2 & (v_0) \\
 & x^1 + x^2 = x & (\gamma) & \\
 & y^1 + y^2 = 1 & (\gamma_0) & \\
 & x^1, x^2, y^1, y^2 \geq 0 & &
\end{array}
$$

Let x^* be the LP relaxation solution of this MIP problem and x_1^* be fractional. Using

the dual variables indicated in the parentheses above, we can generate a cutting plane for x^* by solving the following dual problem:

$$
\begin{aligned}
\max \quad & x^*\gamma + \gamma_0 \\
\text{st.} \quad & A^T u + u_0 e_1 + \gamma \leq 0 \\
& A^T v + v_0 e_1 + \gamma \leq 0 \\
& \gamma_0 \leq b^T u \\
& \gamma_0 \leq b^T v + v_0
\end{aligned}
$$

Since this problem is unbounded, researchers have used a normalization constraint in the form of $\sum_j u_j + \sum_j v_j + u_0 + v_0 = 1$ (see e.g. Balas et al. [1996]), which has the drawback of adding a slack variable to all the constraints in the primal problem. Alternatively, Manhattan distance was used in this area, where the distance of x^* to the convex hull of the union of disjunctions is minimized, and duality is used to generate the disjunctive cuts. We believe that using Euclidean distance has a potential to improve the procedure of generating disjunctive cuts by generating better quality cuts.

Finally, we also want to note that we plan to study whether the simple approach of the heuristic framework presented in Chapter 3 can be extended to other challenging MIP problems. We have used the special structure of production planning problems for this framework, however, similar structures exist in other MIP problems, where decisions can be ranked with regards to importance, such as facility location problems with facility decisions having different significance.

Bibliography

Quarterly financial report for U.S. manufacturing, mining, and wholesale trade corporations. *U.S. Census Bureau*, 2006. Available from `http://www.census.gov`.

P. Afentakis and B. Gavish. Optimal lot-sizing algorithms for complex product structures. *Operations Research*, 34(2):237–249, 1986.

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

A. Agra and M. Constantino. Lotsizing with backlogging and start-ups: The case of Wagner-Whitin costs. *Operations Research Letters*, 25(2):81–88, 1999.

R.K. Ahuja and D. Hochbaum. Solving linear cost dynamic lot sizing problems in $O(nlogn)$ time. To appear in *Operations Research*, 2006.

R.K. Ahuja, T.L. Magnanti, and J.B. Orlin. *Network Flows*. Prentice-Hall, 1993.

K. Andersen, G. Cornuéjols, and Y. Li. Split closure and intersection cuts. *Mathematical Programming*, 102(3):457–493, 2005.

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

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

K.R. Baker. *Requirements planning*. In Graves et al. [1993]. 1993.

E. Balas. Disjunctive programming. *Annals of Discrete Mathematics*, 5:351, 1979.

E. Balas and M. Perregaard. A precise correspondence between lift-and-project cuts, simple disjunctive cuts, and mixed integer Gomory cuts for 0-1 programming. *Mathematical Programming*, 94(2-3):221–245, 2003.

E. Balas and A. Saxena. Optimizing over the split closure. To appear in *Mathematical Programming*, 2007.

E. Balas, S. Ceria, and G. Cornuéjols. A lift-and-project cutting plane algorithm for mixed 0-1 programs. *Mathematical Programming*, 58(1-3):295–324, 1993.

E. Balas, S. Ceria, and G. Cornuéjols. Mixed 0-1 programming by lift-and-project in a branch-and-cut framework. *Management Science*, 42(9):1229–1246, 1996.

E. Balas, S. Ceria, M. Dawande, F. Margot, and G. Pataki. OCTANE: A new heuristic for pure 0-1 programs. *Operations Research*, 49(2):207–225, 2001.

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

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

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

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

D. Bienstock and M. Zuckerberg. Approximate fixed-rank closures of covering problems. *Mathematical Programming*, 105(1):9–27, 2006.

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

A. Caprara and A.N. Letchford. On the separation of split cuts and related inequalities. *Mathematical Programming*, 94(2-3):279294, 2003.

V. Chvátal. Edmonds polytopes and a hierarchy of combinatorial problems. *Discrete Mathematics*, 4(4):305–337, 1973.

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

M. Constantino. Lower bounds in lot-sizing models: A polyhedral study. *Mathematics of Operations Research*, 23(1):101–118, 1998.

W. Cook, R. Kannan, and A. Schrijver. Chvátal closures for mixed integer programming problems. *Mathematical Programming*, 102(1-3):155–174, 1990.

W. Cook, D. Espinoza, and V. Chvátal. Local cuts for mixed integer programming. Working Paper, *Presented at ISMP Meeting, Rio De Janeiro, Brazil*, 2006.

G. Cornuéjols and Y. Li. Elementary closures for integer programs. *Operations Research Letters*, 28(1):1–8, 2001.

E. Danna, E. Rothberg, and C. Le Pape. Exploring relaxation induced neighborhoods to improve MIP solutions. *Mathematical Programming*, 102(1):71–90, 2005.

S. Dash and O. Günlük. Valid inequalities based on simple mixed-integer sets. *Mathematical Programming*, 105(1):29–53, 2006.

T.H. Davenport. Putting the enterprise into the enterprise system. *Harvard Business Review*, 76(4):121–131, 1998.

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

D. Erlenkotter. Ford Whitman Harris and the economic order quantity model. *Operations Research*, 38(6):937–946, 1990.

A. Federgruen and J. Meissner. Probabilistic analysis of multi-item capacitated lot sizing problem. Working paper, under revision, 2004.

A. Federgruen and M. Tzur. A simple forward algorithm to solve general dynamic lot sizing models with n periods in O($nlogn$) or O(n) time. *Management Science*, 37(8):909–925, 1991.

A. Federgruen and M. Tzur. Time-partitioning heuristics: Application to one-warehouse, multiitem, multiretailer lot-sizing problems. *Naval Research Logistics*, 46(5):463–486, 1999.

A. Federgruen, J. Meissner, and M. Tzur. Progressive interval heuristics for the multi-item capacitated lot sizing problem. Under revision for publication in *Operations Research* , 2002.

M. Fischetti and A. Lodi. Local branching. *Mathematical Programming*, 98(1-3): 23–47, 2003.

M. Fischetti and A. Lodi. Optimizing over the first Chvátal closure. Integer Programming and Combinatorial Optimization-IPCO 2005, LNCS 3509, pages 12–22. Springer, 2005.

M. Fischetti, F. Glover, and A. Lodi. The feasibility pump. *Mathematical Programming*, 104(1):91–104, 2005.

M. Fisher, K. Ramdas, and Y.S. Zheng. Ending inventory valuation in multiperiod production scheduling. *Management Science*, 47(5):679–692, 2001.

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

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

K. Fukuda. cdd and cddplus homepage. `http://www.ifor.math.ethz.ch/~fukuda/cdd_home/`, 2005.

R.E. Gomory. An algorithm for mixed integer programming. RM-2597, The RAND Corporation, 1960.

M. Gopalakrishnan, K. Ding, J.M. Bourjolly, and S. Mohan. A tabu-search heuristic for the capacitated lot-sizing problem with set-up carryover. *Management Science*, 47(6):851–863, 2001.

S.C. Graves, A.H.G. Rinnooy Kan, and P.H. Zipkin, editors. *Logistics of Production and Inventory*, volume 4 of *Handbooks in Operations Research and Management Science*. North-Holland, 1993.

F.W. Harris. How many parts to make at once. *Factory: The Magazine of Management*, 10(2):135–136,152, 1913. (Also reprinted in *Operations Research*, 38(6): 947-950, 1990).

W. J. Hopp and M. L. Spearman. *Factory Physics*. McGraw-Hill, 2000.

V.N. Hsu. Dynamic economic lot size model with perishable inventory. *Management Science*, 46(8):1159–1169, 2000.

R. Jans and Z. Degraeve. Improved lower bounds for the capacitated lot sizing problem with setup times. *Operations Research Letters*, 32:185–195, 2004.

E. Katok, H.S. Lewis, and T.P. Harrison. Lot sizing in general assembly systems with setup costs, setup times, and multiple constrained resources. *Management Science*, 44(6):859–877, 1998.

J. Krarup and O. Bilde. *Plant location, set covering and economic lotsizes: An O(mn) algorithm for structured problems*, pages 155–180. Optimierung bei Graphentheoretischen und Ganzzahligen Probleme. Birkhauser Verlag, 1977.

S. Küçükyavuz and Y. Pochet. Uncapacitated lot-sizing with backlogging: The convex hull. Working Paper, *Presented at ISMP Meeting, Rio de Janeiro, Brazil*, 2006.

C.Y. Lee, S. Çetinkaya, and A.P.M. Wagelmans. A dynamic lot-sizing model with demand time windows. *Management Science*, 47(10):1384–1395, 2001.

H.L. Lee and S. Nahmias. *Single-Product, Single-Location Models*. In Graves et al. [1993]. 1993.

A.N. Letchford. On disjunctive cuts for combinatorial optimization. *Journal of Combinatorial Optimization*, 5:299–315, 2001.

J.M.Y. Leung, T.L. Magnanti, and R. Vachani. Facets and algorithms for capacitated lot sizing. *Mathematical Programming*, 45(1-3):331–359, 1989.

M. Loparic, Y. Pochet, and L.A. Wolsey. The uncapacitated lot-sizing problem with sales and safety stocks. *Mathematical Programming*, 89(3):487–504, 2001.

LOTSIZELIB. Lot-sizing problems: A library of models and matrices. `http://www.core.ucl.ac.be/wolsey/lotsizel.htm`, 1999.

J. Maes and L. van Wassenhove. Multi item single level capacitated dynamic lotsizing heuristics: A computational comparison (part i: Static case; part ii: Rolling horizon). *IIE Transactions*, 18:114–129, 1986.

O. Mangasarian. *Nonlinear Programming*. SIAM Classics in Applied Mathematics, 1994.

H. Marchand and L.A. Wolsey. Aggregation and mixed integer rounding to solve mips. *Operations Research*, 49(3):363–371, 2001.

H. Marchand and L.A. Wolsey. The 0-1 knapsack problem with a single continuous variable. *Mathematical Programming*, 85(1):15–33, 1999.

H. Marchand, A. Martin, R. Weismantel, and L. Wolsey. Cutting planes in integer and mixed integer programming. CORE Discussion Paper 1999/53, *CORE, UCL, Belgium*, 1999.

A.J. Miller and L.A. Wolsey. Tight MIP formulations for multi-item discrete lot-sizing problems. *Operations Research*, 51(4):557–565, 2003.

A.J. Miller, G.L. Nemhauser, and M.W.P. Savelsbergh. Solving the multi-item capacitated lot-sizing problem with setup times by branch-and-cut. CORE Discussion Paper 2000/39, *CORE, UCL, Belgium*, 2000.

A.J. Miller, G.L. Nemhauser, and M.W.P. Savelsbergh. On the polyhedral structure of a multi-item production planning model with setup times. *Mathematical Programming*, 94(2-3):375–405, 2003.

MIPLIB. A library of pure and mixed integer problems. `http://miplib.zib.de/`, 2003.

T.S. Motzkin, H. Raiffa, G. Thompson, and R.M. Thrall. The double description method. volume 2 of *Contributions to the Theory of Games*, pages 51–73. Princeton University Press, 1953.

Multi-LSB. Multi-item lot-sizing problems with backlogging: A library of test instances. `http://homepages.cae.wisc.edu/~kerema/research/multi-lsb`, 2006.

S. Nahmias. *Production and Operations Analysis*. McGraw-Hill, 2001.

G.L. Nemhauser and L.A. Wolsey. A recursive procedure to generate all cuts for 0-1 mixed integer programs. *Mathematical Programming*, 46(1-3):379–390, 1990.

G.L. Nemhauser and L.A. Wolsey. *Integer and Combinatorial Optimization*. Wiley-Interscience, 1999.

T. Ohno. *Toyota Production System: Beyond Large-Scale Production*. Productivity Press, 1988. (Translated from his original 1978 publication).

J. Orlicky. *Material Requirements Planning*. McGraw-Hill, 1975.

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

J-S. Pang. An equivalence between two algorithms for quadratic programming. *Mathematical Programming*, 20(1):152–165, 1981.

N.C. Petruzzi and M. Dada. Pricing and the newsvendor problem: A review with extensions. *Operations Research*, 47(2):183–194, 1999.

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

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

Y. Pochet and L.A. Wolsey. Lot-size models with backlogging: Strong reformulations and cutting planes. *Mathematical Programming*, 40(1-3):317–335, 1988.

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

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

Y. Pochet and L.A. Wolsey. Polyhedra for lot-sizing with Wagner-Whitin costs. *Mathematical Programming*, 67(1-3):297–323, 1994.

R.L. Rardin and L.A. Wolsey. Valid inequalities and projecting the multicommodity extended formulation for uncapacitated fixed charge network flow problems. *European Journal of Operational Research*, 71(1):95–109, 1993.

J.-P.P. Richard, I.R. de Farias, and G.L. Nemhauser. Lifted inequalities for 0-1 mixed integer programming: Basic theory and algorithms. *Mathematical Programming*, 98(1-3):89–113, 2003a.

J.-P.P. Richard, I.R. de Farias, and G.L. Nemhauser. Lifted inequalities for 0-1 mixed integer programming: Superlinear lifting. *Mathematical Programming*, 98 (1-3):115–143, 2003b.

S.M. Robinson. Personal communication. 2007.

R.S. Russell and B.W. Taylor III. *Production and Operations Management*. Prentice Hall, Inc., 1995.

A. Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*. Springer, 2003.

A. Schrijver. *Theory of Linear and Integer Programming*. John Wiley and Sons, 1986.

E.A. Silver and H. Meal. A heuristic for selecting lotsize quantities for the case of a deterministic time-varying demand rate and discrete opportunities for replenishment. *Production Inventory Management*, 14:64–74, 1973.

N.C. Simpson and S.S. Erenguc. Modeling multiple stage manufacturing systems with generalized costs and capacity issues. *Naval Research Logistics*, 52(6):560–570, 2005.

H. Stadtler. Improved rolling schedules for the dynamic single-level lot-sizing problem. *Management Science*, 46(2):318–326, 2000.

H. Stadtler. Multilevel lot sizing with setup times and multiple constrained resources: Internally rolling schedules with lot-sizing windows. *Operations Research*, 51(3):487–502, 2003.

C. Suerie and H. Stadtler. The capacitated lot-sizing problem with linked lot sizes. *Management Science*, 49(8):1039–1054, 2003.

R. Suri. *Quick Response Manufacturing : A Companywide Approach to Reducing Lead Times*. Productivity Press, 1998.

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

R.J. Tersine. *Principles of Inventory and Materials Management*. North-Holland, 1982.

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

W. van den Heuvel and A.P.M. Wagelmans. A comparison of methods for lot sizing in a rolling horizon environment. *Operations Research Letters*, 33(5):486–496, 2005.

C.P.M. van Hoesel and A.P.M. Wagelmans. Fully polynomial approximation schemes for single-item capacitated lot-sizing problems. *Mathematics of Operations Research*, 42(2):339–357, 2001.

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

S. van Hoesel, H.E. Romeijn, D.R. Morales, and A.P.M. Wagelmans. Integrated lot sizing in serial supply chains with production capacities. *Management Science*, 51(11):1706–1719, 2005.

T.J. Van Roy and L.A. Wolsey. Solving mixed integer programming problems using automatic reformulation. *Operations Research*, 35(1):45–57, 1987.

M. Van Vyve. Personal communication. 2003.

M. Van Vyve and F. Ortega. Lot-sizing with fixed charges on stocks: the convex hull. *Discrete Optimization*, 1(2):189–203, 2004.

M. Van Vyve and Y. Pochet. A general heuristic for production planning problems. *INFORMS Journal of Computing*, 16(3):316–327, 2004.

M. Van Vyve and L. Wolsey. Approximate extended formulations. *Mathematical Programming*, 105(3):501–522, 2005.

F. Vanderbeck. Lot-sizing with start-up times. *Management Science*, 44(10): 1409–1425, 1998.

N. Viswanadham and Y. Narahari. *Performance Modeling of Automated Manufacturing Systems*. Prentice Hall, 1992.

T.E. Vollmann, W.L. Berry, and D.C. Whybark. *Manufacturing Planning and Control Systems*. Irwin, 1992.

A.P.M. Wagelmans, S. van Hoesel, and A. Kolen. Economic lot sizing: An $O(nlogn)$ algorithm that runs in linear time in the Wagner-Whitin case. *Operations Research*, 40(1):S145–S156, 1992.

H.M. Wagner and T.M. Whitin. Dynamic version of the economic lot size model. *Management Science*, 5(1):89–96, 1958.

O. Wight. *Manufacturing Resource Planning: MRP-II*. Oliver Wight Ltd., 1984.

L.A. Wolsey. Solving multi-item lot-sizing problems with an MIP solver using classification and reformulation. *Management Science*, 48(12):1587–1602, 2002.

L.A. Wolsey. Lot-sizing with production and delivery time windows. *Mathematical Programming*, 107(3):471–489, 2007.

L.A. Wolsey. *Integer Programming*. Wiley-Interscience, 1998.

W.I. Zangwill. A backlogging model and a multi-echelon model of a dynamic economic lot size production system-a network approach. *Management Science*, 15(9):506–527, 1969.

Appendix A

Sample Heuristic File in Mosel

```
model "Heuristic_TDS"
uses "mmxprs" !Load the libraries
declarations
NI=10 !number of items
NT=24 !number of periods
NK=3  !number of machines
nbendp=1 !number of enditems
item= 1..NI
enditem=1..nbendp
period= 1..NT
mc= 1..NK
bSolutionFound : boolean
cost: mpvar
x: array(period,item) of mpvar     !production amount
y: array(period,item) of mpvar     !setup variable
s: array(period,item) of mpvar     !stock variable
ot: array(period,mc) of mpvar      !overtime variable
echs: array (period,item) of mpvar          !echelon stock
echd: array (period,item) of real           !echelon demand
f: array(item) of real             !setup cost for each item
h: array(item) of real             !holding cost for each item
oc: array(mc) of real              !overtime cost for each mc
```

```
demand: array(period,item) of real        !demand
totdem: array(period,item) of real        !total demand for t,..,NT
avgdem: array(item) of real               !mean primary demand
netdem: array(item) of real               !mean net demand
r: array(item,item) of real           !number req to produce successor
ech: array(item,enditem) of real      !number required to produce enditem
a: array(mc,item) of real             !prod. coefficient for mc/item pair
ST: array(mc,item) of real            !setup time for mc/item pairs
RU: array(mc) of real                 !resource utilization for each mc
tbo: array(item) of real              !TBO for each product
e: array(item) of real                !marginal holding costs
cap: array(mc) of real                !available capacity
int1: integer             !int. values used for data reading purposes
int2: integer
maxiter = 50              !Max. number of iterations-(l,S) inequalities
iter = 1..maxiter
setS: array(iter,period,period,item) of integer ! Set S as defined
dem: array(period,period,item) of real       ! Total demand (t,l)
countviol: integer        !counter for # of violations at each iteration
length = 3                !Length of the window (Relax & Fix)
overlap = 1               !# of overlapping periods (Relax & Fix)
minextratime = 5.00       !LP-and-Fix parameter (min time)
maxextra = 50.00          !LP-and-Fix parameter (max time)
total_allowed_time = 180.00    !Total time (different for diff.prob.)
end-declarations
setparam("XPRS_CPUTIME",1)     !Use CPU time instead of actual time
```

```
! ==========================
! INPUT DATA FROM FILES ===
! ==========================
!Read the demand amounts:
fopen("p-bedarf.prn",F_INPUT)
forall (i in item) do
   forall (t in 1..(NT-1)) read(demand(t,i),',')
   readln(demand(NT,i))
end-do
fclose(F_INPUT)
!Read the mean primary demand:
fopen("mitt_bed.prn",F_INPUT)
forall (i in item) read(avgdem(i),',')
fclose(F_INPUT)
!First assign all r values 0. Then, Read the values from the file:
forall (i in item,j in item) r(i,j):=0
fopen("direkt-b.prn",F_INPUT)
forall (k in 1..100) readln(int1,',',int2,',',r(int1,int2))
fclose(F_INPUT)
!Calculate the echelon values for each item/enditem pair:
forall(i in item,j in enditem) ech(i,j):=0
!Since with increasing index number, the level becomes lower...
forall (i in (nbendp+1)..NI) do
   forall (j in enditem) ech(i,j):= r(i,j)
   forall (j in enditem,k in (nbendp+1)..i)
      ech(i,j):= ech(i,j) + r(i,k)*ech(k,j)
end-do
!Production coefficients: Initialize 0. Read the file.
```

```
forall(i in mc,j in item) a(i,j):=0
fopen("prodkoef.prn",F_INPUT)
forall(k in 1..NI) readln(int1,',',int2,',',a(int1,int2))
fclose(F_INPUT)
!Setup times: Initialize 0. Read the file.
forall(i in mc,j in item) ST(i,j):=0;
fopen("ruestz.prn",F_INPUT)
forall(k in 1..NI) readln(int1,',',int2,',',ST(int1,int2))
fclose(F_INPUT)
!Read the resource utilization factors from the data file
fopen("auslast.prn",F_INPUT)
forall(i in mc) read(RU(i),',')
fclose(F_INPUT)
!Read the TBO values from the data file
fopen("tbo.prn",F_INPUT)
forall(i in item) readln(tbo(i))
fclose(F_INPUT)
!Read the overtime cost coefficients for each mc:
fopen("ueber-ks.prn",F_INPUT)
forall(i in mc) readln(oc(i))
fclose(F_INPUT)
! Read the marginal holding cost coefficients for each mc:
fopen("zfkoef.prn",F_INPUT)
forall(i in item) read(e(i),',')
fclose(F_INPUT)
! ==================================
! CALCULATION OF MODEL PARAMETERS ===
! ==================================
```

```
!Calculate mean net demand:
netdem(1):= avgdem(1)
forall (i in 2..NI)
   netdem(i):= avgdem(i)+ sum(j in 1..(i-1)) r(i,j)*netdem(j)
!Calculate available capacity on each machine:
forall(i in mc)
   cap(i):= (sum(j in item)(a(i,j)*netdem(j) + ST(i,j)))/RU(i)
!Setup costs:
forall(i in item)
   f(i):= 0.5*e(i)*netdem(i)*tbo(i)^2
!Holding costs:
h(NI):= e(NI)
int1:= NI
repeat
   int1:= int1-1
   h(int1):= e(int1) + sum(j in (int1+1)..NI) r(j,int1)*h(j)
until (int1<=1)
!Calculate total demand:
forall(t in period,i in item)
   totdem(t,i):= sum(t2 in t..NT) demand(t2,i)
!=============================================
! MODEL: CONSTRAINTS&OBJECTIVE FUNCTION ====
!=============================================
!Objective function:
cost= sum(t in period,j in item) (f(j)*y(t,j))
+ sum(t1 in period,j in item) (h(j)*s(t1,j))
+ sum(t in period,k in mc) (oc(k)*ot(t,k))
```

```
!Flow balance for enditems:
forall (i in enditem) bal1(i):= x(1,i)-s(1,i)=demand(1,i)
forall (t in 2..NT,i in enditem)
bal2(t,i):= x(t,i) + s(t-1,i)-s(t,i)=demand(t,i)
!Flow balance for non-enditems:
forall (i in (nbendp+1..NI))
bal3(i):= x(1,i)-s(1,i)=sum(j in 1..(i-1)) r(i,j)*x(1,j)
forall (t in 2..NT,i in (nbendp+1..NI))
bal4(t,i):= x(t,i)+s(t-1,i)-s(t,i)=sum(j in 1..i-1) r(i,j)*x(t,j)
!Capacity constraint:
forall (t in period,k in mc)
mcc(t,k):= sum(j in item)(a(k,j)*x(t,j)+ST(k,j)*y(t,j))<=cap(k)+ot(t,k)
!Production variable-binary variable relations:
forall (t in period, i in enditem)
rel1(t,i):= x(t,i)<=totdem(t,i)*y(t,i)
forall (t in period, i in (nbendp+1..NI))
rel2(t,i):= x(t,i) <= y(t,i)*(sum(j in enditem) ech(i,j)*totdem(t,j))
forall (t in period, i in item, k in mc) do
   if (a(k,i)>0) then
      rel3(t,i,k):= a(k,i)*x(t,i)<=(cap(k)-ST(k,i))*y(t,i)+ot(t,k)
   end-if
end-do
! ==================================================
! ADD (l,S) INEQUALITIES TO BASIC FORMULATION ====
! ==================================================
!Define echelon demand:
forall (t in period, i in item) do
   if (i in enditem) then
```

```
        echd(t,i):= demand(t,i)
    else
        echd(t,i):= sum(j in enditem) ech(i,j)*demand(t,j)
    end-if
end-do
!Define echelon stock:
forall (t in period, i in item) do
    echs(t,i) = s(t,i)+sum(j in 1..(i-1)) r(i,j)*echs(t,j)
end-do
!Relax the setup variables:
forall (t in period, i in item) necessary(t,i):= y(t,i)<=1
!Prevent the Gomory cuts for lot-sizing problem:
setparam("XPRS_GOMCUTS",0)
!Keep track of time:
starttime := gettime
! First, calculate the demands dem(t,l) for each period t,l:
forall (i in item, l in 1..NT) do       ! for each l and item i
    forall (t in 1..l) dem(t,l,i):= 0
    dem(l,l,i) := echd(l,i)
    if (l >= 2) then
        forall (t in 1..(l-1))
            dem(l-t,l,i):= dem(l-t+1,l,i)+echd(l-t,i)
    end-if
end-do
!First, solve the LP relaxation:
minimize(cost)
writeln('The LP relaxation solution...')
writeln('The total cost:', getobjval)
```

```
ls_soln:= getobjval
totalviol:= 0        !Counter for total number of inequalities added
forall(it in iter, i in item,l in 1..NT) deleted_ls_index(it,i,l):= 0
total_deleted:= 0   !Counter for number of inequalities deleted
forall(iteration in iter) do   !(l,S) separation starts here
! initialize the counter for each iteration:
countviol := 0
forall (i in item) do
   forall (l in 1..NT) do
      ! Initialize the set S:
      forall (t in 1..l) setS(iteration,t,l,i) := 0
      ! Check whether x(t,i) > d(t,l,i)*y(t,i) or not:
      forall (t in 1..l) do
         if (getsol(x(t,i))>dem(t,l,i)*getsol(y(t,i))) then
            setS(iteration,t,l,i):= 1
         end-if
      end-do
      ! Add the maximum violated (L,S) inequality, IF there is any:
      if ((sum(t in 1..l) setS(iteration,t,l,i)*getsol(x(t,i))) >
      getsol(echs(l,i))+(sum(t in 1..l) setS(iteration,t,l,i)*
      dem(t,l,i)*getsol(y(t,i))) + 0.000001) then
         addcons(iteration,i,l):= (sum(t in 1..l) setS(iteration,t,l,i)
         *x(t,i))<=echs(l,i)+sum(t in 1..l) setS(iteration,t,l,i)*
         dem(t,l,i)*y(t,i)
         countviol := countviol+1
      end-if
   end-do
end-do
```

```
if (countviol = 0) then break
else
!Solve the strengthened LP relaxation:
minimize(cost)
if (getobjval-ls_soln < 0.001) then     !Just a backup to stop
   break
end-if
ls_soln:= getobjval
!Delete inactive l,s cuts:
deleted_ls:= 0
forall(it in 1..iteration, i in item,l in 1..NT) do
   if ((-0.1>getsol(addcons(it,i,l)) or getsol(addcons(it,i,l))>0.1)
   and deleted_ls_index(it,i,l)<>1) then
      deleted_ls_index(it,i,l):= 1
      sethidden(addcons(it,i,l),true)
      deleted_ls:= deleted_ls + 1
   end-if
end-do
total_deleted:= total_deleted + deleted_ls
writeln('LPR solution with (l,S) inequalities, iter=',iteration)
writeln('Number inequalities added:',countviol)
writeln('Total cost:',getobjval)
totalviol:= totalviol + countviol
num_iter:= iteration
end-if
end-do
!Print out the general statistics for (l,s) inequalities:
writeln("(l,S) inequalities stats:")
```

```
writeln("# valid inequalities = ",totalviol)
writeln('Total number deleted:',total_deleted)
ls_time:= gettime-starttime
writeln("Time spent = ",ls_time)
writeln("Number of iterations = ",num_iter)
! ======================
! INITIAL LP-AND-FIX ===
! ======================
!Default values:
extra_on_first:= 0.00
time_on_first:= 0.00
best_late_soln:= 1.0e+40
!Calculate number of windows:
winfirst:= length-overlap
numwin:= ceil((NT-overlap)/winfirst)
!Assign time:
time_for_first:= 0.75*(total_allowed_time - ls_time)/numwin
!First, get the LPR solution:
declarations
initial_soln: array(period,item) of integer
end-declarations
minimize(cost)
!Initialize solution for LP-and-Fix (round IF too close to 1/0):
forall (t in period, i in item)
   if (getsol(y(t,i)) < 0.00000001) then
      initial_soln(t,i):= 0
      first_binaries(t,i):= y(t,i) = initial_soln(t,i)
   elif (getsol(y(t,i)) > 0.99999999) then
```

```
        initial_soln(t,i):= 1
        first_binaries(t,i):= y(t,i) = initial_soln(t,i)
    else
        first_binaries(t,i):= y(t,i) is_binary
    end-if
setparam("XPRS_MAXTIME",time_for_first)
starttime:= gettime
minimize(cost)
time_on_first:= gettime - starttime
firstSolutionFound := getparam("XPRS_MIPSTATUS")=XPRS_MIP_SOLUTION or
getparam("XPRS_MIPSTATUS")=XPRS_MIP_OPTIMAL
    if(not firstSolutionFound)then
        writeln("WARNING: Solution not found at the beginning...")
    else
        best_late_soln:= getobjval
        writeln("Cutoff value = ",best_late_soln)
    end-if
!Delete all the fixings of binaries:
forall (t in period, i in item) sethidden(first_binaries(t,i),true)
! ==================================================================
! RELAX & FIX (LP-and-Fix in the loop, after each window) ====
! ==================================================================
!Allocate time to windows:
declarations
total_allowed_forwin: array (1..numwin) of real
quart: integer
half: integer
threeq: integer
```

```
end-declarations
quart:= floor(0.25*numwin)
half:= floor(0.5*numwin)
threeq:= floor(0.75*numwin)
!Average time left for each window is:
ave_left:= (total_allowed_time - ls_time - time_on_first)/numwin
forall (i in 1..quart) total_allowed_forwin(i):= 1.75*ave_left
forall (i in quart+1..half) total_allowed_forwin(i):= 1.25*ave_left
forall (i in half+1..threeq) total_allowed_forwin(i):= 0.75*ave_left
forall (i in threeq+1..numwin) total_allowed_forwin(i):= 0.25*ave_left
totaltime:= time_on_first+ls_time  !Keep track of total time
extratime:= 0.00        !Keep track of extra time in a window
totalextra:= 0.00       !Keep track of total extra time
late_total:= 0.00       !Keep track of total time used in Lp-and-fix
!Keep track of the fixed y variables
!Default value=2, meaning they're not fixed yet:
forall (t in period, i in item) notfixed(t,i):= 2
!Main loop for relax-and-fix starts here:
forall (win in 1..(numwin-1)) do
   forall (t in (win-1)*winfirst+1..(win-1)*winfirst+length,i in item)
      binaryy(win,t,i):= y(t,i) is_binary
!Prefered duality gap and cutoff value:
setparam("XPRS_MIPRELSTOP",0.005)
setparam("XPRS_MIPABSCUTOFF",best_late_soln)
!Set notfixed parameter for later periods to its default value:
forall (t in (win-1)*winfirst+1..NT,i in item) do
   notfixed(t,i):=2
end-do
```

```
! The following helps to re-fix the 0/1's that have been already fixed:
if (win >= 2) then
   forall (t in 1..(win-1)*winfirst,i in item) do
       if (notfixed(t,i)=1) then
           fixy(win,t,i):= y(t,i)= 1
       elif (notfixed(t,i)=0) then
           fixy(win,t,i):= y(t,i)= 0
       end-if
   end-do
end-if
!Set max time for each window:
setparam("XPRS_MAXTIME",-total_allowed_forwin(win))
starttime := gettime
!Solving the window...
minimize(cost)
!Calculate the extratime:
timeforwin:= gettime-starttime
totaltime := totaltime + timeforwin
starttime:= gettime
if (timeforwin < total_allowed_forwin(win)) then
   extratime:= total_allowed_forwin(win)-timeforwin
   totalextra:= totalextra + extratime
else
   extratime:= 0.00
end-if
if (win=1) then
   !The first window also provides a lower bound...
   lower_bound:= getparam("XPRS_bestbound")
```

```
end-if
bSolutionFound:= getparam("XPRS_MIPSTATUS")=XPRS_MIP_SOLUTION or
getparam("XPRS_MIPSTATUS")=XPRS_MIP_OPTIMAL
if(not bSolutionFound)then
   writeln("WARNING: Solution not found for time window : ", win)
   no_soln:= true
   break
else
   writeln('The solution obtained by time window ',win,':')
   writeln('Total cost = ',getobjval)
   writeln('Total time = ',totaltime)
   if (win=1) then
      writeln('Lower bound=',lower_bound)
   end-if
   !Fixing the variables from the window (both 0's and 1's):
   forall (t in (win-1)*winfirst+1..win*winfirst,i in item)
      if (getsol(y(t,i))=1) then
         fixy(win,t,i):= y(t,i)= 1
         notfixed(t,i):= 1
      elif (getsol(y(t,i))=0) then
         fixy(win,t,i):= y(t,i)= 0
         notfixed(t,i):= 0
      end-if
   forall (t in win*winfirst+1..NT,i in item)
      if (getsol(y(t,i))=1) then
         notfixed(t,i):= 1
      end-if
   forall (t in win*winfirst+1..(win-1)*winfirst+length,i in item)
```

```
        sethidden(binaryy(win,t,i),true)
    totaltime := totaltime + gettime - starttime
    starttime:= gettime
! ==================================================================
! LP-AND-FIX AFTER EACH WINDOW (if available time is enough) ===
! ==================================================================
if (win>=2 and (totalextra > minextratime)) then
    setparam("XPRS_MIPRELSTOP",0.0005)
    forall (t in win*winfirst+1..NT,i in item)
        extrawin(win,t,i):= y(t,i) is_binary
    !Also fix the exact binary solutions for later windows here:
    forall (t in win*winfirst+1..NT,i in item)
    if (notfixed(t,i)=1) then
        extrafixy(win,t,i):= y(t,i)= 1
    end-if
    !Set the max time as the extra time:
    if (totalextra>maxextra) then
        setparam("XPRS_MAXTIME",-maxextra)
    else
        setparam("XPRS_MAXTIME",-totalextra)
    end-if
    setparam("XPRS_MIPABSCUTOFF",best_late_soln)
    minimize(cost)
    timeforwin:= gettime-starttime
    late_total:= late_total + timeforwin
    totaltime := totaltime + timeforwin
    totalextra:= totalextra - timeforwin
    starttime:= gettime
```

```
    bSolnFound := getparam("XPRS_MIPSTATUS") = XPRS_MIP_SOLUTION or
    getparam("XPRS_MIPSTATUS") = XPRS_MIP_OPTIMAL
    if (bSolnFound) then
    writeln('The solution obtained by LP-and-Fix window',win,':')
    writeln('Total cost  = ',getobjval)
    writeln('Total time  = ',totaltime)
    if (lower_bound <= getobjval and getobjval < best_late_soln) then
       best_late_soln:= getobjval
    end-if
    else
       writeln("WARNING! No solution found on Lp-and-Fix, window ",win)
    end-if
    !Now, we are done with Lp-and-Fix, let's relax all the
    !variables back to continuous for further windows:
    forall (t in win*winfirst+1..NT,i in item) do
       sethidden(extrawin(win,t,i),true)
       sethidden(extrafixy(win,t,i),true)
    end-do
end-if
!Let's forget about the already fixed var.s in the original problem:
forall (t in 1..win*winfirst,i in item) sethidden(fixy(win,t,i),true)
!Reset the y's to be binary again:
forall (t in win*winfirst+1..(win-1)*winfirst+length,i in item)
    sethidden(binaryy(win,t,i),false)
end-if
end-do !End of the main loop for relax-and-fix
if (no_soln) then
    writeln("The best solution obtained is by LP-and-Fix!")
```

```
else
    setparam("XPRS_MIPABSCUTOFF",best_late_soln)
    forall (t in (numwin-1)*winfirst+1..NT,i in item)  y(t,i) is_binary
    !Set the fixed binaries:
    forall (t in 1..(numwin-1)*winfirst,i in item)
    if (notfixed(t,i)=1) then
        fixy_final(t,i):= y(t,i)= 1
    elif (notfixed(t,i)=0) then
        fixy_final(t,i):= y(t,i)= 0
    end-if
    setparam("XPRS_MAXTIME",-total_allowed_forwin(numwin))
    minimize(cost)
    bSolutionFound := getparam("XPRS_MIPSTATUS") = XPRS_MIP_SOLUTION or
    getparam("XPRS_MIPSTATUS") = XPRS_MIP_OPTIMAL
    if(not bSolutionFound)then
        no_soln:= true
    else
        best_late_soln:= getobjval
    end-if
end-if
writeln('Final stats...')
writeln('The total cost:', best_late_soln)
writeln('Lower bound=',lower_bound)
totaltime := totaltime + gettime - starttime
writeln("Total time in sec.= ", totaltime)
writeln("Total time in late= ", late_total)
end-model
```

Appendix B

Detailed Results of the Heuristic

B.1 TDS Instances

Instance	SH UB	LB from ℓ, S	Xpress UB	Xpress LB	Heuristic UB	Heuristic LB
AG501130	153,418	116,183	157,433	133,715	154,515	119,146
AG501131	146,500	107,829	157,450	116,004	145,225	109,714
AG501132	156,822	118,677	160,678	127,827	145,191	121,740
AG501141	175,619	133,424	186,362	140,206	171,895	134,421
AG501142	195,110	145,508	197,582	154,895	192,582	148,911
AG502130	168,707	122,353	170,059	146,383	167,927	128,101
AG502131	145,322	109,085	157,623	115,833	146,361	111,001
AG502141	179,371	134,971	188,933	141,121	173,640	136,353
AG502232	124,015	97,032	123,304	103,715	121,108	97,632
AG502531	129,080	102,340	140,294	108,749	129,640	103,506
AK501131	128,095	96,968	132,020	102,978	123,366	99,020
AK501132	124,499	101,699	126,919	106,109	123,473	103,077
AK501141	180,695	134,805	186,662	139,754	170,897	136,428
AK501142	174,701	134,880	176,504	138,587	161,262	135,875
AK501432	110,851	92,533	114,088	99,194	109,249	93,546
AK502130	129,307	102,222	133,429	110,583	127,889	103,949
AK502131	126,394	93,369	122,273	98,627	115,819	94,969
AK502132	119,175	96,312	121,712	100,679	118,319	97,233
AK502142	146,616	127,792	164,357	131,234	147,729	129,034
AK502432	110,094	88,980	110,896	94,459	105,415	89,609

Instance	SH UB	LB from ℓ, S	Xpress UB	Xpress LB	Heuristic UB	Heuristic LB
BG511132	140,155	108,772	166,747	115,116	137,637	110,466
BG511142	159,769	133,158	183,318	138,130	167,262	133,880
BG512131	139,839	104,054	151,552	112,311	138,752	105,804
BG512142	199,051*	142,917	219,790	150,058	227,996	143,848
BG521132	138,133	108,324	145,975	115,055	146,709	110,024
BG521142	156,694	131,363	179,427	136,920	157,802	132,604
BG522130	154,581	113,540	164,913	131,120	156,075	121,578
BG522132	147,894	113,382	185,786	120,032	162,389	115,158
BG522142	186,268	137,126	201,118	142,692	202,851	138,077
BK511131	123,699	92,602	128,861	97,923	120,303	94,411
BK511132	125,658	95,323	120,227	99,889	115,416	95,938
BK511141	162,629*	125,307	183,541	129,896	172,762	126,769
BK512131	113,996	90,733	121,357	94,459	113,536	92,058
BK512132	115,697	90,814	118,918	95,929	112,809	91,346
BK521131	118,217	92,350	127,362	97,784	121,292	94,164
BK521132	117,423	94,257	119,690	99,359	118,464	94,957
BK521142	153,805*	124,988	157,186	128,203	154,258	125,480
BK522131	116,340	90,532	119,591	95,153	111,339	91,742
BK522142	154,286	119,559	148,471	122,836	156,557	119,625

Note: * indicates instances that could not be run on our computer with SH's executable, in which case their published results are used.

Instance	SH UB	LB from ℓ, S	Xpress UB	Xpress LB	Heuristic UB	Heuristic LB
CG501120	1,252,308	1,011,260	1,383,799	1,037,547	1,296,913	1,027,177
CG501131	614,303	472,421	652,793	480,446	614,971	478,437
CG501141	777,831	627,035	854,196	631,554	803,432	628,114
CG501121	1,261,525	945,696	1,376,403	964,910	1,247,493	959,756
CG502221	889,548	724,648	1,005,102	730,536	911,616	728,105
CG501132	842,734	561,827	851,427	619,623	849,500	606,568
CG501222	858,289	697,129	983,752	702,660	877,364	699,021
CG501142	1,146,638	754,238	1,169,704	847,853	1,150,718	824,887
CG501122	1,814,877	1,161,383	1,832,565	1,274,366	1,787,833	1,281,687
CG502222	873,858	704,096	1,024,136	709,326	899,970	708,597
CK501120	176,187	141,900	187,551	145,740	179,099	143,260
CK501221	123,206	101,028	133,710	101,409	123,066	101,105
CK501121	173,083	131,993	185,519	134,662	169,804	132,840
CK502221	123,657	101,478	133,305	102,193	122,596	101,899
CK501222	122,485	97,937	132,502	98,522	123,298	98,096
CK501422	128,558	101,864	134,032	104,802	124,315	102,150
CK502222	119,965	98,052	134,820	98,625	122,302	98,282
CK501122	210,824*	153,861	212,018	162,312	206,646	155,485
CK501132	98,363*	75,257	101,602	78,690	98,248	75,782
CK501142	120,722*	90,218	122,122	92,471	115,918	90,673
DG512141	736,181	609,464	1,841,593	613,816	759,136	615,992
DG512131	581,932	465,272	2,052,160	496,633	596,395	469,460
DG012132	3,625,599*	554,595	3,661,662	559,893	3,160,347	555,689
DG012142	3,815,898*	756,588	3,730,720	761,229	3,121,762	756,588
DG012532	1,194,003*	554,167	1,558,743	558,186	1,228,463	555,032
DG012542	1,413,475*	756,062	1,652,806	759,010	1,420,613	756,062
DG512132	2,909,628*	512,330	5,570,581	518,927	3,010,696	514,682
DG512142	3,779,026*	678,733	5,655,268	688,424	3,583,354	682,205
DG512532	584,491	509,567	1,031,986	516,101	597,404	512,147
DG512542	767,428	674,241	1,342,964	683,455	796,831	677,189

Note: * indicates instances that could not be run on our computer with Stadtler's executable, in which case their published results are used.

B.2 Multi-LSB Instances

Instance	LB from ℓ, S	Xpress UB	Xpress LB	Heuristic UB	Heuristic LB	Improvement in UB
SET1_01	17,691	25,873	17,793	22,781	18,840	13.57%
SET1_02	23,534	30,317	23,934	28,624	24,134	5.91%
SET1_03	21,227	26,943	21,638	26,349	21,676	2.25%
SET1_04	21,903	30,352	22,075	26,337	23,175	15.24%
SET1_05	20,446	27,064	21,782	25,621	21,994	5.63%
SET1_06	22,727	30,211	22,857	27,949	23,383	8.09%
SET1_07	20,293	27,316	20,772	24,693	21,125	10.62%
SET1_08	25,278	33,537	25,566	29,810	26,227	12.50%
SET1_09	16,321	24,015	16,780	21,146	17,013	13.57%
SET1_10	17,795	23,985	18,243	22,863	18,945	4.91%
SET1_11	11,080	13,010	12,374	12,956	11,407	0.42%
SET1_12	24,656	27,377	25,211	26,985	25,238	1.45%
SET1_13	20,772	24,003	21,581	23,129	21,195	3.78%
SET1_14	22,162	25,931	22,388	25,720	22,745	0.82%
SET1_15	12,109	15,023	13,070	14,121	12,255	6.39%
SET1_16	14,589	17,542	16,030	17,826	14,861	-1.59%
SET1_17	20,296	24,284	20,576	23,404	20,864	3.76%
SET1_18	10,437	13,272	11,858	12,300	10,859	7.90%
SET1_19	12,964	18,091	13,405	17,448	13,032	3.69%
SET1_20	14,101	19,433	14,236	17,167	14,612	13.20%
SET1_21	10,159	12,428	11,003	12,421	10,392	0.06%
SET1_22	38,040	40,419	38,556	40,158	38,040	0.65%
SET1_23	29,215	30,687	29,622	30,606	29,355	0.26%
SET1_24	28,858	32,282	30,272	32,174	29,250	0.34%
SET1_25	51,194	53,835	51,552	53,279	51,194	1.04%
SET1_26	39,265	42,555	39,376	41,442	39,488	2.69%
SET1_27	40,758	43,929	41,424	43,320	40,918	1.41%
SET1_28	39,800	40,993	40,993	40,993	40,144	0.00%
SET1_29	22,698	25,633	23,698	25,606	23,232	0.11%
SET1_30	68,909	71,387	69,834	70,877	68,909	0.72%

Instance	LB from ℓ, S	Xpress UB	Xpress LB	Heuristic UB	Heuristic LB	Improvement in UB
SET2_01	45,636	59,845	45,977	55,039	45,886	8.73%
SET2_02	47,780	59,525	47,748	57,825	48,159	2.94%
SET2_03	40,551	52,438	40,551	49,147	40,814	6.70%
SET2_04	36,347	47,828	36,347	44,656	36,808	7.10%
SET2_05	45,395	57,950	45,395	55,650	45,784	4.13%
SET2_06	45,902	57,791	45,902	54,361	45,902	6.31%
SET2_07	52,825	63,216	52,825	61,140	53,108	3.40%
SET2_08	48,033	60,193	48,198	56,444	48,632	6.64%
SET2_09	37,064	50,346	37,419	44,523	37,180	13.08%
SET2_10	38,249	51,740	38,616	49,481	38,705	4.57%
SET2_11	65,182	71,019	65,205	69,177	65,648	2.66%
SET2_12	62,179	68,578	62,613	68,578	62,179	0.00%
SET2_13	34,774	41,254	34,774	40,114	34,987	2.84%
SET2_14	62,633	68,070	62,754	68,373	62,633	-0.44%
SET2_15	59,073	62,884	59,078	62,113	59,073	1.24%
SET2_16	75,682	81,490	75,695	79,576	75,682	2.41%
SET2_17	36,014	45,597	36,556	44,047	36,202	3.52%
SET2_18	77,459	84,288	77,728	83,200	77,688	1.31%
SET2_19	54,971	59,724	55,055	59,010	55,484	1.21%
SET2_20	119,376	124,272	119,460	124,482	119,376	-0.17%
SET2_21	20,864	24,517	22,520	24,459	20,864	0.24%
SET2_22	50,537	55,686	51,208	55,539	50,616	0.26%
SET2_23	29,291	35,851	29,921	33,969	29,320	5.54%
SET2_24	65,716	69,365	65,783	68,727	65,716	0.93%
SET2_25	75,243	79,274	75,604	78,428	75,326	1.08%
SET2_26	60,618	63,998	61,564	63,733	60,634	0.42%
SET2_27	52,724	56,100	53,983	54,929	52,769	2.13%
SET2_28	44,310	46,884	45,293	46,772	44,335	0.24%
SET2_29	93,552	97,474	93,735	96,317	93,552	1.20%
SET2_30	68,075	72,830	68,288	71,947	68,353	1.23%

Note: None of these 30 SET2 instances resulted in optimal solution in that allowed time.

Instance	LB from ℓ, S	Xpress UB	Xpress LB	Heuristic UB	Heuristic LB	Improvement in UB
SET3_01	65,668	274,297	74,333	209,129	70,207	31.16%
SET3_02	82,342	337,279	88,125	243,511	88,681	38.51%
SET3_03	74,209	329,113	79,993	235,198	81,830	39.93%
SET3_04	78,282	340,701	86,922	240,339	85,499	41.76%
SET3_05	76,607	331,850	84,114	227,758	83,975	45.70%
SET3_06	79,093	295,468	82,420	235,642	87,790	25.39%
SET3_07	72,979	313,851	78,187	237,218	78,976	32.30%
SET3_08	88,610	332,507	93,913	251,628	96,111	32.14%
SET3_09	64,180	274,953	70,971	216,025	70,301	27.28%
SET3_10	66,878	285,525	73,164	229,242	75,040	24.55%
SET3_11	42,946	184,964	48,991	152,962	48,565	20.92%
SET3_12	86,047	279,693	97,536	217,497	94,746	28.60%
SET3_13	74,643	270,744	84,968	224,670	83,763	20.51%
SET3_14	85,209	296,748	93,522	225,657	95,835	31.50%
SET3_15	40,715	222,557	44,794	167,494	46,252	32.87%
SET3_16	46,548	216,952	53,472	162,616	50,979	33.41%
SET3_17	71,555	241,420	81,449	212,399	79,287	13.66%
SET3_18	39,533	206,693	45,876	112,468	46,879	83.78%
SET3_19	47,495	204,225	54,267	154,981	55,749	31.77%
SET3_20	58,189	257,030	64,603	191,639	64,141	34.12%
SET3_21	44,182	209,485	54,340	150,758	53,149	38.95%
SET3_22	130,235	367,229	138,697	292,199	134,729	25.68%
SET3_23	96,810	282,565	106,346	240,643	108,870	17.42%
SET3_24	105,300	330,999	114,461	292,996	113,949	12.97%
SET3_25	203,044	387,648	209,797	349,975	210,315	10.76%
SET3_26	145,184	349,324	156,009	323,870	162,088	7.86%
SET3_27	145,420	389,594	158,120	343,486	156,422	13.42%
SET3_28	145,227	280,123	154,258	254,008	155,123	10.28%
SET3_29	79,813	276,899	88,871	207,127	90,600	33.69%
SET3_30	274,018	489,154	284,240	431,136	278,826	13.46%

Note: These 30 SET3 instances have an average duality gap of 236% in the allowed time.

Instance	LB from ℓ, S	Xpress UB	Xpress LB	Heuristic UB	Heuristic LB	Improvement in UB
SET4_01	16,353	66,394	22,119	58,720	24,809	13.07%
SET4_02	31,541	83,818	39,072	82,496	41,655	1.60%
SET4_03	24,864	76,954	31,191	73,740	33,577	4.36%
SET4_04	27,786	78,073	33,786	73,651	36,553	6.00%
SET4_05	25,450	75,596	33,836	67,874	34,995	11.38%
SET4_06	30,632	80,132	38,733	79,781	40,893	0.44%
SET4_07	22,650	77,171	29,248	65,736	30,514	17.40%
SET4_08	40,532	100,638	46,045	88,388	49,106	13.86%
SET4_09	13,490	58,504	19,061	57,070	22,860	2.51%
SET4_10	15,542	62,611	22,729	59,319	25,077	5.55%
SET4_11	12,802	35,384	17,589	28,989	17,446	22.06%
SET4_12	43,341	84,540	49,504	78,062	50,961	8.30%
SET4_13	28,152	61,670	34,355	53,833	36,046	14.56%
SET4_14	56,174	85,443	64,907	82,406	64,257	3.69%
SET4_15	14,628	27,634	18,706	26,980	15,797	2.42%
SET4_16	17,171	36,115	23,792	35,280	22,606	2.37%
SET4_17	29,001	55,101	34,116	54,515	36,832	1.07%
SET4_18	19,184	26,279	24,679	26,596	22,950	-1.19%
SET4_19	10,724	40,237	16,512	31,974	15,037	25.84%
SET4_20	18,718	43,396	26,400	39,983	24,035	8.54%
SET4_21	15,812	26,322	22,061	25,899	18,149	1.63%
SET4_22	91,715	123,711	96,535	120,166	93,603	2.95%
SET4_23	55,058	79,912	60,799	76,857	57,446	3.97%
SET4_24	58,919	86,332	64,856	85,119	64,305	1.43%
SET4_25	171,987	201,717	176,087	202,501	173,205	-0.39%
SET4_26	110,570	145,290	117,138	142,090	117,379	2.25%
SET4_27	101,114	146,827	105,713	139,874	103,814	4.97%
SET4_28	112,892	126,027	125,250	129,789	115,176	-2.90%
SET4_29	51,149	71,618	59,369	68,320	56,930	4.83%
SET4_30	241,678	270,746	248,344	267,976	241,712	1.03%

Note: These 30 SET4 instances have an average duality gap of 79% in the allowed time.

Appendix C

Detailed Results on Lower Bounds

C.1 TDS Instances

	Lower Bounds					Upper Bounds	
Instance	ℓ, S	MC	FL	1st LR (Cap)	2nd LR (Lev)	1st LR (Cap)	2nd LR (Lev)
AG501130	116,183	116,600	118,340	117,808	120,764	123,203	127,683
AG501131	107,829	108,106	108,987	109,298	108,822	115,656	117,533
AG501132	118,677	118,957	119,986	120,163	120,454	123,663	128,249
AG501141	133,424	134,008	135,519	135,078	136,547	141,548	147,696
AG501142	145,508	145,873	147,646	146,527	149,002	151,197	156,488
AG502130	122,353	123,904	125,925	125,087	127,119	125,472	134,118
AG502131	109,085	109,501	110,500	111,043	109,959	116,443	121,005
AG502141	134,971	135,527	136,973	136,792	139,060	141,900	146,767
AG502232	97,032	97,488	97,890	98,529	98,206	101,859	102,415
AG502531	102,340	103,252	102,817	103,216	103,211	105,542	109,727
AK501131	96,968	96,983	99,966	97,892	97,811	98,030	112,060
AK501132	101,699	101,781	103,276	102,289	102,847	102,887	109,206
AK501141	134,805	134,943	139,399	135,487	137,303	136,315	163,011
AK501142	134,880	135,006	138,151	135,122	137,867	137,204	151,661
AK501432	92,533	92,605	92,968	94,679	93,270	94,679	93,645
AK502130	102,222	102,245	106,358	103,054	104,351	103,460	117,191
AK502131	93,369	93,423	95,912	93,778	94,338	94,145	101,804
AK502132	96,312	96,396	98,423	96,933	97,644	97,092	104,528
AK502142	127,792	127,977	129,654	128,226	129,863	130,758	138,752
AK502432	88,980	89,088	89,550	90,193	89,995	91,779	91,225

	Lower Bounds					Upper Bounds	
Instance	ℓ, S	MC	FL	1st LR (Cap)	2nd LR (Lev)	1st LR (Cap)	2nd LR (Lev)
BG511132	108,772	109,045	109,875	110,136	109,545	114,629	116,781
BG511142	133,158	133,652	134,424	134,500	134,648	137,991	146,913
BG512131	104,054	104,483	105,158	105,469	104,580	110,855	112,766
BG512132	114,786	115,314	115,894	115,931	115,156	119,395	125,132
BG512142	142,917	143,659	144,840	144,161	145,305	148,340	158,261
BG521132	108,324	108,559	109,338	109,805	109,109	113,609	115,077
BG521142	131,363	131,908	132,996	132,905	133,224	137,629	141,350
BG522130	113,540	114,876	116,472	115,240	115,961	119,850	123,968
BG522132	113,382	113,838	114,305	114,551	114,262	119,158	121,255
BG522142	137,126	137,782	138,608	138,405	138,851	142,417	144,180
BK511131	92,602	92,640	93,964	93,107	93,304	94,310	99,779
BK511132	95,323	95,355	97,283	95,942	96,310	96,844	103,668
BK511141	125,307	125,494	126,753	125,679	126,534	127,256	135,597
BK512131	90,733	90,787	92,253	91,391	91,568	92,036	96,009
BK512132	90,814	90,858	92,896	91,738	91,870	92,208	98,554
BK521131	92,350	92,382	93,469	92,881	92,884	94,004	97,318
BK521132	94,257	94,317	96,197	94,932	95,110	95,914	101,441
BK521142	124,988	125,257	126,384	125,333	126,548	128,448	134,871
BK522131	90,532	90,588	91,731	91,131	91,291	91,802	96,184
BK522142	119,559	119,739	120,794	120,047	120,956	124,160	127,283

	Lower Bounds					Upper Bounds	
Instance	ℓ, S	MC	FL	1st LR (Cap)	2nd LR (Lev)	1st LR (Cap)	2nd LR (Lev)
CG501120	1,011,260	1,012,042	1,025,118	1,012,992	1,017,258	1,022,396	1,109,345
CG501131	472,421	472,711	475,464	473,125	472,947	476,392	513,188
CG501141	627,035	627,631	630,113	628,641	627,980	631,308	678,899
CG501121	945,696	946,442	953,112	948,052	946,612	953,730	1,045,688
CG502221	724,648	725,517	725,827	726,515	724,779	743,421	765,713
CG501132	561,827	562,158	566,137	562,887	567,379	567,636	597,061
CG501222	697,129	698,410	699,934	699,024	697,860	718,231	723,508
CG501142	754,238	757,449	761,826	757,128	764,794	758,835	802,021
CG501122	1,161,383	1,162,216	1,171,502	1,165,839	1,174,289	1,178,726	1,243,710
CG502222	704,096	705,161	707,153	706,766	704,971	725,192	753,284
CK501120	141,900	142,034	143,869	142,581	143,212	145,659	156,264
CK501221	101,028	101,108	101,570	101,299	101,114	103,024	106,030
CK501121	131,993	132,185	133,494	132,708	132,496	137,522	147,865
CK502221	101,478	101,740	102,242	101,968	101,623	103,730	107,423
CK501222	97,937	98,050	98,858	98,313	98,267	100,271	102,163
CK501422	101,864	102,007	102,660	102,135	103,846	102,981	107,102
CK502222	98,052	98,236	98,898	98,450	98,333	100,835	104,359
CK501122	153,861	154,358	156,048	154,841	155,016	155,914	165,574
CK501132	75,257	75,301	76,198	75,648	75,780	76,311	80,388
CK501142	90,218	90,347	91,277	90,477	90,701	91,215	96,230
DG512141	609,464	610,630	611,291	610,613	609,599	610,613	659,071
DG512131	465,272	466,156	466,203	466,333	465,372	466,333	495,481
DG012132	554,595	556,651	559,610	556,441	554,922	556,441	781,344
DG012142	756,588	758,120	763,304	757,387	756,898	757,387	1,001,177
DG012532	554,167	555,261	556,877	555,045	554,167	555,045	775,666
DG012542	756,062	756,956	759,793	756,563	756,159	756,563	982,363
DG512132	512,330	513,440	514,386	512,722	512,376	512,722	554,333
DG512142	678,733	679,821	681,450	679,062	678,777	679,062	854,902
DG512532	509,567	511,041	510,510	510,670	509,587	510,670	542,328
DG512542	674,241	675,180	675,969	674,734	674,241	674,734	715,533

C.2 Multi-LSB Instances

	Lower Bounds					Upper Bounds	
Instance	ℓ, S	MC	FL	1st LR (Cap)	2nd LR (Lev)	1st LR (Cap)	2nd LR (Lev)
SET1_01	17,888	17,888	18,173	17,888	17,888	17,888	17,972
SET1_02	23,534	23,534	23,656	23,534	23,534	23,534	23,534
SET1_03	21,227	21,227	21,346	21,227	21,227	21,227	21,227
SET1_04	22,232	22,232	22,334	22,232	22,232	22,232	22,232
SET1_05	21,446	21,446	21,540	21,446	21,446	21,446	21,446
SET1_06	22,974	22,974	23,072	22,974	22,974	22,974	22,974
SET1_07	20,360	20,360	20,386	20,360	20,360	20,360	20,360
SET1_08	25,582	25,582	25,616	25,582	25,582	25,582	25,582
SET1_09	16,321	16,321	16,442	16,321	16,321	16,321	16,338
SET1_10	17,998	17,998	18,151	17,998	17,998	17,998	18,011
SET1_11	11,080	11,080	11,237	11,080	11,164	11,080	11,169
SET1_12	24,721	24,721	24,762	24,721	24,721	24,721	24,725
SET1_13	20,782	20,788	20,830	20,782	20,782	20,782	20,786
SET1_14	22,264	22,268	22,331	22,264	22,264	22,264	22,264
SET1_15	12,401	12,404	12,805	12,401	12,564	12,401	12,564
SET1_16	15,122	15,122	15,356	15,122	15,543	15,122	15,543
SET1_17	20,468	20,475	20,498	20,468	20,468	20,468	20,468
SET1_18	11,075	11,077	11,366	11,075	11,462	11,075	11,462
SET1_19	13,276	13,276	13,528	13,276	13,388	13,276	13,388
SET1_20	14,101	14,101	14,177	14,101	14,101	14,101	14,113
SET1_21	10,159	10,166	10,429	10,159	10,325	10,159	10,325
SET1_22	38,040	38,056	38,166	38,040	38,040	38,040	38,077
SET1_23	29,331	29,343	29,376	29,331	29,331	29,331	29,331
SET1_24	28,858	28,858	29,074	28,858	28,886	28,858	28,886
SET1_25	51,371	51,371	51,403	51,371	51,371	51,371	51,371
SET1_26	39,379	39,379	39,463	39,379	39,402	39,379	39,402
SET1_27	40,838	40,838	40,838	40,838	40,838	40,838	40,838
SET1_28	39,846	39,864	39,894	39,846	39,857	39,846	39,857
SET1_29	23,155	23,165	23,275	23,155	23,182	23,155	23,182
SET1_30	68,989	68,989	69,074	68,989	68,989	68,989	68,989

	Lower Bounds					Upper Bounds	
Instance	ℓ, S	MC	FL	1st LR (Cap)	2nd LR (Lev)	1st LR (Cap)	2nd LR (Lev)
SET2_01	46,116	46,116	46,207	46,116	46,116	46,116	46,116
SET2_02	47,780	47,780	47,861	47,780	47,780	47,780	47,780
SET2_03	40,551	40,551	40,610	40,551	40,551	40,551	40,551
SET2_04	36,347	36,347	36,564	36,347	36,347	36,347	36,430
SET2_05	45,395	45,395	45,508	45,395	45,395	45,395	45,395
SET2_06	45,902	45,902	45,939	45,902	45,902	45,902	45,902
SET2_07	52,825	52,825	52,939	52,825	52,825	52,825	52,825
SET2_08	48,033	48,033	48,280	48,033	48,084	48,033	48,084
SET2_09	37,553	37,553	37,661	37,553	37,553	37,553	37,553
SET2_10	38,751	38,751	38,898	38,751	38,751	38,751	38,751
SET2_11	65,210	65,211	65,213	65,210	65,210	65,210	65,210
SET2_12	62,792	62,792	62,979	62,792	62,803	62,792	62,803
SET2_13	34,778	34,778	34,882	34,778	34,885	34,778	34,885
SET2_14	62,907	62,907	62,993	62,907	62,907	62,907	62,916
SET2_15	59,079	59,079	59,125	59,079	59,079	59,079	59,079
SET2_16	75,682	75,682	75,698	75,682	75,682	75,682	75,682
SET2_17	36,809	36,818	36,918	36,809	36,826	36,809	36,935
SET2_18	77,873	77,874	77,935	77,873	77,873	77,873	77,873
SET2_19	54,981	54,981	55,120	54,981	55,026	54,981	55,026
SET2_20	119,568	119,568	119,588	119,568	119,568	119,568	119,568
SET2_21	22,281	22,315	22,557	22,281	22,643	22,281	22,643
SET2_22	51,279	51,279	51,439	51,279	51,414	51,279	51,414
SET2_23	29,793	30,067	30,210	29,793	29,814	29,793	29,815
SET2_24	65,891	65,891	65,984	65,891	65,891	65,891	65,891
SET2_25	75,627	75,628	75,745	75,627	75,705	75,627	75,705
SET2_26	60,952	61,002	61,173	60,952	60,988	60,952	60,988
SET2_27	53,016	53,016	53,052	53,016	53,016	53,016	53,441
SET2_28	44,545	44,552	44,705	44,545	44,923	44,545	44,923
SET2_29	93,631	93,638	93,659	93,631	93,632	93,631	93,632
SET2_30	68,324	68,333	68,573	68,324	68,324	68,324	68,324

	Lower Bounds					Upper Bounds	
Instance	ℓ, S	MC	FL	1st LR (Cap)	2nd LR (Lev)	1st LR (Cap)	2nd LR (Lev)
SET3_01	65,668	71,594	71,584	66,984	65,761	66,984	112,652
SET3_02	82,342	89,855	89,887	84,865	82,704	84,865	105,740
SET3_03	74,209	82,398	82,440	77,086	74,611	77,086	99,483
SET3_04	78,282	85,258	85,229	80,716	78,436	80,716	108,664
SET3_05	76,607	83,692	83,667	78,931	76,884	78,931	102,852
SET3_06	79,093	88,689	88,737	82,910	79,625	82,910	112,534
SET3_07	72,979	79,067	79,181	75,365	73,098	75,365	105,466
SET3_08	88,610	94,504	94,481	92,108	89,213	92,108	129,505
SET3_09	64,180	67,768	67,760	64,336	64,180	64,336	85,114
SET3_10	66,878	74,333	74,324	67,928	66,912	67,928	92,540
SET3_11	42,946	46,063	45,997	43,902	43,012	43,902	69,501
SET3_12	86,047	95,953	95,980	90,412	87,641	90,412	112,402
SET3_13	74,643	81,477	81,348	75,379	74,987	75,379	102,771
SET3_14	85,209	91,252	91,435	86,813	85,493	86,813	102,438
SET3_15	40,715	43,551	43,343	40,843	40,750	40,843	74,085
SET3_16	46,548	50,868	50,784	48,528	48,360	48,528	62,509
SET3_17	71,555	78,132	77,988	72,458	71,837	72,458	95,764
SET3_18	39,533	40,406	40,259	39,658	39,616	39,658	57,199
SET3_19	47,495	50,636	50,497	48,266	47,636	48,266	84,711
SET3_20	58,189	60,240	60,125	58,529	59,753	58,529	95,852
SET3_21	44,182	45,435	45,383	44,359	44,182	44,359	60,262
SET3_22	130,235	138,607	138,279	133,995	130,930	133,995	142,716
SET3_23	96,810	102,993	102,912	99,719	96,939	99,719	115,205
SET3_24	105,300	110,117	109,994	105,327	105,300	105,327	136,353
SET3_25	203,044	210,031	209,928	204,955	203,044	204,955	212,110
SET3_26	145,184	152,864	152,545	146,938	145,198	146,938	155,347
SET3_27	145,420	154,121	153,805	148,698	145,674	148,698	169,988
SET3_28	145,227	153,083	153,327	147,940	145,927	147,940	162,729
SET3_29	79,813	87,043	86,551	81,494	80,206	81,494	96,912
SET3_30	274,018	283,252	282,958	276,810	274,018	276,810	284,338

	Lower Bounds					Upper Bounds	
Instance	ℓ, S	MC	FL	1st LR (Cap)	2nd LR (Lev)	1st LR (Cap)	2nd LR (Lev)
SET4_01	16,353	16,532	18,093	16,353	16,951	16,353	23,694
SET4_02	31,541	32,773	34,074	31,541	31,726	31,541	33,919
SET4_03	24,864	25,616	27,464	24,864	24,864	24,864	28,061
SET4_04	27,786	28,837	30,023	27,786	27,928	27,786	31,426
SET4_05	25,450	26,353	27,335	25,450	25,450	25,450	29,755
SET4_06	30,632	31,495	32,990	30,632	31,054	30,632	35,402
SET4_07	22,650	23,189	24,599	22,650	23,884	22,650	30,365
SET4_08	40,532	42,512	43,131	40,532	40,538	40,532	41,812
SET4_09	13,490	13,557	14,687	13,490	14,650	13,490	19,585
SET4_10	15,542	15,553	16,857	15,542	16,041	15,542	26,902
SET4_11	12,802	12,996	13,825	12,802	13,675	12,802	15,205
SET4_12	43,341	44,527	45,100	43,341	44,523	43,341	46,502
SET4_13	28,152	28,736	30,049	28,152	28,152	28,152	33,352
SET4_14	56,174	57,052	57,302	56,174	56,406	56,174	57,049
SET4_15	14,628	14,715	15,304	14,628	15,244	14,628	16,260
SET4_16	17,171	17,529	17,990	17,172	17,662	17,172	19,874
SET4_17	29,001	29,886	30,581	29,225	29,237	29,225	31,729
SET4_18	19,184	19,213	19,309	19,185	19,705	19,185	19,997
SET4_19	10,724	10,769	11,780	10,724	12,581	10,724	15,411
SET4_20	18,718	18,858	19,702	18,731	19,420	18,731	21,014
SET4_21	15,812	16,243	16,819	15,812	16,386	15,812	17,720
SET4_22	91,715	93,010	93,185	91,733	92,228	91,733	92,310
SET4_23	55,058	55,601	56,077	55,151	55,562	55,151	56,132
SET4_24	58,919	59,231	59,512	58,919	59,213	58,919	60,947
SET4_25	171,987	172,779	172,904	171,987	171,987	171,987	171,988
SET4_26	110,570	111,393	111,703	110,570	110,570	110,570	110,577
SET4_27	101,114	102,197	102,182	101,471	101,267	101,471	101,340
SET4_28	112,892	113,353	114,022	112,892	112,987	112,892	112,987
SET4_29	51,149	51,394	51,776	51,149	51,253	51,149	51,253
SET4_30	241,678	243,702	243,998	241,801	241,678	241,801	241,693

Appendix D

Characteristics of 2PCLS

Instance	Demand	Capacity	Setup Time	Inv.?
2pcls01	19,3,7,7,14,11	53,47	7,4,5	Yes
2pcls02	1,39,41,42,7,32	201,147	31,29,11	Yes
2pcls03	21,18,19,23,20,22	91,89	3,5,4	Yes
2pcls04	21,19,27,11	51,53	4,5	Yes
2pcls05	7,9,15,18,11,5	44,49	0,0,0	Yes
2pcls06	1,7,13,11,9,3	36,29	0,4,0	Yes
2pcls07	1,3,5,4	14,11	0,0	Yes
2pcls08	11,13,3,7	28,13	0,0	Yes
2pcls09	11,13,3,7	28,13	0,0	No
2pcls10	21,18,19,23,20,22	91,89	3,5,4	No
2pcls11	1,39,41,42,7,32	201,147	31,29,11	No
2pcls12	7,9,15,18,11,5	47,30	0,0,0	No
2pcls13	7,9,15,3,18,11,5,13	49,40	0,0,0,0	No
2pcls14	26,47,43,58,23,13,9,58	201,93	0,0,0,0	No
2pcls15	3,1,12,41,7,23,38,44	89,106	0,0,0,0	No
2pcls16	21,17,21,11,14,6,20,4	89,39	0,0,0,0	Yes
2pcls17	24,39,15,36,40,13,26,26,34,28	199,111	0,0,0,0,0	Yes
2pcls18	44,14,7,5,31,49,16,4,55,46	179,179	2,6,2,12,3	Yes
2pcls19	49,29,14,32,51,48,51,49,37,18,36,16	348,288	10,13,1,11,5,12	Yes
2pcls20	5,4,20,31,29,14,17,27,14,13,34,25	121,121	0,0,0,0,0,0	Yes

Appendix E

Sample cdd Files

Input Files

* file name: lotsize1.ine

* 3 items, 2 periods

* Demands = (19,7;3,14;7,11)

* Setup times = (7,4,5)

* Capacities = (53,47)

H-representation

begin

41 16 integer

26	-1	-1	0	0	1	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	-1	-1	0	0	1	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	-1	-1	0	0	1
0	-1	0	46	0	0	0	0	0	0	0	0	0	0	0	0
0	0	-1	0	40	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	-1	0	49	0	0	0	0	0	0	0
0	0	0	0	0	0	0	-1	0	43	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	-1	0	48	0	0

0	0	0	0	0	0	0	0	0	0	0	0	-1	0	42	0
53	-1	0	-7	0	0	-1	0	-4	0	0	-1	0	-5	0	0
47	0	-1	0	-7	0	0	-1	0	-4	0	0	-1	0	-5	0
0	-1	0	26	0	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	-1	0	17	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	-1	0	18	0	1
0	0	-1	0	7	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	-1	0	14	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	-1	0	11	1
0	-1	-1	26	7	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	-1	-1	17	14	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	-1	-1	18	11	1
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

```
 0  0  0  0   0  0  0  0  0   0  0  0  0  1  0  0
 0  0  0  0   0  0  0  0  0   0  0  0  0  0  1  0
 0  0  0  0   0  0  0  0  0   0  0  0  0  0  0  1
 1  0  0 -1   0  0  0  0  0   0  0  0  0  0  0  0
 1  0  0  0  -1  0  0  0  0   0  0  0  0  0  0  0
 1  0  0  0   0  0  0  0 -1   0  0  0  0  0  0  0
 1  0  0  0   0  0  0  0  0  -1  0  0  0  0  0  0
 1  0  0  0   0  0  0  0  0   0  0  0  0 -1  0  0
 1  0  0  0   0  0  0  0  0   0  0  0  0  0 -1  0
end
```

```
* file name: lotsize4.ine
* 3 items, 2 periods
* Demands = (21,27;19,11)
* Setup times = (4,5)
* Capacities = (51,53)
H-representation
begin
28          11          integer
 48  -1  -1   0   0   1   0   0   0   0   0
 40   0   0   0   0   0  -1  -1   0   0   1
  0  -1   0  47   0   0   0   0   0   0   0
  0   0  -1   0  46   0   0   0   0   0   0
  0   0   0   0   0   0  -1   0  49   0   0
```

0	0	0	0	0	0	0	-1	0	48	0
51	-1	0	-4	0	0	-1	0	-5	0	0
53	0	-1	0	-4	0	0	-1	0	-5	0
0	-1	0	48	0	1	0	0	0	0	0
0	0	0	0	0	0	-1	0	40	0	1
0	0	-1	0	27	1	0	0	0	0	0
0	0	0	0	0	0	0	-1	0	11	1
0	-1	-1	48	27	1	0	0	0	0	0
0	0	0	0	0	0	-1	-1	40	11	1
0	1	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0	1
1	0	0	-1	0	0	0	0	0	0	0
1	0	0	0	-1	0	0	0	0	0	0
1	0	0	0	0	0	0	0	-1	0	0
1	0	0	0	0	0	0	0	0	-1	0

end

Output Files

```
* cdd: Double Description Method C-Code:Version 0.61 (December 1, 1997)
* Copyright (C) 1996, Komei Fukuda, fukuda@ifor.math.ethz.ch
*Input File:lotsize1.ine ( 41 x 16)
*HyperplaneOrder: LexMin
*Degeneracy preknowledge for computation: None (possible degeneracy)
*Vertex/Ray enumeration is chosen.
*Computation starts at Sun Feb 11 19:31:11 2007
*                    terminates at Sun Feb 11 19:33:03 2007
*Total processor time = 112 seconds
*                              = 0 hour 1 min 52 sec
*FINAL RESULT:
*Number of Vertices =          13984, Rays =          3
V-representation
begin
13987          16          real
1 26 0 1 0 0 0 0 1 0 0 7 42 1 1 31
1 26 0 1 0 0 0 0 1 0 0 7 42 3.888888889E-01 1 31
1 26 0 1 0 0 0 0 0 0 0 7 42 1 1 31
1 26 0 1 0 0 0 0 0 0 0 7 42 3.888888889E-01 1 31
1 26 0 1 0 0 0 43 1 1 29 0 0 1 0 0
1 26 0 1 0 0 0 43 0 1 29 0 0 1 0 0
```

```
* cdd: Double Description Method C-Code:Version 0.61 (December 1, 1997)
* Copyright (C) 1996, Komei Fukuda, fukuda@ifor.math.ethz.ch
*Input File:lotsize4.ine ( 28 x 11)
*HyperplaneOrder: LexMin
*Degeneracy preknowledge for computation: None (possible degeneracy)
*Vertex/Ray enumeration is chosen.
*Computation starts at Sun Feb 11 21:23:53 2007
*                    terminates at Sun Feb 11 21:23:53 2007
*Total processor time = 0 seconds
*                              = 0 hour 0 min 0 sec
*FINAL RESULT:
*Number of Vertices =          591, Rays =          2
V-representation
begin
593          11          real
1 47 1 1 1 0 0 0 0 1 0
1 47 1 1 3.703703704E-02 0 0 0 0 1 0
1 47 1.703703704E+00 1 3.703703704E-02 7.037037037E-01 0 0 0 1 0
1 47 1 1 1 0 0 11 0 1 0
1 47 1 1 3.703703704E-02 0 0 11 0 1 0
1 47 1.703703704E+00 1 3.703703704E-02 7.037037037E-01 0 11 0 1 0
1 47 1 1 1 0 0 0 0 0 0
1 47 46 1 1 45 0 0 0 0 0
```

Appendix F

2-Period Convex Hull Closure Implementation in Mosel

```
model "closure"

uses "mmxprs"              !Load the libraries
uses "mmsystem"

declarations
NI=6          !number of items
NT=15          !number of periods
item= 1..NI
period= 1..NT
x: array(period,item) of mpvar        !production amount
y: array(period,item) of mpvar        !setup variable
s: array(period,item) of mpvar        !stock variable
dem: array(period,item) of real       ! demand
c: real                               ! general capacity
cap: array(period,item) of real       ! capacity/item
rho: array(item) of real
st: array(item) of real               ! setup time
f: array(item) of real                ! setup cost
```

```
h: array(item) of real                  ! inventory cost
echd: array (period,item) of real       !echelon demand
! The following used for (l,S) inequalities:
maxiter = 150                           ! Max num of iterations
iter = 1..maxiter                       ! iterations
setS: array(iter,period,period,item) of integer
! Set S (1 if t in S, 0 otherwise) for each iteration + period l
d: array(period,period,item) of real    ! Total demand until period l
countviol: integer        ! counter for # of violations at each iteration
end-declarations

! ==================================================================
! The following block is primarily used to input data and calculate some
! parameters:
fopen("g30n",F_INPUT)
readln
readln(c)
forall (i in item) do
readln(rho(i),", ",h(i),", ",st(i),", ",f(i))
end-do
forall (t in period) do
   forall (i in item) read(dem(t,i),", ")
   readln
end-do
fclose(F_INPUT)
forall (t in period, i in item) cap(t,i):= c - st(i)
! ==================================
! Main block of the basic formulation:
```

```
! Flow balance:
forall (i in item)
FL1(i):= x(1,i) = dem(1,i)+s(1,i)
forall (t in 2..NT, i in item)
FL2(t,i):= s(t-1,i)+x(t,i) = dem(t,i)+s(t,i)
! Capacity contraint per item:
forall (t in period, i in item)
VUB(t,i):= x(t,i) <= cap(t,i)*y(t,i)
! Total capacity constraint:
forall (t in period)
capa(t):= sum(i in item) (x(t,i)+ st(i)*y(t,i)) <= c
! Objective function:
cost:= sum(t in period,i in item) (f(i)*y(t,i)+h(i)*s(t,i))
! ==================================================================
! BASE FORMULATION ENDS HERE, NEXT, L,S INEQUALITIES ARE GENERATED
! ==================================================================
! Define echelon demand:
forall (t in period, i in item) do
      echd(t,i):= dem(t,i)
end-do
! Relax the setup variables:
forall (t in period, i in item)
necessary(t,i):= y(t,i) <= 1
starttime := gettime
! First, calculate the demands dem(t,l) for each period t,l:
forall (i in item, l in 1..NT) do      ! for each l and item i
      ! First, initialize the d(t,l) as zero:
      forall (t in 1..l) d(t,l,i):= 0
```

```
      ! Then, calculate all the d(t,l) quantities:
      d(l,l,i) := echd(l,i)
      if (l >= 2) then
      forall (t in 1..(l-1))
         d(l-t,l,i) := d(l-t+1,l,i) + echd(l-t,i)
      end-if
end-do
! First, solve the LP relaxation:
minimize(cost)
writeln('The LP relaxation solution:')
writeln('==========================')
writeln('The total cost:', getobjval)
writeln(' ')
ls_soln:= getobjval
! Use a counter for total number of ineq. added to original problem:
totalviol:= 0
forall(it in iter, i in item,l in 1..NT) deleted_ls_index(it,i,l):= 0
total_deleted:= 0
forall(iteration in iter) do   !!! MAIN LOOP STARTS HERE !!!
! initialize the counter:
countviol := 0
!!!!!! (l,S) INEQUALITIES:
forall (i in item) do          ! for each item
   forall (l in 1..NT) do       ! for each l
      ! Initialize the set S:
      forall (t in 1..l) setS(iteration,t,l,i) := 0
      ! Determine on the set of S, i.e.
```

```
        ! whether x(t,i) > d(t,l,i)*y(t,i) or not,
        ! and then add the ones satisfying this inequality:
        forall (t in 1..l) do
           if (getsol(x(t,i)) > d(t,l,i)*getsol(y(t,i))) then
              setS(iteration,t,l,i) := 1
           end-if
        end-do
        ! Add the maximum violated (L,S) inequality, if there is any,
        ! and increase the counter:
        if ((sum(t in 1..l) setS(iteration,t,l,i)*getsol(x(t,i))) >
        getsol(s(l,i)) + (sum(t in 1..l) setS(iteration,t,l,i)*d(t,l,i)
        *getsol(y(t,i))) + 0.000001) then
           addcons(iteration,i,l):= (sum(t in 1..l) setS(iteration,t,l,i)
           *x(t,i)) <= s(l,i) + sum(t in 1..l) setS(iteration,t,l,i)
           *d(t,l,i)*y(t,i)
           countviol := countviol + 1
        end-if
     end-do
  end-do
  if (countviol = 0) then break
  else
  ! All the max violated (l,S) inequalities for this iteration are added.
  ! Solve the strengthened LP relaxation:
  minimize(cost)
  if (getobjval-ls_soln < 0.1) then
     writeln(" !!! loop control in effect !!! ")
     break
  end-if
```

```
ls_soln:= getobjval
! Delete inactive l,s cuts:
deleted_ls:= 0
forall(it in 1..iteration, i in item,l in 1..NT) do
   if ((-0.1>getsol(addcons(it,i,l)) or getsol(addcons(it,i,l))>0.1)
   and deleted_ls_index(it,i,l)<>1) then
      deleted_ls_index(it,i,l):= 1
      sethidden(addcons(it,i,l),true)
      deleted_ls:= deleted_ls + 1
   end-if
end-do
total_deleted:= total_deleted + deleted_ls
writeln("DENEME -- ",deleted_ls)
writeln('The LP relaxation solution with added (l,S) inequalities:')
writeln("=================================================================")
writeln('Iteration:',iteration)
writeln('Number of constraints added:',countviol)
writeln('The total cost:', getobjval)
totalviol:= totalviol + countviol
writeln(' ')
num_iter:= iteration
end-if
end-do        !!! l,s LOOP ENDS HERE !!!
! Print out the general statistics for ls inequalities:
writeln(' ')
writeln("LS INEQUALITIES STATS:")
writeln("=========================")
writeln("# valid inequalities = ",totalviol)
```

```
writeln('Total number deleted:',total_deleted)
ls_time:= gettime-starttime
writeln("Time spent            = ",ls_time)
writeln("Number of iterations = ",num_iter)
writeln(' ')

! ================================
!  2-PERIOD CONVEX HULL CLOSURE ===
! ================================

declarations
two=0..1
horizon: array(iter,period,item) of integer
xx: array(two,item) of mpvar          !production var used for 2-closure
yy: array(two,item) of mpvar          !setup var used for 2-closure
ss: array(item) of mpvar              !stock var used for 2-closure
linearx: array(two,item) of real      !LPR value for x
lineary: array(two,item) of real      !LPR value for y
linears: array(item) of real          !LPR value for s
errorxp: array(two,item) of mpvar     !positive error for prod var
erroryp: array(two,item) of mpvar     !positive error for binary var
errorxn: array(two,item) of mpvar     !negative error for prod var
erroryn: array(two,item) of mpvar     !negative error for binary var
errorsn: array(item) of mpvar         !negative error for stock var
lambda: array(iter) of mpvar          !lambda variable
bSolutionNotFound: boolean
totalerror: mpvar       !obj func value for error minimization
minredcost: mpvar       !obj func value for reduced cost minimization
```

```
extx: array(iter,two,item) of real   !extreme points
exty: array(iter,two,item) of real   !extreme points
exts: array(iter,item) of real       !extreme points
end-declarations

forall (it in iter,t in period,ite in iter) minredcost is_free
forall (it in iter,t in period,ite in iter) totalerror is_free
forall (it in iter,t in period,ite in iter) obj_dual(it,t,ite) is_free

!!!!!!!!!!!!!!!!!!!!!
!!   MAIN LOOP 1    !!
!!!!!!!!!!!!!!!!!!!!!
forall (it in iter) do ! ITERATIONS SOLVING LPR
countcuts:= 0
! Obtain the LPR solution
setparam("XPRS_VERBOSE",true)
minimize(cost)
setparam("XPRS_VERBOSE",false)
writeln("FINAL SOLUTION =   ",getobjval, " at round ",it)
! Initialize linearx,y,s values...
forall (t in period,i in item) linx(it,t,i):= getsol(x(t,i))
forall (t in period,i in item) liny(it,t,i):= getsol(y(t,i))
forall (t in period,i in item) lins(it,t,i):= getsol(s(t,i))
   !!!!!!!!!!!!!!!!!!!!!
   !!   MAIN LOOP 2    !!
   !!!!!!!!!!!!!!!!!!!!!
   forall (t in 1..NT-1) do ! ITERATION FOR EACH t PROBLEM
   ! Initialize horizon:
```

```
forall (i in item) do
   ini:= 0.99*random+0.01
   ttt:= t+1
   repeat
      horizon(it,t,i):= ttt
      ttt:= ttt+1
   until ttt=NT+1 or (liny(it,t+1,i)+ini <
   sum(tt in t+1..ttt) liny(it,tt,i))
   writeln("Horizon for item ",i," = ",horizon(it,t,i))
end-do
! Initialize LPR soln
forall (tt in two,i in item) linearx(tt,i):=linx(it,t+tt,i)
forall (tt in two,i in item) lineary(tt,i):=liny(it,t+tt,i)
forall (i in item) linears(i):=lins(it,horizon(it,t,i),i)
! Initialize ext points to 0
forall (tt in two,i in item) extx(1,tt,i):= 0
forall (tt in two,i in item) exty(1,tt,i):= 0
forall (i in item) exts(1,i):= 0
ite:=0
itiszero:= false
itisover:= false
   !!!!!!!!!!!!!!!!!!!!!
   !!    MAIN LOOP 3    !!
   !!!!!!!!!!!!!!!!!!!!!
   repeat ! Iteration for each distance prob + dual + col gen.

   ite:= ite + 1
```

```
! Solve the distance problem
obj_func(it,t,ite):= totalerror = (sum(i in item) errorsn(i)) +
sum(tt in two,i in item) (errorxp(tt,i) + errorxn(tt,i) +
erroryp(tt,i) + erroryn(tt,i))
forall (tt in two,i in item) equalityx(it,t,ite,tt,i):=
linearx(tt,i) = sum(k in 1..ite) lambda(k)*extx(k,tt,i) +
errorxp(tt,i) - errorxn(tt,i)
forall (tt in two,i in item) equalityy(it,t,ite,tt,i):=
lineary(tt,i) = sum(k in 1..ite) lambda(k)*exty(k,tt,i) +
erroryp(tt,i) - erroryn(tt,i)
forall (i in item) equalitys(it,t,ite,i):= linears(i) >=
sum(k in 1..ite) lambda(k)*exts(k,i) - errorsn(i)
convexsum(it,t,ite):= sum(k in 1..ite) lambda(k) <= 1
minimize(totalerror)
bSolutionNotFound := getparam("XPRS_LPSTATUS") = XPRS_LP_INFEAS
writeln("Total Manhattan distance = ", getobjval)
if (bSolutionNotFound) then
   writeln("WARNING! Something badly wrong!... ")
elif (getobjval<=0) then
   itiszero:= true
   writeln("Distance zero; LPR solution in the convex hull")
elif (ite=maxiter) then
   writeln("Max # for iterations reached ... ")
else
   ! Dual of the distance problem for dual variable values
   writeln("Solving dual of distance problem ... ")
   forall (tt in two,i in item) alpha(it,t,tt,i):=
   -getdual(equalityx(it,t,ite,tt,i))
```

```
forall (tt in two,i in item) beta(it,t,tt,i):=
-getdual (equalityy(it,t,ite,tt,i))
forall (i in item) gamma(it,t,i):=
-getdual(equalitys(it,t,ite,i))
eta(it,t):= getdual(convexsum(it,t,ite))
dualobj:= (sum(i in item) gamma(it,t,i)*linears(i)) +
sum (tt in two,i in item) (alpha(it,t,tt,i)*linearx(tt,i) +
beta(it,t,tt,i)*lineary(tt,i)) + eta(it,t)
!!! Column generation for ext points - solve max problem
forall (i in item) closure1(it,t,ite,i):= sum(tt in two) xx(tt,i)
<= (sum(tt in t..horizon(it,t,i)) dem(tt,i))+ss(i)
forall (tt in two, i in item) closure2(it,t,ite,tt,i):= xx(tt,i)
<= cap(t+tt,i)*yy(tt,i)
forall (tt in two) closure3(it,t,ite,tt):= sum(i in item)
(xx(tt,i) + st(i)*yy(tt,i)) <= c
forall (tt in two, i in item) closure4(it,t,ite,tt,i):= xx(tt,i)
<= (sum(tt2 in t+tt..horizon(it,t,i)) dem(tt2,i))*yy(tt,i)+ss(i)
forall (i in item) closure5(it,t,ite,i):= sum(tt in two) xx(tt,i)
<= (sum(tt in t..horizon(it,t,i)) dem(tt,i))*yy(0,i) +
(sum(tt in t+1..horizon(it,t,i)) dem(tt,i))*yy(1,i) + ss(i)
forall (tt in two,i in item) setyy(it,t,ite,tt,i):= yy(tt,i)
is_binary
obj_func2(it,t,ite):= minredcost = (sum(i in item) gamma(it,t,i)
*ss(i)) + sum (tt in two,i in item) (alpha(it,t,tt,i)*xx(tt,i)
+ beta(it,t,tt,i)*yy(tt,i)) + eta(it,t)
SolnNotFound := false
maximize(minredcost)
SolnNotFound:= getparam("XPRS_MIPSTATUS") = XPRS_MIP_INFEAS
```

```
    obj_minred:= getsol(minredcost)
    if (SolnNotFound) then
       writeln("WARNING! Something wrong in 2-closure problem!... ")
    elif (getobjval <= 0) then
       itisover:= true
       writeln("No more useful ext point - col gen over")
    else
       writeln("Generating extreme point...")
       forall(tt in two,i in item) extx(ite+1,tt,i):=getsol(xx(tt,i))
       forall(tt in two,i in item) exty(ite+1,tt,i):=getsol(yy(tt,i))
       forall (i in item) exts(ite+1,i):= getsol(ss(i))
    end-if
    sethidden(obj_func2(it,t,ite),true)
    forall (i in item) sethidden(closure1(it,t,ite,i),true)
    forall (tt in two, i in item) sethidden(closure2(it,t,ite,tt,i),
    true)
    forall (tt in two) sethidden(closure3(it,t,ite,tt),true)
    forall (tt in two, i in item) sethidden(closure4(it,t,ite,tt,i),
    true)
    forall (i in item) sethidden(closure5(it,t,ite,i),true)
    forall (tt in two,i in item) sethidden(setyy(it,t,ite,tt,i),true)
end-if
sethidden(obj_func(it,t,ite),true)
forall (tt in two,i in item) sethidden(equalityx(it,t,ite,tt,i),
true)
forall (tt in two,i in item) sethidden(equalityy(it,t,ite,tt,i),
true)
forall (i in item) sethidden(equalitys(it,t,ite,i),true)
```

```
      sethidden(convexsum(it,t,ite),true)
      until (ite=maxiter) or (itiszero) or (itisover)
      or (bSolutionNotFound) or (SolnNotFound)
      ! END OF Iteration for each dist prob + dual + col gen.
      if (bSolutionNotFound or SolnNotFound) then
         writeln("Stopping execution ... ")
         break 2
      elif(ite=maxiter) then
         writeln("Max # iterations - problem unfinished for t = ",t)
         writeln("TEST CUT ... t = ",t)
         countcuts:= countcuts + 1
         cut_extra(it,t):= obj_minred
         add_cut(it,t):= (sum(i in item) gamma(it,t,i)*
         s(horizon(it,t,i),i)) + sum (tt in two,i in item)
         (alpha(it,t,tt,i)*x(t+tt,i) + beta(it,t,tt,i)*y(t+tt,i)) +
         eta(it,t) <= cut_extra(it,t)
      elif (itiszero) then
         writeln("LPR solution in the convex hull of problem at t = ",t)
      elif (itisover) then
         countcuts:= countcuts + 1
         writeln("Adding a cut ... at t = ",t)
         add_cut(it,t):= (sum(i in item) gamma(it,t,i)*
         s(horizon(it,t,i),i)) + sum (tt in two,i in item)
         (alpha(it,t,tt,i)*x(t+tt,i) + beta(it,t,tt,i)*y(t+tt,i)) +
         eta(it,t) <= 0
      end-if
   end-do ! END OF ITERATION FOR EACH t PROBLEM
if (countcuts=0) then
```

```
    writeln("No cuts generated at iter ",it,", stopping execution ... ")
    break
end-if
end-do   ! END OF ITERATIONS SOLVING LPR
setparam("XPRS_VERBOSE",true)
minimize(cost)
setparam("XPRS_VERBOSE",false)
writeln("FINAL SOLUTION =    ",getobjval)
end-model
```