DECISION MODELS FOR MULTISTAGE PRODUCTION PLANNING

by

Paul Thayer Chapman

B.A., University of North Carolina at Chapel Hill
(1979)

S.M., Massachusetts Institute of Technology
(1982)

SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS OF THE
DEGREE OF

DOCTOR OF PHILOSOPHY
IN OPERATIONS RESEARCH

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

October, 1985

I hereby assign my copyright of this thesis to The Charles Stark Draper Laboratory, Inc.,
Cambridge, Massachusetts.

Paul Thayer Chapman

Permission is hereby granted by The Charles Stark Draper Laboratory, Inc. to the
Massachusetts Institute of Technology to reproduce any or all of this thesis.

Signature of Author

Operations Research Center, October 11, 1985

Certified by

Jeremy F. Shapiro, Thesis Supervisor

Accepted by

Jeremy F. Shapiro, Chairman, Interdepartmental
Committee in Operations Research

APR 11, 1986

LIBRARIES Archives
Publication of this document does not constitute approval by The Charles Stark Draper Laboratory, Inc. of the findings or conclusions contained herein. It is published for the exchange and stimulation of ideas.
DECISION MODELS FOR MULTISTAGE PRODUCTION PLANNING

by

PAUL THAYER CHAPMAN

Submitted to the Department of Electrical Engineering and Computer Science
on October 11, 1985 in partial fulfillment of the
requirements for the Degree of Doctor of Philosophy in
Operations Research

ABSTRACT

In this work we address production scheduling models for multistage discrete parts
manufacturing environments. The models are mathematical programming formulations whose
objective is to meet demands for production items while minimizing avoidable production costs.
The manufacturing environment is multistage in the sense that a production item may require
subassemblies as components, and the subassemblies themselves may require other
subassemblies as components, etc.

The models are quite unrestricted, general purpose in nature, and relatively complex. For
example, a general product structure is allowed where there are virtually no limits on the
relations of production items. Finished goods and subassemblies can share components in a
variety of ways, and production items may have multiple successors and predecessors within
the product structure. We deal with environments with capacitated production facilities, a
factor which contributes to the complexity of the problem. Moreover, we allow for "initial state"
conditions that include inventories and "holdover production" at the beginning of the planning
horizon. The rationale behind working with such general models is to develop methodologies
for obtaining workable production schedules from models that reflect the real complexities of a
manufacturing environment, verses simplifying the models so that theoretically elegant
optimization algorithms can be created at the expense of practical relevance.

Most production scheduling models assume fixed demands for production items in the time
periods that comprise the planning horizon. Whereas we deal with such demand schedules, we
also generalize to models in which a scheduler has specific customer orders for finished goods.
Part of the scheduling problem, then, is to determine which of the orders can be met and when
these orders will be shipped. Thus the models schedule orders, and the schedule of orders
determines the demands for production items in time periods. Such models are a generalization
of models with fixed demands.
The models in this work are generally large-scale mixed-integer programming formulations. The main focus of this work is to develop a methodology for generating a variety of schedules relatively quickly. That is, the emphasis is on developing a methodological basis for a "real-time" scheduling system. Thus the emphasis is not necessarily on solving the models to global optimality.

The goal of this work is to attempt to bridge the gap between the research literature on production scheduling, which is largely concerned with mathematical optimization, and the methods for scheduling presently used in industry. Material Requirements Planning (MRP) is today's most widely used design methodology in discrete parts manufacturing. MRP, however, is largely an informational technology and not a scheduling methodology. MRP does not address the global interrelationships of production schedules, capacity limitations, and the lead times for production stages actually experienced in the manufacturing environment.

We develop a solution methodology for our models based on mathematical programming decomposition techniques. A variety of decomposition approaches are used, and they exist within a hierarchical framework that is designed to exploit the structure of the models. Indeed, special structure is present in these models, and decomposition is the natural mechanism for exploiting this structure. Moreover, since we are establishing the theoretical foundation for a methodology for real-time scheduling, mathematical programming decomposition offers a number of advantages over attacking the models as monolithic optimization models. This work attempts to use decomposition as a foundation for guiding the search for good schedules in an integrated and cohesive manner. We argue that decomposition can be used as a high-level mechanism that provides useful input to low-level heuristic routines for determining feasible schedules in complex multistage production environments.

There are many issues raised by this proposed methodology that are addressed in this work. For example, what is the appropriate hierarchical framework of decomposition schemes for the proposed models? Once this framework is established, a number of interesting theoretical and practical problems appear when one hierarchically integrates the information provided by the decomposition subproblems. We explore these research questions in depth.

Thesis Supervisor: Dr. Jeremy F. Shapiro

Title: Professor of Operations Research and Management; Codirector of the Operations Research Center
Table of Contents

Abstract ................................................................. 1

Acknowledgements ...................................................... 6

Mathematical Conventions .............................................. 7

Chapter 1 Decision Models for Multistage Production Planning ........................................... 9
  1.1 Introduction .......................................................... 9
  1.2 Optimization Models ............................................... 16
  1.3 The Role of Optimization Models in Multistage Scheduling ....................................... 25
  1.4 Solution Methodology Research ................................... 27

Chapter 2 Decomposition Strategies ...................................... 28
  2.1 Introduction .......................................................... 28
  2.2 Historical Perspectives and Decomposition Options ........................................... 30
  2.3 A Nesting of Decomposition Strategies ...................................... 39
  2.4 Some Alternate Formulations .......................................... 49

Chapter 3 Special Structure Constraints ..................................... 53
  3.1 Introduction .......................................................... 53
  3.2 A Row-Column Sequencing Algorithm .................................... 56
  3.3 Inventory-Production Constraint Types and Special Constraint Sets .......................... 59
  3.4 Special Structure in Inventory-Production Balance Constraints .................................. 61
  3.5 Properties of Totally Dynamic Systems and Matrices ............................................ 72
3.6 An Algorithm for Solving Linear Programs Defined by
   Totally Dynamic Systems ........................................ 78

3.7 Totally Dynamic Systems and the Theory of Leontief Substitution Systems . . . 87

3.8 The Inventory-production-Orders Balance Constraint Set of Model (EMRP) ... 91

3.9 Solving LPs Over General Inventory-Production Balance Constraints .......... 96

3.10 Alternate Methodologies for Solving LPs Over General Inventory-Production
     Balance Constraints ............................................ 112

Chapter 4 Analysis and Reintegration of the Benders Subproblem ................... 119

   4.1 Introduction ................................................... 119

   4.2 Benders Subproblem Formulations ................................ 121

   4.3 Alternate Subproblem Formulations ................................ 127

   4.4 Analysis of Subproblem Duals .................................. 140

   4.5 A Revised Feasibility Algorithm ................................ 149

   4.6 Cuts for the Benders Master Problem ............................ 152

   4.7 Equi-Cost Dual Solutions Along Rays and Strong Benders Cuts .............. 164

Chapter 5 Implementation Issues ........................................ 175

   5.1 Introduction ................................................... 175

   5.2 A Numerical Example ........................................... 177

   5.3 A Simple Data Structure ....................................... 181

   5.4 First and Last Constraints .................................... 184

   5.5 Data Structures ............................................... 187

   5.6 Computer Algorithms .......................................... 194

Chapter 6 Extensions of the Research ........................................ 208

   6.1 Model Reformulations ........................................... 208

4
Acknowledgements

Sir Isaac Newton once said, "If I have seen farther than you, it is because I have stood on the shoulders of giants." In reference to this work, I feel that if I have seen anything at all, it is due in large part to some individual of grand stature.

Jerry Shapiro has been, at various times, my boss, my friend, and my mentor. His significant research contributions to mathematical programming, his writings on the subject, and his overall approach to optimization theory are among my main inspirations for devoting as much energy as I have to this field. I thank him for all of this, as well as for serving as my thesis advisor during this dissertation.

I want to thank the Charles Stark Draper Laboratory of Cambridge, Massachusetts for generously providing me with financial support throughout the time I worked on this research. In particular, Cynthia Whitney has been a great help as my advisor at Draper Labs and as a member of my thesis committee. I also thank the other members of my committee, Steve Graves and Charlie Fine, for taking the time to help me along with this task. Jean-Paul Calamaro has been a great colleague in implementing some of the ideas in this work on computers. It has been an adventure I've really enjoyed.

The M.I.T. Operations Research Center has provided a wonderful setting for my graduate education, and I'm appreciative of its warm and stimulating atmosphere. It has been a rewarding experience being at the ORC: a lot of hard work, but also a lot of fun.

The Xerox Corporation has provided M.I.T. with a number of their incredible 8010 Workstations. I spent an unbelievable number of hours composing this document with the aid of these versatile, and patient, machines. The quality of the product (despite my typing) speaks for itself.

I thank the Cambridge Zen Center family for supporting me at home throughout this work with their strength and compassion. My parents, Ann and Stu Chapman, have helped me in so many ways, many of which I think only they can understand, that words seem almost inappropriate.

Lastly, Marcia Ryder has been the source of the joy in my life during this arduous process, as well as an expert professional editor who is largely responsible for any semblance of coherence in this document. To her I dedicate this thesis with much love.
Mathematical Conventions

A matrix of size \( m \times n \), where both \( m \) and \( n \) can be greater than 1, is represented by a capital boldface letter (e.g., \( \mathbf{A} \)). A vector (i.e., an \( m \times n \) matrix where either \( m \) or \( n \) must equal 1) is represented by small boldface letters (e.g., \( \mathbf{x} \)). We do not define explicitly a vector as a row or column vector, and we use it in both senses in matrix multiplication. The dimensions of the matrix and the nature of the matrix multiplication will usually determine unambiguously whether the vector is a row or column. Hence vector transposes are not explicitly indicated.

The \( i \)th component of vector \( \mathbf{x} \) is represented by \( X_i \), whereas \( X'_i \) is the \( i \)th component of vector \( \mathbf{y} \). For vector \( \mathbf{x} \), the notation \( \mathbf{x} \geq \mathbf{0} \) (\( > \mathbf{0} \)) means that every component of \( \mathbf{x} \) is nonnegative (positive). The notation \( (\mathbf{x}, \mathbf{y}, \mathbf{z})' \) has the same meaning as \( (\mathbf{x}' \mathbf{y}' \mathbf{z}') \). That is, \( (\mathbf{x}, \mathbf{y}, \mathbf{z})' \) is the vector constructed from the concatenation of the vectors \( \mathbf{x}', \mathbf{y}', \) and \( \mathbf{z}' \). The notation \( \mathbf{1} \) represents a vector where all components are equal to 1, whereas \( \mathbf{0} \) is a vector with every component equal to 0.

Let \( \mathbb{R}^m \) be the set of \( m \)-dimensional vectors of real numbers. Consider the generic optimization problem

\[
\begin{align*}
\text{min } & f(\mathbf{x}) \\
\text{s.t. } & \mathbf{x} \in X
\end{align*}
\]

where \( f \) is a function from \( \mathbb{R}^m \) to \( \mathbb{R}^1 \), and \( X \) is a subset of \( \mathbb{R}^m \). (We assume the infimum of \( f \) over \( X \) is attained.) We represent optimization problems as follows:

\[
\begin{align*}
u(P) &= \text{min } f(\mathbf{x}) \quad (P) \\
\text{s.t. } & \mathbf{x} \in X
\end{align*}
\]
where \((P)\) is the name of the problem, and \(v(P)\) is the optimal objective function value of problem \((P)\). We often refer to optimization problems that are linear programming problems as LPs.

The empty set is indicated by \(\emptyset\). The notation \(G^c\) may be used to indicate the complement of set \(G\) relative to some defined universal set. It is also used to indicate the convex hull of set \(G\). The context will make the meaning clear. The notation \(G - W\) is used to represent the difference between the sets \(G\) and \(W\). (That is, all elements of \(G\) that are not elements of \(W\).) Further conventions will be explained as they arise.
Chapter 1  Decision Models for Multistage Production Planning

1.1 Introduction

Multistage production planning in discrete parts manufacturing environments is an important and generally difficult problem. Material Requirements Planning (MRP) is today's most widely used design methodology in industry for such problems. The advent of large-scale, high-speed computers combined with the technique of "parts explosion" has given rise to a wide spectrum of such systems with which schedulers interact in various ways. These systems gained rapidly in popularity, especially after APICS (the American Production and Inventory Control Society) began strongly advocating their use in the first half of the 1970s. (See, for example, Berry, Vollman, and Whybark (1979) Orlicky (1973, 1975), Plossl and Wight (1971), and Wight (1981).) The sophistication of MRP systems has steadily increased over the last decade, allowing schedulers greater freedom and effectiveness in planning for future requirements.

An integral element of multistage production planning is the product structure, or bill of materials. We begin this work by establishing the terminology and rules concerning product structures that will be used throughout this dissertation. We refer the reader to Figure 1-1 for an example of a typical product structure. Each box with a number refers to an item to be manufactured, and the number in the box is the index of the item. We assume there is always a finite number of items in the product structure. Every item is associated with exactly one level of the product structure and the levels are indexed from 0 to \( L-1 \) where \( L \) is the number of levels. The assignment of items to levels, and the means by which levels are established will be discussed shortly.

In multistage production, the manufacturing of an item may require certain other manufactured items as subassemblies. Each of these subassemblies may themselves require
Figure 1-1: A Typical Product Structure

still other manufactured items as subassemblies, and so forth. If an item $i$ requires another manufactured item $j$ as a subassembly, and $j$ is not always found as a subassembly in an item $k$ that is a subassembly of $i$, then we say that $j$ is an immediate predecessor of $i$ and $i$ is an immediate successor of $j$. If, for distinct items $i$ and $j$, there exists a sequence of items beginning
with $i$ and ending with $j$ such that each item in the sequence (except the last) is an immediate successor of the item that follows it, then we say that $j$ is a predecessor of $i$ and $i$ is a successor of $j$. We call such a sequence that ends in $j$ a successor-generating sequence of $j$. We do not allow an item to be a predecessor or successor of itself, and hence no item in a successor generating sequence is ever repeated.

The level association of an item $i$ is accomplished by assigning each item to the level whose index equals the length of the longest successor-generating sequence of $i$ minus 1. (An item with no successor-generating sequence is placed on level 0.) The depth of an item is the index of the level to which it is assigned. From our assumption of a finite number of items in the product structure and the stipulation that an item cannot be a predecessor or successor of itself, it can be shown that there is always some item on each level $l \in \{0, 1, \ldots, L-1\}$ where $L$ is the length of the longest successor generating sequence over all items in the product structure. Moreover, item $i$ is a successor (predecessor) of item $j$ only if $i$ is on a lower (higher) level than $j$. Level 0 items (i.e., items with no successors) are often referred to as finished goods. Items with no predecessors are often referred to as raw materials. If item $j$ is an immediate predecessor of item $i$, then there is some positive, integral number of finished (i.e., manufactured) $j$s required to be on hand when the manufacturing of any item $i$ begins. When we move from higher indexed levels to lower indexed levels, we say that we are moving upstream in the product structure. (The opposite direction of movement is called downstream.)

We put no further restrictions on the product structure. Hence a wide variety of relationships among items is allowed. For example, multiple immediate successors and predecessors are possible for all items, and distinct items on the same or different levels may share common predecessor items in a number of ways. As we will see in Chapter 2, these possibilities constitute a relaxation of many of the product structure restrictions considered in the literature on multistage lot sizing.

We assume there are a finite number of work stations or machines used to manufacture all items in the product structure. We refer to these manufacturing centers as facilities. The
process of manufacturing each item requires the utilization of some (perhaps more than one) facility. A facility may be used for manufacturing one or more items. We are concerned with production environments in which facilities can potentially become constrained (that is, overloaded with work), thereby producing bottlenecks in the process flow. It becomes clear when one studies multistage, discrete parts manufacturing environments that production schedules, facility utilization, and production item manufacturing lead times can be highly interdependent: Schedules determine facility utilization, and, to the extent that facilities become capacity constrained, work-in-process backs up behind the constrained facilities. Consequently, manufacturing lead times of production items awaiting service on bottlenecked facilities increase.

While traditional MRP totally ignored capacity considerations, more modern MRP systems (often referred to as MRP II systems) attempt to deal with capacity limitations in a "closed loop" fashion, as illustrated in Figure 1-2. (See Wight (1981) for details.) Basically, the MRP II strategy is as follows: After devising an initial master production schedule for finished goods by "rough-cut" capacity planning, a detailed production schedule for all items in the product structure is calculated by MRP from the master schedule, and the effect of this schedule on facilities is determined by a capacity requirements planning (CRP) phase. The CRP phase yields a load summary of facility utilization generated by the MRP schedule. If facilities are overconstrained, the master schedule is revised by the scheduler in the hope that constraint problems will be relieved. MRP again creates a detailed schedule from the revised master schedule. Again the effect of this new schedule on facilities is determined by CRP. This loop process continues until a detailed schedule is found which satisfactorily stays within capacity boundaries.

This "hunt and peck" strategy is MRP's way of attempting to deal with the interdependency of schedules and capacity utilization. It also reveals MRP's inability to produce a schedule that is guaranteed not to violate capacity limitations. Moreover, the MRP literature does not make clear how one should adjust the master production schedule to alleviate capacity violations.
Figure 1-2: "CLOSED LOOP" MRP Planning and Scheduling Process

Given the fact that the production environment often incorporates complex relationships
among production items, it is not at all straightforward in general to decide how master schedule adjustments should be made within the MRP II framework. (See Berry, Vollman, and Whybark (1979).)

MRP requires manufacturing lead times as input. We have observed, however, that lead times are, in part, a function of the production schedule and facility capacity availability. That is, to the extent that the current schedule violates capacity limitations, lead times will grow accordingly as bottlenecking occurs. The quality of the estimates of lead times directly affects the quality of schedules: artificially inflated lead times cause poor facility utilization and excess work-in-process inventories. Under-guessing lead times may lead to infeasible schedules. Moreover, if the lead time of an item $i$ is underestimated, a production bottleneck may occur on a facility producing the item, if the facility is incapable of producing item $i$ as quickly as the schedule calls for. This in turn can create a buildup of work-in-process inventory of the predecessors of $i$.

In addition to a master production schedule, MRP requires lot-sizing rules as input to every stage of the system. Multistage lot sizing is a complex subject and clearly affects schedule quality through its effect on capacity utilization, setup requirements, and cycle stock. Lot sizing is done myopically in MRP; that is, it is done a level at a time starting at the finished goods level and working downstream. When considering a given level, lot sizing for each item of the level is carried out according to some rule. This rule may be a local optimization procedure, although in practice it is often just a relatively simple algorithm. Hence the lower level items must conform to the decisions handed down from above, and this can lead to poor schedules. The issue of lot sizing touches upon an important related issue – namely, that MRP is not an optimization system. As we mentioned, the only place MRP possibly considers avoidable production costs is in the rules for establishing item-by-item lot sizes. But again, the fact that these issues are dealt with locally and sequentially in MRP means that there is no guarantee that implementing these rules will result in acceptable production strategies, especially in the presence of capacitated facilities.
We suggest that mathematical programming models are a potential means for alleviating the aforementioned shortcomings of MRP systems. In addition to being able to schedule production while simultaneously staying within capacity constraints, such models can also, in theory, minimize avoidable production costs. From a practical standpoint, we will argue that the employment of some of the sophisticated techniques of mathematical programming can be used in the multistage manufacturing environment for establishing a solution methodology that represents a marked improvement over current approaches for finding good, cost conscious schedules in acceptable amounts of time. Research has recently been published (e.g., Billington, McClain, and Thomas (1983), Steinberg and Napier (1980)) advocating the use of mathematical programming for analyzing decision alternatives to MRP. This dissertation presents research on several specific mathematical programming models for requirements planning with capacitated facilities. The intent of these models is to address the interrelationships of schedules, capacity planning, lead times while minimizing avoidable production and inventory costs.
1.2 Optimization Models

Billington, McClain, and Thomas (1983) have devised a mathematical programming model that we feel succinctly captures many of the major issues discussed in Section 1.1. Their model proposes to find production schedules that stay within capacity limitations and determine actual lead times as a function of the schedule and facility utilization. Furthermore, the objective function of the model is constructed to minimize avoidable costs. The model is as follows:

Indices:

\[ i = 1, \ldots, M \]  
index of finished goods

\[ i = M + 1, \ldots, N \]  
index of intermediate products

\[ t = 1, \ldots, T \]  
index of planning periods

\[ k = 1, \ldots, K \]  
index of facilities

Parameters:

\[ h_i = \text{inventory holding cost (} \$ \text{ per unit of item} \ i \) \]

\[ cs_i = \text{setup cost (} \$ \text{ per setup of item} \ i \) \]

\[ co_{kt} = \text{overtime cost (} \$ \text{ per unit of capacity in period} \ t \text{ at facility} \ k \) \]

\[ L_i = \text{minimum lead time for item} \ i \]

\[ y_i = \text{yield of item} \ i \text{ (fraction)} \]

\[ a_{ij} = \text{number of units of item} \ i \text{ required for the production of one unit of item} \ j \]

\[ d_{it} = \text{demand for item} \ i \text{ in period} \ t \]

\[ b_{ik} = \text{capacity utilization rate of item} \ i \text{ at facility} \ k \text{ (capacity units per unit)} \]

\[ s_{ik} = \text{setup utilization of facility capacity} \ k \text{ by item} \ i \text{ (capacity units)} \]

\[ CAP_{kt} = \text{capacity of facility} \ k \text{ at time} \ t \text{ (units of capacity)} \]

\[ q_{it} = \text{upper bound on the production of item} \ i \text{ that can be initialized in period} \ t \]
Variables:

\( I_{it} \) = inventory of item \( i \) at the end of period \( t \)

\( X_{it} = 1 \) if item \( i \) made in period \( t \), 0 otherwise

\( O_{kt} \) = overtime capacity at facility \( k \) in period \( t \)

\( P_{it} \) = production of item \( i \) initiated in period \( t \)

Model:

\[
\text{minimize} \quad \sum_{i=1}^{N} \sum_{t=1}^{T} (h_{i}I_{it} + c_{i}X_{it}) + \sum_{k=1}^{K} \sum_{t=1}^{T} (c_{o_{kt}}O_{kt}) \quad (1-1) \quad \text{(BMRP)}
\]

\[
\text{s.t.} \quad I_{i,t-1} + y_{i}P_{i,t-L_{i}} - I_{it} - \sum_{j=1}^{N} a_{ij}F_{jt} = d_{it} \quad i=1,...,N \quad t=1,...,T \quad (1-2)
\]

\[
\sum_{i=1}^{n} (b_{ik}P_{it} + s_{ik}X_{it}) - O_{kt} \leq CAP_{kt} \quad k=1,...,K \quad t=1,...,T \quad (1-3)
\]

\[
P_{it} - q_{it}X_{it} \leq 0 \quad i=1,...,N \quad t=1,...,T \quad (1-4)
\]

\[
I_{it} P_{it} O_{kt} \geq 0 \quad X_{it} \in \{0,1\} \quad (1-5).
\]

BMRP stands for basic MRP. The model does not deal with all of the complexities inherent in devising detailed facility schedules in multistage discrete parts manufacturing. However, we feel the model captures enough elements of the issues we have raised to warrant our investigating it seriously as a prototype for future alternatives to MRP for multistage production planning. The objective function (1-1) in (BMRP) reflects avoidable costs to be minimized. The equation set (1-2) represents the inventory-production balance constraints for
each item in each time period. Note the existence of internally generated, dependent demand $\Sigma a_{ij} P_{jt}$ as well as the independent demand $d_{it}$ in each balance constraint. The parameters $a_{ij}$ will be nonzero if and only if item $j$ is an immediate successor of item $i$ in the product structure. In general, independent demand may exist for any item in the product structure. In many situations, there will be independent demand for finished goods only. The relations (1-3) are the capacity constraints on facilities over the planning horizon. Note that facility capacity is used up by setups as well as actual production. Relations (1-4) ensure no positive production can take place without incurring a setup, and the $q_{it}$s serve as upper bounds on the corresponding production variables.

The parameters $L_i$ represent minimum item lead times—that is, the minimum number of elapsed time periods from production initialization to completion of an item. Hence the parameters are restricted to be some nonnegative integer. Such a measure is independent of facility restrictions. For example, a positive minimum lead time for a component may be due in part to the time required for paint to dry before it can be used in successor components or for transportation of components between facilities. Observe the model assumption that $P_{it}$ represents the production of $i$ initialized in period $t$ and that this production becomes available for use in period $t + L_i$. The utilization of facility capacity incurred by the manufacturing of any item is assumed to take place in the period when production is initialized. The model implicitly computes actual lead times (that is, lead times actually experienced in the manufacturing process because of bottlenecked facilities) by scheduling the production required to meet a specific demand during, or before, the latest possible period for meeting the demand. In the "before" case, the finished production is carried as inventory until it is needed.

The model allows the purchase of "overtime" capacity in the facility-periods. Whereas overtime is indicated as unbounded in (BMRP), this is clearly an unrealistic assumption in many manufacturing environments. Throughout this work, we will indicate when our analysis of the mathematical programming models differs in the presence of bounded overtime, and we will outline how results vary with bounded overtime in such situations.
Note the constraints of (1-2) for any item \( i \) in any period \( t \) where \( 1 \leq t \leq L_i \). For these constraints, \( t - L_i \leq 0 \), and hence the production of \( i \) that becomes available during period \( t \) (\( y_{i,t} P_{i,t-L_i} \)) was initialized before the planning horizon (i.e., periods 1 through \( T \)) begins. Thus we think of the quantity of this production as being outside of the control of the optimization problem. Therefore the entities \( P_{i,t-L_i} \forall i,t \) such that \( 1 \leq t \leq L_i \) are not variables whose values are to be determined by the mathematical program. They are, in fact, known quantities that become available through the early stages of the planning horizon. We call these quantities *holdover production*. Also, the entities \( I_{i0} \forall i \) are not variables. They represent the known inventories of all items in the product structure at the beginning of the planning horizon and are known as *initial inventories*.

We can, then, bring these known quantities to the right hand side of the constraint set by subtracting them from the exogenous demands, since they represent, if you will, negative demands. We define \( r \) to be the vector of *net demand* for model (BMRP) where

\[
\begin{align*}
    r_{i1} &= d_{i1} - I_{i0} - y_{i} P_{i,1-L_i} \forall i \\
    r_{it} &= d_{it} - y_{i} P_{i,t-L_i} \forall i,t : 2 \leq t \leq L_i \\
    r_{it} &= d_{it} \text{ otherwise.}
\end{align*}
\]

Naturally, components of \( r \) can be negative, because holdover production and/or initial inventories made available in an early period may exceed the period's demand.

It follows that constraint set (1-2) of (BMRP) can be rewritten as

\[
-I_{i1} - \sum_{j=1}^{N} a_{ij} P_{j1} = r_{i1} \forall i
\]
\[ I_{i,t-1} - I_{i,t} - \sum_{j=1}^{N} a_{ij} P_{jt} = r_{it} \quad \forall i, t : 2 \leq t \leq L_i \]

\[ I_{i,t-1} + y_i P_{i,t-L_i} - I_{i,t} - \sum_{j=1}^{N} a_{ij} P_{jt} = r_{it} \quad \forall i, t : L_i + 1 \leq t \leq T. \]

Now only true production and inventory variables are present in the model, and the right hand side vector of constraint set (1-2) is no longer necessarily nonnegative.

In most discrete parts manufacturing environments, schedulers must attempt to meet specific orders placed by customers. Often these orders have a promised delivery date associated with them, and the scheduler is faced with the task of trying to satisfy the orders while staying within the limitations of production capacity. Whereas models such as (BMRP) stipulate that all demands must be met on time, in reality schedulers must juggle conflicting demands and push customer orders around to come up with feasible schedules. This may mean shipping orders to customers at less than ideal times or even having to push certain orders out of the planning horizon.

An enhancement to (BMRP) that models such an environment is the following:

**Additional indices:**

\( Z(i) \) = the number of orders for item \( i \)

\( z \) = index of an order

**Additional parameters:**

\( d_{iz} \) = the demand for item \( i \) in the \( z \)-th order for \( i \)

\( q_{ist} \) = the penalty for shipping the \( z \)-th order for item \( i \) in period \( t \)

**Additional variables:**
$W_{izt} =$ the fraction of order $z$ for item $i$ shipped in period $t$
Model:

\[
\text{minimize} \sum_{i=1}^{N} \sum_{t=1}^{T} (h_{it} + cs_{it} X_{it}) + \sum_{k=1}^{K} \sum_{t=1}^{T} (co_{kt} O_{kt}) + \sum_{i=1}^{N} \sum_{z=1}^{Z(i)} \sum_{t=1}^{T+1} q_{itz} W_{itz} [1 - 6] \quad (EMRP)
\]

\[
\text{s.t. } I_{it} - L_{it} - I_{it} - \sum_{j=1}^{N} a_{ij} P_{jt} = \sum_{z=1}^{Z(i)} d_{iz} W_{itz} \quad i=1,...,N \quad t=1,...,T \quad (1 - 7)
\]

\[
\sum_{i=1}^{n} (b_{ik} P_{it} + s_{ik} X_{it}) - O_{kt} \leq \text{CAP}_{kt} \quad k=1,...,K \quad t=1,...,T \quad (1 - 8)
\]

\[
P_{it} - q_{itz} X_{itz} \leq 0 \quad i=1,...,N \quad t=1,...,T \quad (1 - 9)
\]

\[
\sum_{t=1}^{T+1} W_{itz} = 1 \quad i=1,...,N \quad z \in Z(i) \quad (1 - 10)
\]

\[
l_{itz} P_{itz} O_{kt} W_{itz} \geq 0, \quad X_{itz} \in \{0,1\} \quad (1 - 11)
\]

In this extended MRP decision model, the order variables \(W_{itz}\) can be constrained to be binary if customers will not take partial shipment of an order. The inclusion of a \(W_{itz,T+1}\) variable in a planning horizon of \(T\) periods reflects the fact that orders may have to be pushed out of the planning horizon, at least in part, to obtain a feasible schedule at the price of the corresponding penalty \(q_{itz,T+1}\). The penalties on order shipment periods can represent quantifiable losses incurred by the firm when orders are shipped at less than ideal times. For example, the firm may have contractual agreements with customers guaranteeing price breaks for customers when orders are shipped late. Penalties can also reflect the opportunity loss suffered by the firm when it receives customer payments later than planned because of late order shipments to customers. We allow for situations where no part of a given order can be shipped in certain specified periods. This can be modeled by making penalties inordinately
high in such periods or by specifying a subset of periods in which shipment of each order is allowed. For ease of exposition, we will continue to model the constraint set (1-10) of (EMRP) as it currently exists, with the understanding that, for any order, shipment may be allowed in only a subset of the periods. We will assume, however, that each order has the potential to be left out of the planning horizon, in whole or in part.

Model (EMRP) also has the potential to incorporate initial inventories and holdover production that are not already targeted to satisfy specific customer orders. Hence a nonnegative vector representing these quantities can be subtracted from the right hand side of constraint set (1-7), and again only true production and inventory variables will be present in the model.

Clearly (EMRP) encompasses (BMRP) as a special case. If we bound overtime, thereby making the feasibility of the model more of an issue for (BMRP), we see that (BMRP) may spend a substantial amount of effort determining that the fixed demands $d_{it}$ have created an infeasible problem. The scheduler would then have to adjust the exogenous demands and try again. Feasibility is not an issue with (EMRP), since orders can just be pushed out of the horizon as required. In contrast to (BMRP), (EMRP) would expend its effort juggling orders to get a "good" feasible schedule. This is more a reflection of what a scheduler would have to do manually in attempting to find an acceptable schedule. One can draw parallels between the inventory-production-order balance constraint sets (1-7) and (1-10) and the actions of a scheduler trying to develop a master production schedule in the MRP methodology. Constraint sets (1-7) and (1-10) automate, if you will, the planner's attempts to assign demands to time slots for end-items in order to obtain a workable master production schedule.

We note that either (BMRP) or (EMRP) can be enhanced by adding "make or buy" variables. If the option is available to purchase an item $i$ from an outside vendor, verses manufacturing the item, then the scheduler is faced with deciding when, if ever, to exercise this "buy" option. The enhancement is made by introducing the variables $B_{it}$ to represent the quantity of item $i$, purchased from outside vendors, that is received in period $t$. $B_{it}$ is integrated into the existing
models by adding it to the inventory-production balance constraint for item \( i \) in period \( t \) as follows:

\[
B_{it} + I_{i,t-1} + y_i P_{i,t-L_i} - I_{it} - \sum_{j=1}^{N} a_{ij} P_{jt} = d_{it} \quad \text{for model (BMRP)}.
\]

\[
B_{it} + I_{i,t-1} + y_i P_{i,t-L_i} - I_{it} - \sum_{j=1}^{N} a_{ij} P_{jt} = \sum_{t=1}^{Z} d_{iz} W_{izt} \quad \text{for model (EMRP)}.
\]

We can also allow backlogging of demand in model (BMRP). (Observe that backlogging of demand in (EMRP) is already implicitly present because of the flexibility allowed in scheduling orders.) We introduce \( J_{it} \) as the variable that represents the cumulative backlogged demand (i.e., the cumulative unmet demand) for item \( i \) through the first \( t \) periods of the planning horizon. These variables are integrated into model (BMRP) by adding them to the inventory-production balance constraint set in the following way:

\[
J_{it} + B_{it} + I_{i,t-1} + y_i P_{i,t-L_i} - I_{it} - \sum_{j=1}^{N} a_{ij} P_{jt} - J_{i,t-1} = d_{it} \quad i = 1, ..., N \quad t = 1, ..., T.
\]
1.3 The Role of Optimization Models in Multistage Scheduling

It is apparent that either of our optimization models outlined above are mixed integer programming problems that will grow to enormous size relatively quickly for many real-world problems. Consequently, we must concern ourselves with the issue of such models being impractical to solve to optimality. Indeed, despite MRP's shortcomings as a decision support tool (outlined in Section 1.1), MRP was developed by industrial practitioners because these people need schedules in real time, and MRP is a framework which helps provide answers. Whereas MRP is perhaps not a very sophisticated methodology, it has provided a logical structure to help people who must cope daily with scheduling in environments that are often highly complex and chaotic. MRP operates by making a limited number of planning decisions locally and myopically via relatively simple heuristics and by providing the decision maker with timely and organized information. In contrast, the mathematical programming formulations are, potentially, significantly more sophisticated as methodological bases for decision support systems, but solving these formulations to optimality is, in general, intractable.

There is a wide gap between these two extremes that can be bridged by utilizing the well developed techniques of optimization in the proper manner. If we simply declare that solving the mathematical programming formulations of the multistage production problems to optimality is generally intractable and do no more, we believe an opportunity to bridge this gap and improve the state of the art in decision support for multistage production planning would be lost. Given the limitations of MRP, mathematical programming can certainly be used to obtain better, if not "optimal," schedules – better both from the standpoint of reducing the number of difficult decisions that currently must be input to MRP by relegating them to optimization models, and also better in the sense that schedules can be found which consider costs and consequently save money. Moreover, as we have seen, mathematical programming as a design philosophy has the advantage of capturing the important global interdependencies in
multistage capacitated scheduling. MRP, in contrast, despite all its protagonists and enhancements, is still basically only an information technology which provides little insight concerning lead times and capacity constraints. Hence, even as a system for strictly obtaining feasible schedules, mathematical programming is arguably a superior high-level methodology.

As we consider implementing these models, two major topics arise: The first is reducing model size. Along these lines, recent research efforts include product structure compression, as outlined by Billington, McClain, and Thomas (1983). Also of current interest is problem aggregation (e.g., see Kasper (1983) and Nuttle (1981)) and the model reformulations that result. Research questions that arise when considering aggregation include: How should aggregation be applied? What types of models result from aggregation? Can aggregation be used judiciously to both reduce the complexity of the model and alleviate certain model restrictions?

The second major topic is solution methodology. Whatever model simplifications have been instituted — via compression, aggregation, or other means — solution strategy is clearly important. In particular, the methodology should be goal oriented. For example, if we are going to forgo seeking global optimality and attempt to generate cost-conscious, capacity-feasible solutions with an acceptable expenditure of computational effort, it is reasonable to expect that our solution methodology will have to be designed with this end in sight.
1.4 Solution Methodology Research

This dissertation is largely devoted to the development of the mathematical properties of the models (BMRP) and (EMRP) and solution methodologies based on these properties. We believe mathematical programming decomposition is an important methodological strategy to consider for our models. This belief has two basic foundations. The first is the aforementioned impracticality of attempting to solve these models to optimality. Decomposition can provide useful information as to how well one is doing in relation to theoretical optimality and thus gives criteria for decision rules as to whether or not optimization should be terminated. Decomposition can also be utilized to generate feasible solutions to the models relatively frequently. The second foundation is the potential existence of special structure within subsets of the constraint sets of the models that may be potentially isolated and exploited via decomposition, thereby allowing the problem to be solved more efficiently. In general, decomposition can offer a desirable alternative to attempting to solve our models as monolithic mixed integer programs, especially when there is structure to exploit by decomposition in the models, and we are willing to forgo global optimality.

We are therefore concerned with the issues of: what special structure exists within subsets of these constraints; what properties do they have within the framework of the theory of mathematical programming; what are appropriate decomposition techniques for isolating special structure constraints; what are the algorithms for solving decomposition subproblems efficiently; and finally, how is subproblem information used coherently in the appropriate master problems implied by the decomposition strategies?
Chapter 2 Decomposition Strategies

2.1 Introduction

In the preceding chapter we presented several mathematical programming formulations for obtaining schedules for capacity-constrained multistage production problems. Therein we outlined the potential advantages of employing mathematical programming decomposition as a solution methodology for these models. In Chapter 2 we examine in depth the applicability of decomposition to the models (BMRP) and (EMRP).

There are several different decomposition approaches that potentially can be used. We present our views on which decomposition techniques appear to have the greatest analytic promise – and those which do not appear so promising – given the structure of our models and our methodological goals. These views are formed in part by what has been learned about quantitative production scheduling by other researchers in the field. Thus we will indicate how the problems addressed in this dissertation relate to past and ongoing research in production scheduling, mathematical programming decomposition, and optimization theory in general.

Section 2.2 gives a historical perspective on research in quantitative production scheduling. Again, the point of this exercise is to elucidate how our models relate to problems that have been addressed in the literature. Also, this perspective provides a useful framework for judging the applicability of various decomposition techniques. This section closes by providing the rationale for employing Benders decomposition as a high-level decomposition strategy.

Section 2.3 addresses the implications of using a resource-directive decomposition strategy on model (BMRP). In particular, the resultant Benders subproblem has special structure that can be exploited through Lagrangian relaxation. Section 2.3 also begins our exploration of the implications of using a decomposition methodology in which a price-directive decomposition scheme is nested within a resource-directive scheme. We also discuss this methodology in relation to current research on mathematical programming decomposition and alternate
formulations of mixed-integer programs. Section 2.4 closes Chapter 2 by extending our analysis to important variations of (BMRP) and to the extended model (EMRP).
2.2 Historical Perspectives and Decomposition Options

Although it is not necessarily our intention to solve our production planning models to optimality, we will generally adhere to a methodological framework throughout this work in which optimality can be theoretically guaranteed, given the expenditure of sufficient computational effort. We believe such an approach is appropriate in the context of a dissertation that is largely theoretical. We also feel that this approach is a good beginning strategy, because the methodology of generating multistage production schedules via mathematical programming decomposition is, to this point, still largely undeveloped. Hence our initial efforts in developing this methodology concentrate on the mathematical properties of prototypical models and on algorithms that have interesting theoretical properties including, for the most part, guaranteed convergence to optimal solutions.

Moreover, this framework avoids a potential problem of heuristic procedures. Namely, in an attempt to decompose a large problem into more manageable parts, each of which is then attacked by some heuristic, one may, so to speak, "lose the forest for the trees." That is, given the information on the subproblems produced by the heuristic procedures, it may not be clear, in general, how one should integrate this information and proceed accordingly. There may be little feedback as to where one stands within the framework of global optimality. Also, it may not be clear what the appropriate direction is for obtaining better solutions once a certain plateau of feasibility has been reached. We believe the theory of mathematical optimization can provide coherent and integrated guidance to the search for good solutions. The justification for this belief will become clear as our analysis proceeds.

We will use model (BMRP) primarily when developing properties of our models in this chapter. Note, however, that our analysis basically applies to both of our mathematical programming models. The last section of the chapter, Section 2.4, reemphasizes this point by indicating how the results of this chapter apply to the extended model (EMRP). Also, we will be referring to our models as special examples of lot-sizing problems. By using the term "lot-sizing
problem," we adhere to some rather loosely defined conventions found throughout the literature. Namely, a lot-sizing problem is an optimization problem in which one is trying to establish production lot sizes to meet external demands in some number of contiguous time periods while minimizing avoidable production costs. We assume that initializing a production run in any period implies a production setup, and the cost of the setup is reflected in the objective function. This assumption, combined with the assumption that all remaining avoidable costs are linear, leads to concave objective functions in the optimization problem.

As one reviews past research on quantitative models in the production planning area, it becomes clear that model (BMRP) is a relatively difficult lot-sizing problem. The difficulty is due, in part, to the problem being capacitated. From a computational complexity perspective, Florian, Lenstra, and Rinnooy Kan (1980) show that the capacitated lot-sizing problem with only a single production item is NP-hard. Clearly, (BMRP) is NP-hard as well, because it includes the single item problem as a special case.

Beyond the capacity issue, difficulty is also introduced by the generally unrestricted rules we have adopted that define the product structures considered in this work. (See Section 1.1.) If the facility capacity constraints in (BMRP) are relaxed, one might inquire whether the uncapacitated lot-sizing subproblem of (BMRP) can be solved to optimality in a computationally efficient manner. (When we say "computationally efficient," we are adhering to the usual convention of complexity theory that an efficient algorithm for solving a problem is one whose run time is guaranteed to be bounded by a polynomial in the size of the problem.) As we will see in this chapter, the existence in the research literature of efficient algorithms for finding optimal solutions to uncapacitated lot-sizing problems appears limited to problems that represent relatively severe restrictions of the wide variety of multistage problems found in the real world. In contrast, our rules on allowable product structures are constructed to be liberal enough to encompass this variety. However, to our knowledge, it has not been demonstrated that the general uncapacitated multistage lot-sizing problem is NP-hard. If research effort has been lacking in this area, then it is a theoretically important topic that warrants investigation.
The quantitative analysis of lot-sizing problems constitutes a large literature, and we will make no attempt to present a detailed history of this research area. (Billington, McClain, and Thomas (1983) provide an interesting compilation of references of research in the field.) Instead, we will mention the research that is most relevant to promoting an understanding of why we choose to proceed the way we do in our decomposition strategies.

Wagner and Whitin (1958) establish that the single-item uncapacitated lot-sizing problem, which is perhaps the simplest of the interesting lot-sizing problems, can be solved very efficiently. Moving in a direction of increasing complexity, a serial system is defined to be a product structure with one finished good, one raw material, and all remaining items have exactly one successor and predecessor. Zangwill (1968, 1969) and Love (1972) have developed efficient solution techniques for uncapacitated serial systems.

Many papers have been written on the single-level multi-item capacitated lot-sizing problem. These are problems where every item in the product structure is a finished good, and items potentially share production facilities. Manne (1958) considers a large-scale LP version of the problem. Dzielinski and Gomory (1965), Lasdon and Terjung (1971), and a number of other researchers take advantage of the fact that relaxing the capacity constraints decomposes the problem into independent single-item uncapacitated problems (i.e., "Wagner-Whitin problems"). The variety of methods available for trying to solve the original problem from this Lagrangian relaxation is responsible for the large number of papers generated by this problem.

A multistage product structure in which every item has a unique successor (predecessor) is known as an assembly (arborescent) product structure. The lot-sizing literature reveals that the development of efficient algorithms for uncapacitated multistage lot-sizing problems appears limited to special cases of assembly and arborescent structures. This is not to say, however, that an efficient algorithm is known for generic lot-sizing problems with assembly or arborescent product structures. To our knowledge, this is not the case.

Veinott (1969), for example, presents an efficient algorithm for finding the optimal solution to an uncapacitated lot-sizing problem with an arborescent product structure. The key to this
algorithm is the development of a sufficient condition for an optimal solution (Theorem 4, pp. 276 of Veinott (1969)) that is quite strong and satisfied by at least one feasible solution. This condition leads to a recursive equation ((29) of Veinott (1969)) that defines the set of optimal solutions satisfying this condition. The recursive equation is efficiently solvable by dynamic programming. Unfortunately, Veinott's algorithm is based on an objective function that must satisfy some rather stringent, and perhaps unrealistic, restrictions. (See (25) - (27) of Veinott (1969).) Therefore the algorithm applies to a very limited class of lot-sizing problems with arborescent product structures.

Crowston, Wagner, and Williams (1973) (CWW) present research results on lot-sizing problems with assembly product structures. The problem CWW address is a simplification of the generic lot-sizing problem with assembly product structures, because CWW assume the demand for the unique finished good is constant, and only production policies that can be characterized by constant, repetitive lot sizes for each item in the product structure are considered. Whereas an algorithm (also based on a dynamic programming recursion) for calculating an optimal solution is found, the algorithm is not computationally efficient. That is, the amount of work required grows exponentially with the number of stages, the number of time periods, or both.

Given the lack of results on the existence of an efficient solution for the generic uncapacitated lot-sizing problem with either an assembly or arborescent product structure, it is not surprising that results are lacking for the generic uncapacitated multistage lot-sizing problem as well. When one studies the dynamic programming-based algorithms that have been developed for the case where the product structure is either assembly or arborescent, it becomes clear that the dynamic programming recursions that define the optimal solutions depend directly on the "tree" structure of the product structure. That is, there is a dependence on each item having either a unique successor or predecessor. For example, in the work of CWW, consider the setting of the lot size for any item $i$ of the assembly product structure. For any item $j$ that is an immediate predecessor of $i$, the setting of the lot size of item $i$ is the only lot-sizing
decision that has any direct effect on the lot-sizing decision for item \( j \). This is because \( i \) is the only item that is an immediate successor of \( j \). This fact is crucial in the recursive equation that defines an optimal solution in CWW’s work.

It is also clear that the generic multistage product structure introduces a significant degree of complexity over the assembly or arborescent structure when one is attempting to define optimal solutions by recursive equations. Once multiple predecessors and successors are allowed, the lot-sizing decision for an item \( j \) is now potentially affected by the decisions of all items that are immediate predecessors of \( j \). There can be a large number of these immediate successors, and they can be found on a variety of levels in the product structure. Whereas the development of such recursive equations for the generic multistage problem is possible, the dynamic programming problem defined by these equations would be enormously complicated in general and certainly subject to the “curse of dimensionality” that so often afflicts dynamic programming formulations.

Research on generic uncapacitated multistage lot-sizing problems has centered on heuristics for obtaining feasible schedules. In fact, the MRP methodology falls within this category. The most common form of heuristic is to consider the levels sequentially, starting at the lowest level of the product structure and working downstream. This, as we saw in Chapter 1, is true of the MRP methodology. Lot sizing is performed for each item in the level currently under consideration. As we discussed in Chapter 1, the item-by-item lot sizing may be done by an optimization procedure (such as the Wagner-Whitin algorithm) or may be done by a heuristic. We have seen that such a methodology implies the items on the higher levels of the product structure must live with the decisions made for upstream items. A number of papers have been written on the empirical effects of employing various lot-sizing rules for single-item lot sizing within this sequential heuristic approach. (See, for example, Berry (1972), New (1974), Biggs, Hahn, and Pinto (1980), and Collier (1980).)

Graves (1981) recognizes the inherent complexity of the generic multistage problem and the potential pitfalls of making lot-sizing decisions in a single downstream pass through the
product structure. He therefore develops a multi-pass heuristic for the uncapacitated multistage problem. Computational results are promising for the test cases presented. Whereas Graves reports that the optimal solution was found by the heuristic for many of the test cases considered, the fact that the procedure is a heuristic means there is no theoretical guarantee an optimal solution will be found.

In light of this historical perspective, it may be fair to say that there is a strong possibility that no efficient algorithm exists for the generic uncapacitated multistage lot-sizing problem. Again, it would be helpful if this problem could be established as NP-hard. The lack of any such known algorithm, and the strong possibility of its nonexistence, has a direct influence on how mathematical programming decomposition will be employed on the optimization models presented in Chapter 1.

For example, we could relax the facility capacity constraints (1-3) of (BMRP) in the hope that the resultant Lagrangian subproblem is significantly easier to solve than the entire model. But in light of the historical perspective on lot-sizing problems, we must conclude that this method of decomposing the problem yields a subproblem that we are incapable of solving to optimality efficiently, at least with the current state of the art in solving uncapacitated multistage lot-sizing problems. In other words, the primary rationale for employing Lagrangian relaxation is to isolate subproblems that are considerably easier to solve. But in the context of model (BMRP), we are faced with a Lagrangian subproblem for which no efficient algorithm is known for solving it to optimality. We are, in effect, left with a large mixed-integer program in the Lagrangian subproblem.

Moreover, the prospect of isolating such Lagrangian subproblems and attacking them by heuristics that may not solve them to optimality is not very appealing, at least from a theoretical perspective. First of all, if the heuristic cannot guarantee to solve the Lagrangian subproblem to optimality, then the best solution to the subproblem found by the heuristic is not necessarily a lower bound on the optimal global objective function value. Moreover, if one studies the various well known methods for solving the dual problem implied by a Lagrangian
relaxation, it is apparent that the convergence of these methods depends on the Lagrangian subproblems being solved to optimality.

For example, subgradient optimization has proved to be one of the most practical schemes for attacking dual problems implied by Lagrangian relaxation. (See Danyanov (1968) and Polyak (1967, 1969).) Let \( L(\gamma) \) be the optimal objective function value of the Lagrangian subproblem given the dual price vector \( \gamma \geq 0 \) on the capacity constraints of model (BMRP). \( L \) as a function of \( \gamma \) is concave (see Rockafellar (1970)), and the subgradient optimization algorithm requires a subgradient of \( L \) as a direction of movement in calculating new dual prices on the facility capacity constraints. But to calculate a subgradient of \( L \) at \( \gamma \), it is necessary to know at least one optimal solution to the Lagrangian subproblem with dual price vector \( \gamma \). (See Grinold (1970).) If one constructs a direction of movement from a heuristic-generated solution to the Lagrangian subproblem that is not necessarily optimal in the subproblem, what can one say about the validity of the direction of movement used to calculate new dual prices and the convergence properties of the subgradient optimization algorithm? (We note that research (Bertsekas and Mitter (1973)) has extended the subgradient optimization algorithm to search for solutions to such dual problems that are within \( \epsilon > 0 \) of optimality. It can be shown that the Lagrangian subproblems need only be solved to within \( \epsilon \) of optimality to generate a valid direction of movement in this extended algorithm. However, the available heuristics for the uncapacitated multistage lot-sizing problem are not \( \epsilon \)-optimal. That is, there is no guarantee that the heuristics will find a solution that is within \( \epsilon \) of optimality for reasonably small values of \( \epsilon > 0 \).)

One might consider pricing out the fixed charge constraints (1-4) as well as the facility capacity constraints in constructing the Lagrangian subproblem. The binary setup variables can then be set by inspection in the subproblem, since the only constraints now affecting them are those requiring them to be binary. The remaining constraints of the subproblem constitutes an LP over the inventory-production balance constraint set. Chapter 3 of this work is devoted to exploring the mathematical structure of this constraint set and developing special algorithms.
for solving LPs with feasible regions defined by this constraint set. Hence the Lagrangian subproblem is relatively easy to solve to optimality. However, we reserve a fair amount of skepticism for an approach in which so many constraints have to be priced out to obtain a subproblem that is easy to solve. The cost for eliminating so many constraints may be a dual solution with a large duality gap and/or dual solutions bearing little resemblance to good primal solutions.

For all of the above reasons, we will consider Benders' decomposition as a decomposition approach to be applied at the highest level to our production scheduling models. (See Benders (1962).) Some successful implementations of Benders' decomposition for solving relatively large mixed-integer programming problems include Geoffrion and Graves (1974) and Bienstock and Shapiro (1984). In our production planning models, the binary setup variables, and only these variables, will be set in the Benders master problem. It is well known that the sequence of Benders master problem solutions generated by the algorithm provide monotonically nondecreasing lower bounds on the optimal solution to the global optimization problem. In addition, the optimal solution to a feasible Benders subproblem, in conjunction with the corresponding master problem variables, constitute a feasible solution to the global optimization problem. Hence one can expect feasible solutions to be generated quite frequently.

This property is particularly useful, given our emphasis on finding acceptable feasible solutions in reasonable amounts of time. Moreover, this property is not guaranteed when one employs Lagrangian relaxation as the highest-level decomposition scheme, nor is it guaranteed when one employs conventional branch and bound to our mixed-integer programming models.

Although it is outside of the immediate scope of this dissertation, we feel it is worth mentioning that Benders’ decomposition may prove to be an appropriate optimization mechanism in research that is now underway on applied production scheduling at MIT. (See Calamaro and Chapman (1985).) This research attempts to identify the proper synthesis of the expertise that a scheduler has developed through experience in scheduling in a particular manufacturing environment and the power of modern mathematical programming tools.
One possible scenario is initially to allow the scheduler to use his experience in making decisions about when and where production runs should be initiated. In essence, then, the scheduler is taking over the role of the Benders master problem, and the implied subproblems can be solved by mathematical programming. Human insight concerning the specific problem at hand may lead to intelligent early setup decisions that are quite good and that may be beyond the capability of a mathematical programming master problem to make, in light of the relatively few algorithm iterations that would have taken place in the early stages of scheduling. These human decisions may be the basis of very useful master problem cuts, and hence they should be retained.

Once this human-powered procedure has reached a certain state of maturity, there may be enough good information generated on setups to allow mathematical programming to take over the role of making setup decisions. That is, once enough useful master problem cuts have been generated via human insight to the problem at hand, the Benders master problem may be capable of generating useful setup decisions on its own. Thus the framework would be in place for potentially generating schedules that are better than those obtained when setups decisions were made by the scheduler.
2.3 A Nesting of Decomposition Strategies

We now pursue the implications of employing Benders' decomposition as a high-level decomposition strategy for model (BMRP). It will be advantageous to recast model (BMRP) in matrix-vector notation. We therefore define

Variables:

\[ p = \text{production vector} \]
\[ v = \text{inventory vector} \]
\[ o = \text{overtime vector} \]
\[ x = \text{setup vector} \]

Objective function coefficient vectors:

\[ h = \text{the inventory holding cost vector} \]
\[ f = \text{the setup cost vector} \]
\[ c = \text{overtime cost vector} \]

Right hand side vectors:

\[ d = \text{demand vector for the I-P balance constraint set} \]
\[ a = \text{capacity vector for the facility capacity constraint set} \]

Constraint matrices:

\[ A = \text{matrix for production vector} \ p \ \text{in the I-P balance constraint set} \]
\[ D = \text{matrix for inventory vector} \ v \ \text{in the I-P balance constraint set} \]
\[ B = \text{matrix for production vector} \ p \ \text{in the facility capacity constraint set} \]
\[ S = \text{matrix for setup vector} \ x \ \text{in the facility capacity constraint set} \]
\[ Q = \text{diagonal matrix containing the components} \ q_{it} \ \text{along the diagonal.} \]

Given these definitions, our basic model can be formulated as follows:
\[ \nu(\text{BMRP}) = \min \{ hv + fx + co \} \]  

\[ \text{s.t. } Ap + Dv = d \]  

\[ Bp + Sx - o \leq a \]  

\[ p - Qx \leq 0 \]  

\[ p,v,o \geq 0 \]  

\[ X_{it} \in \{0,1\} \]  

Within the context of problem (BMRP), presetting the setups in the Benders master problem yields a subproblem that is strictly a linear program: namely,

\[ \nu(\text{SP}x) = fx + \min \{ hv + co \} \]  

\[ \text{s.t. } Ap + Dv = d \]  

\[ Bp - o \leq a - Sx \]  

\[ p \leq Qx \]  

\[ p,v,o \geq 0 \]  

where (SPx) stands for the subproblem of Benders' algorithm formed by presetting vector x.

Since (SPx) is a linear program, it can be solved by the simplex algorithm. However, we have noted that the existence of special structure in the inventory-production balance constraint set (2-8) is the subject of Chapter 3 of this dissertation. Therein we show that, in general, the majority of these constraints constitute a type of constraint system such that LPs over feasible regions defined by these systems can be solved extremely quickly. These systems of contraints are called totally dynamic systems, for reasons made clear in Chapter 3. For mathematical simplicity, we assume this special structure is inherent in all of constraint set (2-8).
Because of (2-8)'s special structure, it is natural to consider Lagrangian relaxation as a mechanism for pricing out constraint sets (2-9) and (2-10). Thus an LP is produced where a totally dynamic system defines the feasible region. We define $\gamma$ and $\lambda$ as the dual vectors for constraint sets (2-9) and (2-10) respectively. For any dual vector $(\gamma, \lambda)$ with $\gamma, \lambda \geq 0$, we define the Lagrangian problem $(L(\gamma, \lambda; x))$ problem formed from $(SPx)$ to be

$$v(L(\gamma, \lambda; x)) = fx + \gamma[Sx - a] - \lambda Qx + \min \left( (\gamma B + \lambda) p + hv + (c \cdot \gamma) o \right) \tag{2-12}$$

s.t. $$Ap + Dv = d \tag{2-13}$$

$$p, v, o \geq 0 \tag{2-14}.$$ 

The terms $\gamma[Sx - a]$ and $\lambda Qx$ are not part of the optimization for a given vector $(\gamma, \lambda)$ of dual prices. They simply represent the effect of the dual prices on the right hand side vectors of the relaxed capacity and fixed charge constraints respectively. We note that $\gamma$ and $\lambda$ are constrained to be nonnegative in order to assure that $v(L(\gamma, \lambda; x))$ is a lower bound on $v(SPx)$.

The dual problem implied by the Lagrangian $(L(\gamma, \lambda; x))$ is

$$v(d; x) = \sup_{\gamma, \lambda \geq 0} v(L(\gamma, \lambda; x)) \tag{2-15}$$

$$d; x).$$

Observe that the only constraint on overtime vector $o$ in $(L(\gamma, \lambda; x))$ is nonnegativity. If the objective function vector $c \cdot \gamma$ for variable vector $o$ in $(L(\gamma, \lambda; x))$ has a negative component, say $(c \cdot \gamma)_i < 0$, then the objective function can be made arbitrarily small by letting $O_i$ approach $+\infty$, implying $v((L(\gamma, \lambda; x))) = -\infty$. The interpretation of this situation is that if we can buy overtime at a unit cost $c_i$ that is lower than its worth $\gamma_i$, then we should buy all we can. In this particular problem, we can buy an infinite amount, and hence our “cost” can be made infinitely low. This is a rather uninteresting Lagrangian. Clearly then, the only vectors $(\gamma, \lambda)$ worth considering in attempting to solve $(d; x)$ are those with $c \cdot \gamma \geq 0$ or $c \geq \gamma$. Moreover, when $c \geq \gamma$ there is an
optimal solution to \((L(y,A;x))\) with \(o = 0\); that is, an optimal solution exists in which no overtime is purchased.

The convex polytope formed by constraint sets (2-13) and (2-14) without the overtime vector can be considered a convex polyhedron; i.e., a bounded polytope. We briefly indicate why this is so. We mention, while discussing the example of Figure 3-1 in Chapter 3, how final-period inventories of all production items can be constrained to predetermined target values and removed from the optimization problem. This observation applies to all the problems considered in this work, and so final period inventory variables are implicitly left out of all our models.

By direct observation of the inventory-production balance constraints, it is straightforward to show that an unbounded sequence of feasible solutions to (2-13), (2-14) exists if and only if the sequence of aggregate final inventory values derived from this solution sequence is itself unbounded. Hence, if final inventories are finitely constrained, no unbounded sequence of feasible solutions exists, and the polytope resulting from presetting final period inventory variables is bounded.

We define

\[ Y_E = \left\{ (p,v)^i \right\}_{i=1}^L \]

as the extreme points of the convex polyhedron

\[ Y_{IP} = \left\{ (p,v) : Ap + Dv = d : p,v \geq 0 \right\}. \]

In light of our observations, problem \((d;x)\) can be rewritten as
\[ v(d:x) = v(d-2:x) = fx + \max \left[ \gamma[Sx - a] - \lambda Qx + w \right] \]  
(2-16)  \hspace{1cm} (d-2:x)

\[ \text{s.t.} \quad w \leq (\gamma B + \lambda)p^i + hv^i \quad \quad i = 1, \ldots, L \]  
(2-17)

\[ 0 \leq \gamma \leq \epsilon \]  
(2-18)

\[ 0 \leq \lambda \]  
(2-19).

This latest formulation of the dual problem \((d:x)\) is a linear program. We take the dual of \((d-2:x)\) and obtain

\[ v(d:x) = v(d-3:x) = fx + \min \left[ \sum_{i=1}^{L} a_i (hv^i) + co \right] \]  
(2-20)  \hspace{1cm} (d-3:x)

\[ \text{s.t.} \quad \sum_{i=1}^{L} a_i (Bp^i) - o \leq a - Sx \]  
(2-21)

\[ \sum_{i=1}^{L} a_i p^i \leq Qx \]  
(2-22)

\[ \sum_{i=1}^{L} a_i = 1 \]  
(2-23)

\[ \alpha, o \geq 0 \]  
(2-24)

where \(\alpha\) is a vector of length \(L\).

Take note of this last formulation of the dual problem in relation to \((SPx)\). In general, mathematical programming duality theory guarantees \(v(d:x)\) is a lower bound on \(v(SPx)\) with the possibility of a positive duality gap \(v(SPx) - v(d:x)\). However, we can express \(Y_{IP}\) as the convex hull of \(Y_E\). Hence comparing \((d:x)\) to \((SPx)\) shows the two problems to be equivalent, because \(Y_{IP}\) can be expressed as
\[ Y_{lp} = \left\{ (p,v) : (p,v) = \sum_{i=1}^{L} a_i (p,v)^i, \sum_{i=1}^{L} a_i = 1; (p,v)^i \in Y_E, a_i \geq 0, i=1,...,L \right\} \] (2-25).

So \( \nu(SP\mathbf{x}) = \nu(d:\mathbf{x}) \), and no duality gap exists. This result is what one would expect from mathematical duality theory, given the linear objective function of \((SP\mathbf{x})\) and the fact that the feasible region of the Lagrangian subproblem is a convex polyhedron. (See Shapiro (1979).)

The formulation \((d:\mathbf{3:x})\) requires a knowledge of the set of extreme points \(Y_E\) of the inventory-production balance constraints. Naturally, we generally have no knowledge of this set \(a priori\). However, all elements of \(Y_E\) are not needed generally to solve \((d:\mathbf{3:x})\); that is, only those elements whose corresponding \(a_i\) variable must be positive in an optimal solution to \((d:\mathbf{3:x})\) need be known.

Dantzig-Wolfe (D-W) decomposition (Dantzig and Wolfe (1960,1961)) is a natural mechanism for generating only those elements of \(Y_E\) needed to solve \((SP\mathbf{x})\) via formulation \((d:\mathbf{3:x})\). The constraints priced out in the Lagrangian relaxation ((2-9) and (2-10)) form the basis of the D-W master problem, while \(Y_{lp}\) is the feasible region of the D-W subproblem. Formulation \((d:\mathbf{3:x})\) is interpretable as the “full” D-W master problem in which all elements of \(Y_E\) are known. As the D-W algorithm proceeds, the element of \(Y_E\) that has the lowest reduced cost in the current D-W master problem is found by the D-W subproblem by solving an LP over the inventory-production balance constraint set. We have mentioned that Chapter 3 will develop special algorithms for solving such LPs extremely quickly. The D-W algorithm is well known to converge in a finite number of iterations to an optimal solution to \((SP\mathbf{x})\). The D-W algorithm also generates a feasible solution to the global optimization problem at hand ((SP\mathbf{x}) in our context) at every master problem iteration when the global optimization problem is an LP. If the Benders subproblem \((SP\mathbf{x})\) is feasible, the implication of this is that the D-W master problem will generate a feasible solution to \((BMRP)\) at every iteration.

We recall that \((d:\mathbf{2:x})\) is a formulation of the dual problem \((d:\mathbf{x})\) implied by a Lagrangian relaxation of the Benders subproblem \((SP\mathbf{x})\). We have also seen that there is no duality gap.
between this dual problem and \((SPx)\), because \((SPx)\) is an LP. Consequently, any of the known techniques of mathematical programming for solving Lagrangian-derived dual problems can be employed in an attempt to solve \((SPx)\), if one decides that isolating the I-P balance constraint set in a Lagrangian LP subproblem is a strategy worth pursuing. In Chapter 4 we will show that the feasible region \(S\) of \((d-2:x)\) is a valid dual region for generating cuts for the Benders master problem.

Thus we find ourselves with motivation for considering a nesting of decomposition strategies in our pursuit of a decomposition-based methodology for obtaining good feasible schedules to our production problems. Benders decomposition is resource-directive and involves determining when the production setups should take place. We have seen that there are compelling reasons for employing Lagrangian relaxation, which is price-directive, on the resultant Benders subproblem. In Chapter 4 we will show that this nested decomposition scheme can be constructed so that an optimal global solution to \((BMRP)\) is obtained in a finite number of iterations. We will also see that this decomposition scheme can be used in a variety of ways for obtaining good, but not necessarily optimal, production schedules.

Recall that we had serious reservations about relaxing both the facility capacity constraints and the fixed charge constraints in model \((BMRP)\) because of the potentially large duality gap that may result. This duality gap problem is alleviated when these constraints are relaxed in the Benders subproblem, because the subproblem is an LP. One might inquire about the implications of relaxing only the facility capacity constraints in the Benders subproblem. Several interesting issues are raised when we consider performing the price-directive decomposition in more than one way in the Benders subproblem. This topic is addressed at length in Chapter 4 of this work.

We now comment on relatively recent research that is related to our production planning problems and the methodology we are in the process of developing for obtaining solutions to these problems. There are clear surface similarities between the nested decomposition approach discussed in this chapter and Van Roy's (1983) cross decomposition algorithm. Both
algorithms are a hybrid of resource-directive and price-directive decomposition. However, a closer examination reveals that the two algorithms are fundamentally different. Generally speaking, cross decomposition is best employed on optimization problems where there are compelling reasons for attacking the global problem by both resource-directive and price-directive decomposition. That is, the global problem has a structure that can be readily exploited by either type of decomposition.

The analysis in this chapter has shown that our optimization formulations of capacitated multistage production problems are not readily exploitable by price-directive decomposition. We have seen that relaxing only the facility capacity constraints yields a Lagrangian subproblem that appears to be difficult to solve to optimality. In addition, we are skeptical about a price-directive scheme that relaxes the fixed charge constraints as well. This skepticism is due to the fact that this approach relaxes all constraints containing both continuous and binary variables. Hence the Lagrangian subproblem is relatively trivial in relation to the global optimization problem and bears little resemblance to the global problem.

We conclude, then, that cross decomposition may be of limited value as a solution methodology for capacitated multistage production planning problems. The nested decomposition scheme we have outlined is constructed to employ price-directive decomposition on the subproblem that results from first employing resource-directive decomposition. Therefore decomposition is employed sequentially in an ordered (i.e., resource-directive followed by price-directive) fashion. In contrast, cross decomposition proceeds by

1) solving a Lagrangian subproblem of the global problem and using the solution to set the primal master problem variables of the resource-directive decomposition scheme.

2) solving the Benders subproblem of the global problem that results from these preset master problem variables and using the optimal dual variables of the Benders subproblem to set the Lagrangian price vector of the price-directive decomposition scheme:
3) iterating in this fashion between the two subproblems while periodically solving the master problem implied by the two decomposition schemes.

Hence our nested decomposition scheme is fundamentally different from cross decomposition. There is currently a good deal of interest in alternate formulations for mixed-integer programming problems. (See, for example, Van Roy and Wolsey (1984) and Padberg, Van Roy, and Wolsey (1985).) Virtually all commercial mathematical programming computer codes solve mixed-integer programming problems by employing a conventional branch and bound scheme. An implication of this approach is that one would like the LP relaxation of the problem to resemble, as much as possible, the convex hull of the set of feasible solutions to the problem. The alternate formulation research therefore concentrates on trying to find problem formulations that include "strong cuts." That is, one attempts to find inequality constraints that are satisfied by all feasible solutions but which also, in the ideal situation, define facets of the convex hull of the set of feasible solutions.

For example, Barany, Van Roy, and Wolsey (1984) have explicitly identified a set of cuts that defines the convex hull of the set of feasible solutions to the single-item uncapacitated lot-sizing problem. This knowledge is utilized in a cutting-plane procedure for solving the single-level multi-item capacitated lot-sizing problem. As we have seen, there is a quantum leap in difficulty in general when moving from product structures with no predecessor-successor relations between items to generic multistage product structures. There has been, to our knowledge, no published research on extensions of the results on the single-item case to multistage product structures.

The research of Van Roy and Wolsey (1984) and Padberg, Van Roy, and Wolsey (1985) concentrates on reformulating general mixed-integer programming problems, and therefore their results could be applied to mixed-integer programming formulations of multistage
capacitated lot-sizing problems. In fact Van Roy and Wolsey (1984) have tested their approach on uncapacitated assembly-structured lot-sizing problems.

Our approach is fundamentally different, because we are not relying on a conventional branch and bound approach in our solution methodology. There is, however, theoretical (e.g., Magnanti and Wong (1981)) and empirical (e.g., Geoffrion and Graves (1974)) evidence that "tight" LP relaxations of mixed-integer programming formulations are also desirable when employing Benders' decomposition. Hence, for example, we would like the $q_{it}$ parameters that bound the production variables in the fixed charge constraints of our models to be as small as possible without unrealistically restricting the production variables. We will discuss this issue again in Chapter 4.
2.4 Some Alternate Formulations

We finish this chapter by clarifying how the formulations of the previous section change as we deviate from the version of model (BMRP) in which overtime availability is assumed to be unbounded. We also consider how our analysis applies to the extended model (EMRP). We begin by observing how the analysis of the dual problem — implied by the relaxation of the facility capacity and fixed charge constraints of the Benders subproblem of (BMRP) — changes in the presence of finite bounds $u_{kt} \geq 0$ on each of the overtime variables $O_{kt}$. Note that the Lagrangian subproblem is now

$$
\nu(L(\gamma, \lambda; x)) = fx + \gamma(Sx - a) - \lambda Qx + min \{ (\gamma B + \lambda) p + hv + (c - \gamma) o \} \quad (2-26) \quad (L(\gamma, \lambda; x))
$$

s.t. 

$$
Ap + Dv = d \quad (2-27)
$$

$$
p, v \geq 0 \quad (2-28)
$$

$$
0 \leq o \leq u \quad (2-29)
$$

where $\gamma, \lambda \geq 0$.

If $(c - \gamma)_i < 0$, we can no longer drive the objective function value of the Lagrangian to $-\infty$, because $O_i$ is now bounded. Instead, any optimal solution to the Lagrangian will have $O_i = u_i$ when $(c - \gamma)_i < 0$ and $O_i = 0$ when $(c - \gamma)_i > 0$. When $c_i = \gamma_i$, then any feasible value of $O_i$ makes the same contribution (zero) to the objective function value. Since the feasible region of the I-P balance constraint set $Y_{IP}$ has been shown to be bounded, the implication is that $\nu(L(\gamma, \lambda; x))$ is bounded for any $\gamma, \lambda \geq 0$. In light of these observations, the formulation (d-2:x) of the dual problem (d:x) can be written as
\[ v(d; x) = v(d-2; x) = f x + \max \{ y(S x - a) - \lambda Q x - n l + w \} \]  
\[ \text{s.t.} \quad w \leq (y B + \lambda)p^i + h v^i \quad i = 1, \ldots, L \]  
\[ (y - c)_{kt} u_{kt} \leq n_{kt} \quad k = 1, \ldots, K \quad t = 1, \ldots, T \]  
\[ 0 \leq y, \lambda, n \]  
\[ (d-2; x) \]  
\[ (2-30) \]  
\[ (2-31) \]  
\[ (2-32) \]  
\[ (2-33) \]

Observe that \( y \) is no longer explicitly bounded above by \( c \). Also, an optimal solution exists with \( n_{kt} = u_{kt} \max \{ 0, (y - c)_{kt} \} \ \forall \ k, t. \) The dual of this latest formulation of \( (d; x) \) is the same as \( (d-3; x) \) \((2-20) \) through \((2-24)) \) with the exception that the overtime vector \( o \) is now bounded above by \( u \).

One can easily extend the model so that the overtime cost function is a convex, piece-wise linear function, with the availability of overtime being either bounded or unbounded. If \( c^1 \leq c^2 \leq \ldots \leq c^n \) are the overtime cost vectors and \( o^1, o^2, \ldots, o^n \) the corresponding overtime variable vectors in such an objective function, then it clear that there is an optimal solution to the Lagrangian where one sets any overtime variable \( O_{kt} \) to its maximal feasible value when \((o - y)_{kt} \) is negative and zero otherwise \( \forall j, k, t \). The dual problem formulation for this extension that corresponds to \((d-2; x) \) can be constructed similarly.

We now consider how the analysis of this chapter applies to the extended model (EMRP). We let \( w \) be the vector of order variables, \( b \) the vector of cost coefficients for the order variables in the objective function, \( E \) the coefficient matrix of the order variables in the inventory-production-orders balance constraint set \((1-7) \), and \( F \) the coefficient matrix of the order variables in constraint set \((1-10) \) that ensures all orders are accounted for. The Benders subproblem can then be written as

\[ v(S P x) = f x + \min \{ h v + c o + b w \} \]  
\[ \text{s.t.} \quad A p + D v - E w = 0 \]  
\[ F w = 1 \]  
\[ B p - o \leq a - S x \]  
\[ p \leq Q x \]  
\[ (2-34) \]  
\[ (SP x) \]  
\[ (2-35) \]  
\[ (2-36) \]  
\[ (2-37) \]  
\[ (2-38) \]
\[ p, v, o, w \geq 0 \] (2-39).

We can carry through the implications of the relaxing of the facility capacity constraints (2-37) and the fixed charge constraints (2-38) exactly as before. The counterpart of formulation (d-3:x) for (EMRP) is

\[
u(Sp) = \nu(d-4:x) = fx + \min \left[ \sum_{i=1}^{L} \alpha_i (hw_i + bw_i) + co \right] \quad (d-4:x)
\]

s.t. \[ \sum_{i=1}^{L} \alpha_i (Bp_i) - o \leq a - Sx \]

\[ \sum_{i=1}^{L} \alpha_i p_i \leq Qx \]

\[ \sum_{i=1}^{L} \alpha_i = 1 \]

\[ \alpha_i \geq 0, i = 1, \ldots, L ; \quad o \geq 0 \]

where

\[ Y_E \equiv \left\{ (p,v,w)^{t} \right\}_{t=1}^{L} \]

is the set of extreme points of the region

\[ Y_{IP} \equiv \left\{ (p,v,w) : Ap + Dv - Ew = 0 ; Fw = 1 ; p,v,w \geq 0 \right\} . \]
The counterpart of (d-2:x) for model (EMRP) is the same as before except $h^v_i$ is replaced by $h^v_i + b^w_i$. Also, our discussion on the effects of bounding overtime applies verbatim to (EMRP) as well.
Chapter 3  Special Structure Constraints

3.1 Introduction

Chapter 3 deals with the properties of the inventory-production (I-P) balance constraint systems of our multistage models. Whereas Chapter 2's scope is high-level, Chapter 3 is largely a detailed mathematical analysis. We are specifically interested in the mathematical properties of the inventory-production balance constraint system of model (BMRP) (constraint set (1-2) of Section 1.2 plus appropriate nonnegativity restrictions on variables) and the inventory-production-orders balance system of (EMRP) (constraint sets (1-7) and (1-10) plus nonnegativity restrictions). In addition, we explore how the mathematical properties of the balance systems translate into specialized algorithms for solving linear programs with feasible regions defined by these systems. The existence of these algorithms provides the impetus for the nested decomposition approach, outlined in Chapter 2, that isolates and exploits the structure of the balance systems.

We develop an algorithm in Section 3.2, known as the row-column sequencing algorithm, for permuting rows and columns of the I-P balance system of any multistage system within our scope of interest (as defined in Section 1.1). The intent of the algorithm is to produce an equivalent constraint system with specific properties. We show (Section 3.5) that such systems are Leontief substitution systems. (See, for example, Arrow (1951), Gale (1960), Fiedler and Ptak (1962), Veinott (1968,1969) and Koehler, Whinston and Wright (1975).) In Section 3.7 we establish that these systems also have the strong properties of both dynamic Leontief (Dantzig (1955)) and totally Leontief (Veinott (1968) and Koehler, Whinston and Wright (1975)) systems. Hence we call these systems totally dynamic. We show in the second and third sections of this chapter that, in general, the permuted system of the inventory-production balance constraint set of (BMRP) is not totally dynamic. However, the discrepancy can be well defined and, for most applications, the relaxation of relatively few constraints of the permuted system yields a
totally dynamic system. For the sake of completeness (and mathematical interest), we develop the most general conditions under which the constraints that cause the permuted system to violate the properties of totally dynamic systems can be eliminated by inspection without altering the feasible region of the permuted system.

Section 3.5 establishes the mathematical foundation for specialized algorithms that solve linear programs over totally dynamic systems. Therein we develop the mathematical properties of convex polytopes formed by such systems. This section concludes by showing that a set of bases capable of generating all extreme points of these polytopes can be explicitly generated. Each basis in this set is triangular, and the set is independent of the right hand side demand vector, as long as it is nonnegative.

This result leads to Section 3.6, which develops an algorithm for solving LPs over totally dynamic systems. The algorithm scans each column of the constraint set exactly once to determine a primal optimal basis and corresponding shadow prices. Moreover, an optimal solution is determined without doing any matrix inversion anywhere in the algorithm.

Section 3.7 discusses how totally dynamic systems are related to special structure linear constraint sets considered by other researchers. Therein we show how totally dynamic systems relate to the well-developed theory of Leontief substitution systems and the research literature concerning optimization over such systems.

The relationship of the balance system of (EMRP) to totally dynamic systems is the subject of Section 3.8. We show that many of the results found for the (BMRP) system have direct analogues in the extended system of (EMRP). Thus the "closeness" of I-P balance systems to totally dynamic systems is not lost when one generalizes the fixed demands of (BMRP) to the order variables of (EMRP).

Another interesting topic is how to resolve an important discrepancy between an I-P balance system and its idealized totally dynamic counterpart. One would like to exploit the fact that, for most applications, the large majority of constraints form a totally dynamic subsystem. In Section 3.9 we show that a form of resource-directive decomposition is an intriguing avenue
of pursuit, if one is willing to impose certain restrictions that may cut off the global optimal solution (to the LP problem defined over the I-P balance constraint system). Section 3.10 explores other methods for attacking this problem.
3.2 A Row - Column Sequencing Algorithm

We begin our discussion with an example possessing a simple product structure. We refer the reader to Figure 3-1. The planning horizon has four time periods, and minimum lead times and component relationships are indicated in the figure. Notice inventory variables are never needed for the final time period; they can always be constrained to zero without affecting optimality. In general, we can target final inventories to be any nonnegative amount for each item. These values can then be incorporated into the demand for items in the final period. Here we have chosen to have no inventory at the end of the cycle over which the optimization is taking place. In any event, final period inventory variables can be eliminated.

Also, we set \( P_{it} = 0 \) for all \( i, t \) such that \( t > T - L_i \) (\( T = 4 \) in this example) since any production initiated in these periods will not be available to satisfy demand (exogenous or internally generated) for item \( i \) over the time frame of interest. (In general planning environments with rolling horizons, one may want to retain these production variables to meet demands in periods that immediately follow the current planning horizon.)

The constraint matrix corresponding to the I-P balance constraints is displayed with the rows and columns in a particular order in Figure 3-1. The order in which rows and columns are sequenced adheres to general sequencing rules for arbitrary multistage problems. We outline these rules in the following algorithm:

Row-Column Sequencing Algorithm

1. Each row corresponds to some item \( i \) in some period \( t \).

2. All periods for item \( i \) appear as consecutive rows, which we call a block and are ordered from \( t = 1 \) to \( t = T \) in the constraint matrix.
3. All items on the same level (as defined by the production structure outline of Section 1.1) will have their blocks appear consecutively. This construct we call a group.

4. The groups corresponding to levels will be sequenced from highest level (i.e., items with maximal depth) to lowest (finished goods) as we move from the first to last row.

5. Columns will also be blocked by item. Columns refer either to production or inventory of item $i$ in period $t$. The blocks of columns corresponding to items appear in the same sequence moving left to right as blocks of rows appear moving top to bottom.

6. Within a block of columns, the order sequence of columns depends on the minimum lead time $L_i$ of the item $i$, as follows:

$$I_{t_1}, I_{t_2}, \ldots, I_{t_{L_i}}, P_{t_1}, I_{t_{L_i+1}}, P_{t_2}, \ldots, I_{t_{T-1}}, P_{t_{T-L_i}} \quad (3-1).$$

In the example of Figure 3-1, we have four blocks corresponding to the four production items, two levels (0,1), and two groups. One row group refers to the rows of items 1 and 2, the other to the remaining rows. The column groups are analogous to the row groups.

Note the $P_{it}$ variables begin contributing supply of item $i$ from period $L_i$ onward. As we noted in Section 1.2, the $P_{it}$ variables $\forall i,t$ such that $1 - L_i \leq t \leq 0$ are not really variables: they are known holdover production from periods immediately preceding the time frame over which optimization is to take place. We therefore bring these quantities and initial inventories $I_{it} \forall i$ to the right hand side demand vector of the constraint set. This revised right hand side vector $r$ is called the net demand vector. A necessary condition for a feasible solution to exist for our problem is that items with nonzero minimum lead times must have their first period net demand nonpositive. More generally, summing net demands for periods 1 through $t$ for any $i,t$ such that $1 \leq t \leq L_i$ must produce a nonpositive quantity or else the problem is infeasible. (Necessary and sufficient conditions for the feasibility of the inventory-production balance constraints of model (BMRP) are given in Section 3.9.)
Figure 3-1 Inventory-Production Balance Constraints for a Simple Product Structure
3.3 Inventory-Production Constraint Types and Special Constraint Sets

The I-P balance constraint set can be partitioned into three mutually exclusive and collectively exhaustive subsets, which we now describe:

Type 1: First period constraints for items with nonzero minimum lead times: Such constraints have no inventory carried over from the preceding period, and no production becomes available in these initial periods because of the positive minimum lead times. Remember, initial inventory and holdover production have already been incorporated into the demand vector. Thus these constraints have no variables with positive coefficients; they are also the only constraints with this property. A nonpositive right hand side net demand in all such constraints is necessary for feasibility.

Type 2: The right hand side net demand is negative and the constraint is not of type 1: Assume the constraint is for item $i$ in period $t$. If $L_i > 0$, a negative net demand implies $1 \leq t \leq L_i$, since only the incorporation of holdover production or initial inventory can produce a negative net demand. For items with zero minimum lead time, the first period constraint, and only the first period constraint, can be negative because of initial inventory. For type 2 constraints, some variable with a negative coefficient, which will always exist, must be positive in any feasible solution.

Type 3: The constraint is not of type 1 and the right hand side net demand is nonnegative.

Constraints of types 1 and 2 can only be found in the first $L_i$ constraints of any item $i$ with $L_i > 0$, but not all of these $L_i$ constraints must be of type 1 or 2. Intuitively, one thinks of the first $L_i$ periods – the number of periods that must pass until production initiated in the first period becomes available – as relatively few in relation to the number of periods over which we are
optimizing. Consequently, the number of constraints of types 1 and 2 should generally be relatively small.

We now outline a specific class of constraint sets that we hereinafter refer to as *totally dynamic systems*. The rationale for this name was discussed in the introduction of this chapter.

**Defining Properties of Totally Dynamic Systems**

A system \( \{ \mathbf{x} : \mathbf{A}\mathbf{x} = \mathbf{d}, \mathbf{x} \geq \mathbf{0} \} \), where \( \mathbf{A} \) is an \( m \times n \) \((m \leq n)\) matrix, is called totally dynamic if it exhibits the following properties:

**Property 1:** Moving left to right along any constraint, all of the positive coefficients are encountered before any of the negative coefficients.

**Property 2:** At least one variable has a positive coefficient in each constraint.

**Property 3:** The right hand side of any constraint is nonnegative.

**Property 4:** Let \( j_c \) be the number of zero coefficients encountered moving left to right along constraint \( c \) until the first positive coefficient is found. As we move from the first row to the last row of the constraint matrix, \( j_c \) is a strictly increasing function. Hence the constraint matrix is block triangular.

**Property 5:** Each column has exactly one positive coefficient.

We define a *totally dynamic matrix* to be the constraint matrix of a totally dynamic system; that is, \( \mathbf{A} \) is a totally dynamic matrix whenever \( \{ \mathbf{x} : \mathbf{A}\mathbf{x} = \mathbf{d}, \mathbf{x} \geq \mathbf{0} \} \) is a totally dynamic system.

Note that totally dynamic matrices are completely defined by properties 1, 2, 4, and 5 of totally dynamic systems.
3.4 Special Structure in Inventory-Production Balance Constraints

In Figure 3-2 we have taken the example of Figure 3-1 and labelled the constraints of type 1 with arrows. Notice that the constraint matrix of the remaining constraints (illustrated in Figure 3-3) satisfies the properties of totally dynamic matrices. This is an example of a more general result:

**THEOREM 3-1:** For any multistage production system of the type outlined in Section 1.1, the row-column sequencing algorithm applied to the I P balance constraint system produces an equivalent constraint set with the following property: the submatrix of the new constraint matrix obtained by eliminating constraints of type 1 is a totally dynamic matrix.

**PROOF:** We call this submatrix the output matrix. We will verify that the output matrix satisfies each of the properties of totally dynamic matrices (i.e., properties 1,2,4, and 5 of the properties of totally dynamic systems) one-by-one. The reader may want to refer to Figure 3-3 as a visual aid.

**Property 2:** Take an arbitrary constraint of the output matrix and assume the constraint is for item i in period t. We label the constraint it. Since all type 1 constraints are eliminated, constraint it has at least one variable with a positive coefficient. This variable corresponds to inventory carried over from period t-1, production initiated in period t - L_t if t ≥ L_t + 1, or both.

**Property 1:** Note that the verification of property 2 also identified all possible variables with positive coefficients in constraint it. A constraint it has variables with negative coefficients from exactly two potential sources. One of these sources is I_{it}, the inventory to be held from period t to period t+1. The second source is from production variables P_{jt} where j is any immediate successor of i in the product structure; that is, item j has item i as a component. The coefficient of P_{jt} in constraint it is -a_{ij}.

It is clear that within the block of columns for item i, where a block is defined in the row-column sequencing algorithm, at most three variables of block i have nonzero coefficients in
constraint it. These variables correspond to elements of the set $S = \{l_{i,t-1}, p_{i,t-L}, l_d\}$. Elements of $S$ do not necessarily correspond to variables: for example, if $1 \leq t \leq L$, then $p_{i,t-L}$ is not a variable. Let $S^1$ be
\[
\begin{align*}
I_{11} & P_{11} I_{12} P_{12} I_{13} P_{13} I_{21} P_{21} I_{22} P_{22} I_{31} P_{31} I_{32} P_{32} I_{41} P_{41} I_{42} P_{42} I_{43} P_{43} I_{44} P_{44} \\
i = 1, & \quad t = 1 \quad -1 \\
& \quad t = 2 \quad 1 \quad y_1 \quad -1 \\
& \quad t = 3 \\
& \quad t = 4 \\
& \quad 1 \quad y_1 \\
& \quad -2 \quad -3 \\
i = 2, & \quad t = 1 \\
& \quad t = 2 \\
& \quad t = 3 \\
& \quad t = 4 \\
& \quad 1 \quad -1 \\
& \quad 1 \quad y_2 \quad -1 \\
& \quad 1 \quad y_2 \\
& \quad -5 \\
i = 3, & \quad t = 1 \\
& \quad t = 2 \\
& \quad t = 3 \\
& \quad t = 4 \\
& \quad 1 \quad -1 \\
& \quad 1 \quad y_3 \quad -1 \\
& \quad 1 \quad y_3 \\
i = 4, & \quad t = 1 \\
& \quad t = 2 \\
& \quad t = 3 \\
& \quad t = 4 \\
& \quad 1 \quad y_4 \quad -1 \\
& \quad 1 \quad y_4 \quad -1 \\
& \quad 1 \quad y_4 \\
& \quad -5 \\
& \quad -5 \\
\end{align*}
\]

<table>
<thead>
<tr>
<th>level</th>
<th>Product Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3 \quad 4</td>
</tr>
<tr>
<td></td>
<td>a_{13} = 2 \quad L_1 = 1</td>
</tr>
<tr>
<td></td>
<td>a_{14} = 3 \quad L_2 = L_3 = 2</td>
</tr>
<tr>
<td>1</td>
<td>1 \quad 2</td>
</tr>
<tr>
<td></td>
<td>a_{24} = 5 \quad L_4 = 0</td>
</tr>
</tbody>
</table>

Figure 3-2: I-P Balance Matrix with Arrows on Type 1 Rows
Figure 3-3: I-P Balance Matrix with Type 1 Rows Eliminated
the subset of \( S \) that are indeed variables.

From (3-1), the sequence of elements of \( S^1 \) encountered by moving left to right along constraint \( i \) is derived by forming an ordered sublist \( T^1 \) of the ordered list \( T = \langle I_{i,t-1}, P_{i,t-L_t}, I_{it} \rangle \). Specifically, a variable is an element of \( T^1 \) if and only if it is an element of \( S^1 \). Since the first two terms of \( T \) have positive coefficients, and the last has a negative coefficient within constraint \( i \), we see that the variables of block \( i \) with positive coefficients are encountered before those with negative coefficients. Moreover, these variables with positive coefficients constitute the full set of variables with positive coefficients in constraint \( i \).

The row-column sequencing algorithm states that blocks of columns are sequenced in the same order left to right as blocks of rows are sequenced top to bottom. Also, blocks of rows corresponding to items on the same level of the product structure appear contiguously in the rows as groups, and these groups are sequenced from highest to lowest levels as we move down the rows. Hence the same properties hold for blocks and groups as we move left to right along the columns of the constraint matrix.

Consider any immediate successor item \( j \) of item \( i \) and its variable \( P_{ij} \) with coefficient \(-a_{ij}\) in constraint \( i \). \( P_{ij} \) appears in column block \( j \). This block is to the right of column block \( i \), because \( j \) is on a lower level than \( i \). Therefore \(-a_{ij}\) is found to the right of the variables of block \( i \) with nonzero coefficients in constraint \( i \). In particular, \(-a_{ij}\) appears to the right of the variables with positive coefficients in constraint \( i \). Consequently, property 1 of the totally dynamic matrices (systems) is satisfied.

**Property 4:** Consider two constraints \( c \) and \( c+1 \) that appear contiguously as we move down the rows of the constraint matrix. Assume \( c \) and \( c+1 \) correspond to different production items, and let \( i_c, i_{c+1} \) be the items corresponding to constraints \( c, c+1 \) respectively. Because of the sequencing conventions in the row-column sequencing algorithm, the column block for \( i_{c+1} \) appears immediately after the column block for \( i_c \). We've just seen that all of the positive coefficients in constraint \( i \) come from variables within column block \( i \), and all coefficients of the constraint in preceding blocks are zero. Moreover, column blocks always have a positive
number of columns. It follows that $j_{c+1} - j_c \geq 1$, where $j_c$ is defined in property 4 of totally dynamic systems.

Now assume $c$ and $c+1$ are constraints of a common item $i$. Hence $c$, $c+1$ correspond to periods $t$, $t+1$ of item $i$ for some period $t$. Inventory holdover variable $I_{it}$ thus exists with a -1 coefficient in constraint $t$ and a +1 coefficient in constraint $t+1$. Referring to the ordered list $T$ used in the verification of property 1, it follows that the +1 coefficient in constraint $t+1$ is the first nonzero coefficient of the constraint encountered moving left to right. Since $I_{it}$ has a -1 in constraint $t$, and properties 1 and 2 have been verified, it follows that $j_{c+1} - j_c \geq 1$, and property 4 is verified.

**Property 5:** Each column of the inventory-production balance constraint set corresponds either to a production variable $P_{it}$ or an inventory variable $I_{it}$. Consider $P_{it}$ and let $P_i$ be the set of production items which are immediate predecessors of $i$ in the product structure. For every $j \in P_i$, production of $i$ in period $t$ means a demand of $a_{ji}$ item $j$s is created in period $t$ for every unit of $i$ whose production begins in $t$. Hence, a $-a_{ji}$ coefficient exists in constraint $jt$ for variable $P_{it}$. The only other nonzero coefficient for $P_{it}$ occurs in the constraint where the production of $i$ initiated in $t$ becomes available; namely, there is a yield factor $y_i > 0$ in constraint $i$, $t + L_i$. (We assume $t \leq T - L_i$, because production variables for $i$ exist only up to period $T - L_i$.)

For variable $I_{it}$, it is clear $I_{it}$ has a coefficient of -1 in constraint $it$, a coefficient of +1 in constraint $i$, $t+1$, and no other nonzero coefficients. Hence, property 5 holds for our constraint matrix, the four properties of totally dynamic matrices are verified for an arbitrary output matrix, and Theorem 3-1 is proven. □

Since generally only a submatrix of the inventory-production balance constraint matrix is a totally dynamic matrix, and the demand vector may have negative components because of initial inventories and holdover production, we cannot expect the entire I-P balance system to define a feasible region that can be defined alternately by a totally dynamic system. It is therefore natural to attempt to identify the general conditions under which the entire I-P
system is "effectively" totally dynamic. That is, we would like to know the conditions under which constraints violating the totally dynamic properties can be eliminated by inspection from a feasible system without affecting the feasible region of the problem. We now outline these conditions. Figure 3-1 can serve as a visual aid for the reader, if necessary.

Consider constraints of type 1 for items with no predecessors. We have observed that all such constraints have nonpositive coefficients for all variables, implying property 2 of special structure constraint sets is violated. Therefore, given our objectives, we must be able to find conditions under which type 1 constraints can be effectively eliminated from the optimization model. A positive right hand side value obviously produces an infeasible type 1 constraint and is not of interest. A negative right hand side value provides, in general, no grounds for eliminating type 1 constraints. A zero right hand side value does, however, because such a value constrains all variables (which take on only nonnegative values) with nonzero coefficients in the constraint to zero. The constraint can therefore be eliminated. Note that the first constraint of Figure 3-1 is such a constraint. Since right hand side demands are net demands, a zero net demand for the first period constraint of any item \( i \) with \( L_i > 0 \) is equivalent to \( r_{i1} = d_{i1} - l_{i0} - P_{i,1,L_i} = 0 \). Note that since variables with nonzero coefficients in type 1 constraints are now constrained to zero, they can be eliminated from the optimization problem as well.

We cannot have any type 2 constraints, because we have no general rationale for eliminating such constraints by inspection. Consider, then, constraints of the form \( it \) where \( 2 \leq t \leq L_i \) and \( i \) has no predecessors. In other words, these are constraints not of type 1 that receive only holdover production. In particular, let's first examine the case \( t = 2 \). We notice the only positive coefficient for such constraints comes from \( I_{i1} \). But \( I_{i1} \) has been shown to be zero in any feasible solution because it has a -1 coefficient in constraint \( i1 \). (Constraint \( i1 \) is a type 1 constraint.) Hence to maintain feasibility and not violate the conditions we seek to establish, the right hand side value of \( i2 \) must be zero (i.e., \( r_{i2} = d_{i2} - P_{i,2,L_i} = 0 \)), and all variables with nonzero coefficients in \( i2 \) are thus constrained to zero in any feasible solution.
Specifically, \( I_{i2} = 0 \). So if \( L_i \geq 3 \) we see similarly that \( r_{i3} = d_{i3} - P_{i,3-L_i} = 0 \), and all variables with nonzero coefficients in constraint \( i3 \) are zero in any feasible solution. Simple induction shows, then, that all net right hand side values of constraints of the form \( it \) such that \( 1 \leq t \leq L_i \), adjusted for initial inventory and holdover production, must be zero for all items with no predecessors. Furthermore, all variables with nonzero coefficients in such constraints are constrained to zero and thus can be eliminated, along with the constraints, from the optimization problem.

We take note of the fact that constraints of the form \( i, L_i + 1 \) where \( L_i > 0 \) do not benefit from \( I_{i,L} \), since \( I_{i,L} \) is now constrained to 0. However, potential production initiated in period 1 becomes available ("filtered" by the yield factor \( y_i \)) in period \( L_i + 1 \). Also, the demand for this period must be nonnegative because it receives no holdover production. Consequently, constraint \( i, L_i + 1 \) remains intact and forces no variable to zero that has not been constrained to zero already. Moreover, periods \( L_i + 2, \ldots, T-1 \) potentially receive both inventory and production but no holdover production. Thus, these constraints also remain fully intact and force no further variables to zero.

So, necessary conditions for our objective is that net demands \( r_{it} \) must be zero for any item \( i \) that has no predecessors and \( 1 \leq t \leq L_i \). In fact, it is clear that these conditions apply to items on all levels of the product structure. There is a complicating factor for items that have predecessors, however. Namely, the elimination of constraints of immediate predecessor items \( j \) of item \( i \) (i.e., \( j \) is a component of \( i \)) can constrain production of \( i \) to zero for a certain number of additional periods. This can mean the elimination of additional constraints (and hence the variables with nonzero coefficients in these constraints) of \( i \) beyond period \( L_i \).

For example, consider an item \( i \) that has \( j \) as an immediate predecessor, and \( j \) has no predecessors. The elimination of constraints \( j1; j2; \ldots; j, L_j \) implies \( P_{i1} = P_{i2} = \ldots = P_{iL_i} = 0 \). The production of \( i \) reflected in these variables becomes available in periods \( L_i + 1, L_i + 2, \ldots, L_i + L_j \) respectively. Since constraints \( i1; \ldots; i, L_i \) have already been eliminated (implying variable \( I_{i,L_i} \) has been eliminated), the implication of the above is that constraints \( i, L_i + 1; \ldots; i, L_i + L_j \) (and
corresponding variables with negative coefficients in these constraints) must be eliminated as well. However, the production variable $P_{iL_i+1}$ is not eliminated by constraints of item $j$, and so this variable is still available as a variable with its positive coefficient in constraint $i$, $L_i + L_j + 1$, unless of course $P_{iL_i+1}$ is eliminated by some other immediate predecessor of $i$.

Recall that $P_i$ is the set of immediate predecessors of item $i$. We define $L_i^*$ to be the cumulative lead time of item $i$; that is,

$$L_i^* = \begin{cases} L_i & \text{if } P_i = \emptyset \\ L_i + \max_{j \in P_i} L_j^* & \text{otherwise.} \end{cases}$$

If we follow the line of reasoning begun above to its fruition, the result is the constraining of all right hand side net demands $r_{it}$ to zero $\forall i,t$ such that $1 \leq t \leq L_i^*$, the elimination of the corresponding constraints from the optimization problem, and the constraining of all variables with nonzero coefficients in these constraints to zero in any feasible solution to the I-P balance constraint system. The optimization problem is thus reduced to an equivalent problem which deals only with the remaining I-P balance constraints. If we stipulate that the net demands of all first period constraints are nonnegative $\forall i$ such that $L_i^* = 0$ (note that all such items have $L_i = 0$), then all such constraints are of type 3 with nonnegative demands. A straightforward variation of Theorem 3-1 assures us that this constraint set, rearranged by the row-column sequencing algorithm, possesses the properties of totally dynamic systems. As we will see shortly, such a system is always feasible. Hence the I-P balance system is feasible, and we can construct the feasible region from the feasible region of the totally dynamic system and the variables constrained to zero. We conclude, then, that the conditions we seek that allow us simply to eliminate constraints and produce an equivalent totally dynamic system are: the net demands $r_{it}$ must equal zero $\forall i,t$ such that $1 \leq t \leq L_i^*$, and $r_{i0}$ must be nonnegative $\forall i$ such that $L_i^* = 0$. 

69
We give an example of these results by considering a revision of the example of Figure 3.1. The revision consists solely of changing the number of time periods to some number greater than four. Please note Figure 3-4 in which the constraints and variables to be eliminated are indicated by arrows. The net demands of these constraints are zero.

We observe that a sufficient condition for the full I-P balance constraint set to effectively possess the totally dynamic properties and remain feasible is for \( d_{il} = P_{i,t} \cdot L_i = 0 \), \( \forall i, t \) such that \( t = 1, \ldots, L_i \), \( d_{il} = 0 \) \( \forall i, t \) such that \( t = L_i + 1, \ldots, L_i^* \), and \( I_{i,0} = 0 \) \( \forall i \). That is, there is no exogenous demand for items in the first \( L_i^* \) periods \( \forall i \), no holdover production, and no initial inventory. We call any problem satisfying these conditions a pure production problem. This name is due to the fact that all demands occur after the cumulative lead times, and all finished items, subassemblies, and components used to satisfy demand must be manufactured during the time frame over which the optimization is taking place. In other words, all items must be made from "scratch."

We do not claim the conditions we have found are necessary for the feasible region to be equivalent to a region that can be represented by a totally dynamic system. Such a result is stronger than what we have demonstrated. Again, we have demonstrated a sufficient set of conditions under which constraints violating the totally dynamic properties can simply be eliminated without altering the problem's feasible region.

Since holdover production and initial inventories are often present, one might ask what alternatives are available in such situations. We take up this question in Section 3.9.
\[ I_{11} P_{11} I_{12} P_{12} I_{13} P_{13} I_{21} P_{21} I_{22} P_{22} I_{31} P_{31} I_{32} P_{32} P_{33} I_{34} P_{41} I_{41} P_{42} I_{42} P_{43} I_{43} P_{44} \text{ right hand side} \]

\[
\begin{array}{ccccccc}
\text{i=1, t=1} & -1 & -2 & -3 & 0 \\
\text{t=2} & 1 & y_1 & -1 & -2 & -3 \\
\text{t=3} & 1 & y_1 \\
\text{t=4} & 1 & y_1 \\
\text{.} & & & & & & \\
i=2, t=1 & -1 & -5 & 0 \\
\text{t=2} & 1 & y_2 & -1 & -5 & 0 \\
\text{t=3} & 1 & y_2 \\
\text{t=4} & 1 & y_2 \\
\text{.} & & & & & & \\
i=3, t=1 & -1 & 0 \\
\text{t=2} & 1 & y_3 & -1 & 0 \\
\text{t=3} & 1 & y_3 \\
\text{t=4} & 1 & y_3 \\
\text{.} & & & & & & \\
i=4, t=1 & 0 \\
\text{t=2} & 1 & y_4 & -1 & 0 \\
\text{t=3} & 1 & y_4 & -1 & 0 \\
\text{t=4} & 1 & y_4 & 0 \\
\text{.} & & & & & & \\
\end{array}
\]

\[
\text{level} \quad \text{Product Structure}
\]

\[
\begin{array}{cc}
0 & 3 \quad 4 \\
1 & 1 \quad 2 \\
\end{array}
\]

\[
\begin{align*}
a_{13} &= 2 & L_1 &= 1 & L_1^* &= 1 \\
a_{14} &= 3 & L_2 &= L_3 = 2 & L_2^* &= 2, L_3^* &= 3 \\
a_{24} &= 5 & L_4 &= 0 & L_4^* &= 2 \\
\end{align*}
\]

Figure 3-4
3.5 Properties of Totally Dynamic Systems and Matrices

In order to develop a fast algorithm for solving LPs over totally dynamic systems, we first investigate the properties of convex polyhedra representing the feasible regions defined by such systems. An important first step in this investigation is to establish that totally dynamic systems satisfy the properties of Leontief substitution systems. We now make a series of statements leading to the definition of Leontief substitution systems: A matrix is called \textit{pre-Leontief} if each column has at most one positive element. A pre-Leontief matrix \(A\) is \textit{Leontief} if each column has exactly one positive element and there exists \(x \geq 0\) such that \(Ax > 0\). We call the formulation \(\{x : Ax = d; x \geq 0\}\), where \(d \geq 0\) and \(A\) is Leontief, a \textit{Leontief substitution system}. (References for Leontief substitution systems were given in the introduction of this chapter.)

We state the following theorem, which illuminates a well-known property of Leontief models, without proof.

\textbf{THEOREM 3.2:} Assume we have a linear program in which the constraint matrix is a Leontief matrix, the right hand side vector \(d\) is nonnegative, and an optimal solution exists to the LP. Then, the convex polytope, formed by this matrix, nonnegativity restrictions on variables, and any right hand side demand vector \(d \geq 0\), is nonempty. Furthermore, for a given linear objective function, there exists an optimal primal basis which is optimal for any demand vector \(d \geq 0\). \(\square\)

It is clear that any inventory-production matrix of a general (BMRP) model and any totally dynamic matrix are pre-Leontief. Additionally, each column of such matrices has exactly one positive element. We notice immediately the block triangular structure implied by property 4 assures the linear independence of the rows of totally dynamic matrices. As promised, the next
Theorem ties totally dynamic matrices to Leontief matrices: namely, the result of Theorem 3.3 combined with the existence of exactly one positive element in each column implies totally dynamic matrices are Leontief.

**THEOREM 3.3:** Any system $Ax = d$, $x \geq 0$ where $A$ is totally dynamic is feasible for any demand vector $d \geq 0$.

**PROOF:** Initialize all variables to zero. Start with the last row $m$ of the constraint matrix. Find some variable $X_m$ with a positive coefficient in row $m$. Set the distinguished variable $X_m$ to satisfy constraint $m$. Note that property 2 of totally dynamic matrices guarantees the existence of such a variable.

Move to row $m-1$. Setting $X_m$ in row $m$ cannot decrease the demand in row $m-1$ because $X_m$ has a nonpositive coefficient in row $m-1$. Employ the same rule in row $m-1$ to set a distinguished variable as was used in row $m$. We continue in this fashion until all rows have been processed. Upon completion, the right hand side demand has been satisfied by setting a unique variable in each row. This completes the proof. □

The property of this theorem is true of Leontief matrices in general, as we saw in Theorem 3.2. We could have proved that totally dynamic matrices are Leontief by showing the existence of some $x \geq 0$ such that $Ax > 0$ for any totally dynamic matrix $A$. The property of Theorem 3.3 would then have been implicitly established. Instead, we proved the stronger property directly, a consequence of which is that totally dynamic matrices are seen to be Leontief.

For a totally dynamic system with a strictly positive demand vector, some variable with a positive coefficient in arbitrary constraint $m$ must be set positive in order to maintain feasibility. We define a column (variable) to be critical for a constraint if its coefficient in the constraint is positive. Properties 2 and 3 of totally dynamic systems imply that the set of critical variables for any constraint is nonempty and that each variable is critical for exactly
one constraint. The cardinality of the critical variable set for a constraint in an LP balance matrix is either one or two, depending on which of the variables $I_{t,t-1}$ and $P_{t,t-L}$ exist.

There is an interesting relationship between critical columns and basic feasible solutions for totally dynamic systems with positive demand vectors.

**THEOREM 3-4**: Consider an LP with a totally dynamic constraint matrix and a positive right hand side demand vector. Any feasible basis for this system contains exactly one critical column for each constraint.

**PROOF**: Let $m$ be the number of rows in the constraint set. We've noted that the constraint matrix is of rank $m$. We've also seen that some critical variable of each constraint must be positive in any feasible solution. Hence, at least one critical column is in any feasible basis for each constraint. Since each variable is critical for exactly one constraint, a basis with more than one critical column for some constraint implies $m_1 > m$ columns in the basis, an impossibility; hence, the result.

Theorem 3-4 leads to a characterization of feasible bases for totally dynamic systems with positive demand vectors. From this set of bases, extreme points of the feasible region can be obtained.

**THEOREM 3-5**: Consider an LP of the type defined in Theorem 3-4. Let $S$ be the set of feasible bases and let $T$ be the set of all possible combinations of critical columns such that exactly one critical column for each constraint is present in each combination. Then $S = T$.

**PROOF**: Theorem 3-4 guarantees the set of feasible bases $S$ is a subset of set $T$, as defined in the statement of the theorem. An implication of the five properties of totally dynamic systems is the following: if variable $X$ is critical for constraint $n$, then $X$ has all zero coefficients in its column for all constraints $l > n$. Therefore, any combination of columns where each is critical to a unique constraint is easily seen to be a set of linearly independent columns.
Moreover, taking any element \( s \in T \) yields a combination of \( m \) linearly independent columns, and hence the \( m \) columns constitute a basis.

To show the variables of \( s \) constitute a feasible basis, we review the proof of Theorem 3-3 and note that the critical variables of \( s \) can be used as the set of variables to be set positive in the proof. Hence, \( s \) is a feasible basis. Since \( s \) was an arbitrary element of \( T \), we see that \( T \) is a subset of the set of feasible bases \( S \), and the theorem is proven. \( \square \)

The set \( T \) of Theorem 3-5 is independent of the demand vector of the LP referenced in theorems 3-4 and 3-5. Hence, we can state the following corollary to Theorem 3-5:

**COROLLARY 3-1:** The set \( T \) defined in Theorem 3-5 is the set of feasible bases for all LPs formed by taking all positive demand vectors. \( \square \)

Observe how the demand vector \( d \) is constrained to be positive in theorems 3-4 and 3-5. We are interested in generalizing these results, if possible, to include all nonnegative demand vectors \( d \geq 0 \). It is not true that \( S \), the set of feasible bases in this general case is equivalent to \( T \), the set derived by taking all possible combinations of columns such that there is exactly one for every constraint. Consider, for example the totally dynamic system

\[
\begin{bmatrix}
1 & 0 & 0 & -1 & 0 \\
0 & 1 & 0 & 0 & -1 \\
0 & 1 & 1 & 1 & 1 \\
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2 \\
x_3 \\
x_4 \\
x_5 
\end{bmatrix} =
\begin{bmatrix}
0 \\
0 \\
1 
\end{bmatrix}.
\]

The feasible basis
\[
\begin{bmatrix}
0 & -1 & 0 \\
0 & 0 & -1 \\
1 & 1 & 1
\end{bmatrix}
\]

contains no critical variables of either constraint 1 or 2. The generalization we seek is the following:

**THEOREM 3-6:** Consider any totally dynamic system (with \(d \geq 0\)). Define the sets \(S\) and \(T\) as in Theorem 3-5. Then \(T \subseteq S\). Moreover, any extreme point of this system can be generated by an element of \(T\).

**PROOF:** The fact that \(T \subseteq S\) is verified by noting the validity of the second half of Theorem 3-5 for nonnegative as well as strictly positive demand vectors. The only difference here is that a critical variable may be set to zero.

To prove the remainder of the theorem, we require the result of Theorem 3-2. (It is straightforward to show that the feasible region of any totally dynamic system is bounded, and this implies that any totally dynamic matrix is a Leontief matrix that satisfies the conditions of Theorem 3-2.) Theorem 3-2 implies there is some basis \(B_c \in T\) which is optimal for any nonnegative demand vector for a given cost vector \(c\). (\(B_c\) may depend on \(c\).) Assume the theorem is false. Let \(x^*\) be an extreme point of the system which cannot be generated by an element of \(T\), and \(B^* \notin T\) a basis which generates \(x^*\). Construct a cost function \(c\) such that \(c_i\) equals 0 if \(i\) is an index of a column in \(B^*\) and equals 1 otherwise. Clearly \(x^*\) is an optimal extreme point of the system with cost vector \(c\).

Consider the basis \(B_c \in T\) whose existence is guaranteed by Theorem 3-2. Let \(x_c\) be the optimal extreme point generated by \(B_c\). From our assumptions, \(x_c\) and \(x^*\) must be distinct optimal extreme points. But it must be that \(X_c\), equals zero for any index \(i\) which references a column not in \(B^*\), since otherwise \(x_c\)’s cost strictly exceeds \(x^*\)’s cost. This implies \(x^* = x_c\), a contradiction, and establishes the theorem. \(\square\)
In our example above, the identity matrix establishes the extreme point

\[
\begin{bmatrix}
  x_1 \\
  x_2 \\
  x_3 \\
  x_4 \\
  x_5
\end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}
\]

as well as the basis

\[
\begin{pmatrix}
  0 & -1 & 0 \\
  0 & 0 & -1 \\
  1 & 1 & 1
\end{pmatrix}
\]

Theorem 3.6 establishes that even though \( T \) is not necessarily the complete set of feasible bases, it is a sufficient set for generating any extreme point of any feasible region for any nonnegative demand vector. Moreover, all elements of \( T \) always generate an extreme point for any nonnegative demand vector. For any totally dynamic matrix \( A \) and its corresponding set of matrices \( T \), we refer to \( T \) as the generating set of \( A \) because \( T \) generates all extreme points of any totally dynamic system that can be constructed from \( A \). Observe that \( T \) is the set of full, square totally dynamic submatrices of \( A \).
3.6 An Algorithm for Solving Linear Programs Defined by Totally Dynamic Systems

The framework is now in place for developing an algorithm for solving linear programs with totally dynamic systems defining the feasible region. Theorem 3-6 is particularly important in this development. This theorem established that there exists a set $T$ of bases, independent of the right hand side nonnegative demand vector, that can generate all extreme points of the feasible region of the totally dynamic system. Furthermore, an explicit mechanism is at hand for generating these bases. This follows from the fact that $T$ is the set of full, square totally dynamic submatrices of the totally dynamic constraint matrix.

It is well known that the three properties that a primal, dual pair of vectors must possess to ensure they are both optimal in respective primal, dual linear programs are primal feasibility, dual feasibility, and complementary slackness. We will maintain complementary slackness throughout the algorithm in the usual simplex manner: that is, we will choose a primal feasible basis and construct the unique dual vector which satisfies the dual constraints formed by this basis with no slack in any basic constraint. Because of Theorem 3-6, we can mechanically generate primal bases without ever considering the demand vector. Hence, we need not concern ourselves with primal feasibility.

We will see that an optimal primal basis and the corresponding optimal dual variable can be generated very efficiently. Specifically, each column will be accessed exactly once in a scan moving left to right across the columns of the constraint set, and the optimal primal basis and dual vector can be constructed component by component during this scan. The scan terminates with the optimal basis and dual vector at hand.

Since the optimal primal basis is triangular, an optimal primal solution can be obtained by simple backwards substitution involving no matrix inversion. Furthermore, we will see that no matrix inversion is required anywhere within the algorithm.

We now make a series of definitions:
Let

\[ \mathbf{A} = \text{a generic totally dynamic constraint matrix.} \]

\[ a_i = \text{the } i\text{th row of } \mathbf{A}, \ i = 1, \ldots, m. \]

\[ a_j = \text{the } j\text{th column of } \mathbf{A}, \ j = 1, \ldots, n. \]

\[ A_{ij} = \text{the } i,j\text{th component of } \mathbf{A}. \]

\[ X_j = \text{the } j\text{th variable}, \ j = 1, \ldots, n. \]

\[ C_j = \text{the cost of } X_j. \]

\[ \mathbf{d} = \text{a generic } m \times 1 \text{ nonnegative right hand side demand vector.} \]

We will refer to rows, columns, and variables by their indices when no confusion arises by doing so.

For any row \( i \), let

\[ S(i) = \text{the set of indices of the critical columns for row } i. \]

For any column (variable) \( j \), let

\[ r(j) = \text{the unique row for which column } j \text{ is critical; thus } j \notin S(r(j)) \text{ while } j \notin S(k) \text{ for } k \neq r(j). \]

Let \( T \) be the set of feasible bases of \( \mathbf{A} \) defined in Theorem 3-6. Recall that all bases in this set are feasible for any nonnegative right hand side demand vector. Let \( \Omega \) be a collection of sets defined by \( \Omega = \{ W : W \text{ is a set of } m \text{ column indices such that exactly one element of } S(i), \ i = 1, \ldots, m \text{ is in } \omega \} \). From Theorem 3-6, we see that the set \( T \) and the collection \( \Omega \) are equivalent.

For any basis \( \mathbf{B} \in T \), let
\( c^B = \) the vector of objective function components of basic variables of basis \( B \);

\( \Phi(B,i) = \) the index of the unique column (variable) for row \( i \) in basis \( B \); that is, \( \Phi(B,i) \) is the unique element of \( S(i) \) represented in basis \( B \);

\( n^B = \) the dual vector implied by basis \( B \); that is, \( n^B \) is the unique solution to \( c^B = n^B \).

For any basis \( B \in T \), it follows directly from the definition of \( n^B \) that

\[
\sum_{k=1}^{m} \Pi_k^B A_{k,\Phi(B,i)} = C_{\Phi(B,i)} \quad i = 1, \ldots, m
\]  

(3 - 2).

If \( i = r(j) \), then \( A_{ij} \) is the only positive component of column \( j \). From a result within the proof of Theorem 3-5, we also know that \( A_{ij} = 0, \forall i > r(j) \). Therefore, (3-2) can be reduced to

\[
\sum_{k=1}^{i} \Pi_k^B A_{k,\Phi(B,i)} = C_{\Phi(B,i)} \quad i = 1, \ldots, m
\]  

(3 - 3).

In particular, taking \( i = 1 \) yields

\[
\Pi_1^B A_{1,\Phi(B,1)} = C_{\Phi(B,1)} \quad \text{or} \quad \Pi_1^B = C_{\Phi(B,1)} / A_{1,\Phi(B,1)}
\]  

(3 - 4).

That is, the first component of \( n^B \) is uniquely determined by the first constraint's critical column. For \( i = 2 \), (3-3) yields

\[
\Pi_1^B A_{1,\Phi(B,2)} + \Pi_2^B A_{2,\Phi(B,2)} = C_{\Phi(B,2)}
\]  

(3 - 5).

Since \( \Pi_1^B \) was already determined by (3-4), \( \Pi_2^B \) is obtained from (3-5) by
\[ \Pi_{2}^{B} = (C_{\Phi(B,2)} - \Pi_{1}^{B} A_{1,\Phi(B,2)}) / A_{2,\Phi(B,2)} \]  
\hspace{1cm} (3 - 6) 

and, in general, \( \Pi_{i}^{B} \) can be determined recursively by

\[ \Pi_{i}^{B} = (C_{\Phi(B,i)} - \sum_{k=1}^{i-1} \Pi_{k}^{B} A_{k,\Phi(B,i)}) / A_{i,\Phi(B,i)} \quad i = 1, \ldots, m \]  
\hspace{1cm} (3 - 7) 

Equation (3-7) gives an explicit formula for calculating \( \Pi^{B} \) for feasible basis \( B \). Note that no matrix inversion is involved in the calculation. It is important to observe that \( \Pi_{i}^{B} \) depends on the critical columns of the first \( i \) constraints and on nothing else from basis \( B \).

Recall that our goal is to find an optimal primal solution. Consider any constraint \( k \) and let \( W_{k-1} \) be any set of \( k-1 \) column indices where exactly one element of \( S(i) \), \( i = 1, \ldots, k-1 \), is in \( W_{k-1} \).

For any \( j \in S(k) \), let \( W_{kj} = W_{k-1} + \{j\} \). For any basis \( B \in T \) whose critical columns for the first \( k \) constraints are indexed by the elements of \( W_{kj} \), the first \( k \) components of \( \Pi^{B} \) are uniquely determined by (3-7). Hence let \( \Pi_{k}^{W_{kj}} \) be the unique dual price on constraint \( k \) implied by \( W_{kj} \) for each \( j \in S(k) \), and let \( l \in S(k) \) be any index such that \( \Pi_{k}^{W_{kl}} = \min_{j \in S(k)} \Pi_{k}^{W_{kj}} \). Since \( A_{kj} > 0 \forall j \in S(k) \), it follows that the reduced cost of any variable \( j \in S(k) \) is nonnegative relative to any basis \( B \in T \) whose critical columns for the first \( k \) constraints are indexed by the elements of \( W_{k-l} \).

We define the vector \( \Pi^{*} \) recursively by

\[ \Pi_{i}^{*} = \min_{j \in S(i)} \left[ C_{i} - \sum_{k=1}^{i-1} \Pi_{k}^{*} A_{kj} \right] / A_{ij} \quad i = 1, \ldots, m \]  
\hspace{1cm} (3 - 8) 

and the basis \( B^{*} \in T \) by

\[ \Phi(B^{*},i) = \arg \min \{ j \in S(i) : \Pi_{i}^{*} = \left[ C_{j} - \sum_{k=1}^{i-1} \Pi_{k}^{*} A_{kj} \right] / A_{ij} \} \]  
\hspace{1cm} (3 - 9) 

81
It follows directly from our analysis that \( \pi^* \) is dual feasible, because each primal variable has a nonnegative reduced cost. \( B^* \) is primal feasible because it is an element of \( T \). Moreover, since complementary slackness is maintained by construction and \( B^* \) is triangular, a primal optimal solution can be obtained by

1) employing simple backward substitution on the system \( B^* \mathbf{x}_{B^*} = \mathbf{d} \) to set \( \mathbf{x}_{B^*} \), the vector of basic variables relative to basis \( B^* \);

2) setting the remaining components of \( \mathbf{x} \) to zero.

**Theorem 3.7**: The procedure outlined above determines an optimal dual solution \( \pi^* \) and an optimal primal solution \( \mathbf{x} = [\mathbf{x}_{B^*}, \mathbf{x}_N] = [B^* \mathbf{d}, 0] \) to the totally dynamic system \( \{ \mathbf{x} : \mathbf{Ax} = \mathbf{d}, \mathbf{x} \geq 0 \} \).

Note that this procedure requires each column to be considered exactly once in determining \( \pi^* \) and \( B^* \). Also, no matrix inversion is required anywhere.

For any \( W \in \Omega \), let \( B_W \) be its corresponding element in the set \( T \). (Recall that \( T \) and \( \Omega \) are equivalent.) We define the sets \( R(i), i = 1, \ldots, m \), by

\[
R(i) = \{ j \in S(i) : \pi^*_i = \left[ C_j - \sum_{k=1}^{t-1} \pi^*_k A_{kj} / A_{ij} \right] \}
\]

That is, each set \( R(i) \) contains the critical variables of constraint \( i \) that have zero reduced cost relative to basis \( B^* \). Let \( \Omega^* \subseteq \Omega \) be defined by \( \Omega^* = \{ W : \mathbf{W} \text{ is a set of } m \text{ column indices such that exactly one element of } R(i), i = 1, \ldots, m, \text{ is in } W \} \). For any \( W^1 \in \Omega^* \), it is straightforward to show that \( B_{W^1} \) is a primal optimal basis. For any \( W^2 \in \Omega - \Omega^* \), it must be that \( B_{W^2} \) has at least one
basic variable $X_j$ with positive reduced cost relative to basis $B^*$. Hence if $B_{W^2}$ is primal optimal, the optimal solution it produces must have $X_j = 0$. From the properties of totally dynamic systems, it is straightforward to show that this optimal solution can also be produced by any basis $B_{W^3}$ where $W^3$ is constructed from $W^2$ by simply replacing column $j$ with some column whose index is in $R(\nu(j))$.

This implies that any primal optimal solution that can be generated by an element of $T$ can be generated by a basis whose set of column indices is an element of the collection $\Omega^*$. But, since $T$ can generate every extreme point of the totally dynamic system, it follows that the set of bases corresponding to the collection $\Omega^*$ generates every optimal extreme point solution to the totally dynamic system.

We have seen that for the I-P balance constraint system of (BMRP), the critical variables for a constraint it consist of $P_{i,t-L_i}$ and $L_{i,t-1}$ (disregarding the relatively few "boundary" constraints having less than two critical variables.) Hence for any I-P balance constraint system of a pure production problem, which is totally dynamic, Theorem 3-6 implies that every extreme point of the corresponding feasible region satisfies

$$I_{i,t-1}P_{i,t-L_i} = 0$$

(3-11).

This is a generalization of the well known characterization of extreme points in simple single-item lot sizing (see Wagner and Whitin (1958)); namely,

$$I_{t-1}P_t = 0$$

for the single-item problem with no lead time. Simply stated, in any extreme point solution either the production becoming available for use or the inventory on hand for any item in any period must be zero. It is interesting that this extreme point characterization remains true in a
problem as relatively complex as the uncapacitated pure production version of (BMRP). In Section 3.8 we will see that the result remains true for pure production problems for model (EMRP) as well.

If we relax the facility capacity constraints in our models, the setups variables $X_{it}$ can be eliminated from the problem and the functions

$$\delta_i(P_{it}) = c_i s_i \text{ if } P_{it} > 0$$

$$= 0 \text{ if } P_{it} = 0$$

$\forall i$ can reflect setup "charges" in the objective function. The setup functions $\delta_i$ are concave, and hence the objective function becomes concave with the addition of the $\delta_i$ functions. Therefore, in dealing with a pure production, uncapacitated version of (BMRP), an optimal solution would still exist with the properties of (3-11). That is, some basis as characterized by Theorem 3.6 provides an optimal extreme point solution.

Given the ease with which we can characterize these bases and their special structure (i.e., upper triangular, positive diagonals, nonpositive components above the diagonal), it is tempting to pursue a line of research which aims to develop a special structure algorithm to find an optimal solution over totally dynamic I-P balance systems with setup charges on production variables. The existence of such an algorithm would largely obviate our reservations in Chapter 2 on pursuing a global optimization scheme which was based on relaxing capacity constraints at the highest level of the algorithm. However, as mentioned in Chapter 2, no algorithms have been found (to our knowledge) in the twenty-seven years since Wagner and Whitin's pioneering paper on lot sizing for finding optimal solutions to general uncapacitated multistage problems with setup charges which do not resort to generic mixed-integer programming or very complicated dynamic programming formulations. We have spent some time in search of such an algorithm and have come to appreciate the complexity of the problem.
Perhaps, as mentioned in Chapter 2, the problem is NP-hard or NP-complete. (Actually, if it is NP-hard, it would be NP-complete as well.) We leave this area open as a possible future research topic.

The addition of "buy" variables $B_{it}$ and "backlog" variables $J_{it}$, mentioned in Section 1.2, preserves the relationship of I-P balance constraint systems of (BMRP) to totally dynamic systems. Each of these variables has exactly one positive component in the I-P balance constraint system. If we consider an uncapacitated pure production version of (BMRP) with "make or buy" and "backlogging" options, then it is clear from our analysis that every extreme point of the corresponding feasible region has the property

$$I_{i,t-1} P_{i,t-L_i} B_{it} J_{it} = 0.$$ 

That is, the internally generated and exogenous demand for item $i$ in period $t$ is covered from exactly one of the four sources in the above product.

We note that McClain, Thomas, and Weiss (1982) (MTW) have also worked on special structure algorithms for LPs over the I-P balance constraints of pure production versions of model (BMRP). MTW look at the dual of the LP of this specific problem and derive a problem-specific recursive equation for finding a set of optimal dual prices to the LP. Their approach is fundamentally different from ours. Specifically, MTW develop their algorithm by analyzing directly the properties of the dual LP, whereas we have identified properties of the constraint matrix and bases of the primal problem. Both procedures produce a set of dual optimal prices with a single scan of the columns of the primal formulation. Our approach further reveals that certain dual constraints – namely, those for which the computed optimal dual vector leaves no slackness – define a primal optimal basis which is independent of the demand vector and is upper triangular. Thus a primal optimal solution can be established by simple backward substitution. In addition, we have seen that the pure production version of model (BMRP) is a
special case of totally dynamic systems, and any such system can be solved in an equally
efficient manner because of the characteristics of their feasible bases.
3.7 Totally Dynamic Systems and the Theory of Leontief Substitution Systems

In Section 3.5 we defined Leontief substitution systems and showed that totally dynamic systems were a special case of such systems. In this section we will explore further the relationship of totally dynamic systems to the well developed theory of Leontief substitution systems. In particular, we will see that totally dynamic systems have the characteristics of several particular types of Leontief substitution systems.

Dantzig (1955) addresses the issue of solving LPs over dynamic Leontief models that, as we will see shortly, are closely related to totally dynamic systems. A Leontief matrix $A$ is said to be a dynamic Leontief matrix if $A$ is expressable as

$$
\begin{bmatrix}
A_{11} & \cdots & A_{1m} \\
\vdots & \ddots & \vdots \\
A_{m1} & \cdots & A_{mm}
\end{bmatrix}
$$

where the diagonal submatrices $A_{ii}$ of $A$ are Leontief $\forall \, i = 1, \ldots, m$, while the remaining submatrices of $A$ satisfy

$$
A_{ij} = 0, \forall \, i, j \text{ such that } i > j \text{ and } A_{ij} \leq 0, \forall \, i, j \text{ such that } i < j.
$$

Dantzig is concerned with optimizing linear programs of the form
\[
 v(P) = \min c\mathbf{x} \quad (P)
\]

\[
 s.t. \; A\mathbf{x} = \mathbf{b} \geq 0
\]

\[
 \mathbf{x} \geq 0
\]

where \( A \) is a dynamic Leontief matrix.

Dantzig starts his solution procedure with a feasible solution for an arbitrary right hand side vector \( \mathbf{b}^1 > 0 \). Let \( \mathbf{n} \) be a vector of dual optimal variables for the LP with right hand side \( \mathbf{b}^1 \) and \( \mathbf{n}^i, i = 1, \ldots, m \), the subvector of \( \mathbf{n} \) pertaining to the submatrix \([A_{i1}, A_{i2}, \ldots, A_{im}]\) of \( A \). Dantzig shows how the property \( A_{ij} = 0, \forall i, j \) such that \( i > j \) implies the optimal dual subvectors \( \mathbf{n}^i \), \( i = 1, \ldots, m \), of dual optimal vector \( \mathbf{n} \) can be obtained sequentially. Moreover, this \( \mathbf{n} \) vector is dual optimal for all \( \mathbf{b} \geq 0 \).

Let \( P \) be the set of columns forming the submatrix

\[
 A^i = \begin{bmatrix}
 A_{1i} \\
 \vdots \\
 A_{mi}
\end{bmatrix} \quad i = 1, \ldots, m
\]

of \( A \). The original feasible basis with right hand side \( \mathbf{b}^1 \) can be shown to be feasible \( \forall \mathbf{b} \geq 0 \). Dantzig demonstrates how the optimal subvector \( \mathbf{n}^1 \) can be determined by restricting simplex pivots to the elements of \( P \). Once \( \mathbf{n}^1 \) is determined, \( \mathbf{n}^2 \) can be similarly determined by restricting pivoting to elements of \( P_2 \), and \( \mathbf{n}^1 \) is unaffected by the outcome. The procedure continues until \( \mathbf{n} \) and an optimal primal basis (independent of \( \mathbf{b} \geq 0 \)) is determined.

Thus Dantzig shows that LPs over dynamic Leontief systems can be solved by performing a sequence of restricted simplex pivots. This approach clearly represents an improvement over
the general simplex approach, since columns are partitioned \textit{a priori} into subsets, each of which
can be dealt with independently.

It is clear that the totally dynamic matrices considered in this section are special cases of
Dantzig's dynamic Leontief matrices; specifically, each submatrix \([A_{i1}, \ldots, A_{im}]\) has exactly one
row in totally dynamic matrices. Veinott (1969) mentions such special cases and remarks (but
does not demonstrate) that the optimal dual vector \(\pi\) can be recursively calculated, element by
element, in such systems. This recursion appears in equation (3-7) of Section 3.6 of this
document, which was developed independently of Veinott's observation.

The existence of exactly one row in each diagonal submatrix of totally dynamic systems
implies that LPs over constraints formed by such systems can generally be solved more quickly
than Dantzig's LP problems over dynamic systems. Given an \(m \times n\) totally dynamic matrix,
columns are partitioned into \(m\) sets \(P_i\) by defining \(P_i\) as the set of critical columns for row \(i,\)
\(i = 1, \ldots, m\). The fact that an optimal basis exists with exactly one element chosen from each of
the sets \(P_i\) means that each of the \(m\) subsystems can be solved very quickly.

In Dantzig's more general systems, a set \(P_i\) may encompass a diagonal Leontief matrix with
more than one row. Therefore simplex pivots must be carried out on each of these sets, implying
columns may enter and leave the basis multiple times before an optimal subbasis is found.
Matrix inversion is thus unavoidable to the extent that it is unavoidable in the simplex
algorithm. Moreover, the requirement of a feasible solution for some positive right hand side
demand vector as a starting point in Dantzig's algorithm is not required in the optimization
algorithm for totally dynamic systems of Section 3.6.

We now discuss the second category of special structure Leontief substitution systems of
interest in this section: namely, totally Leontief systems. We owe the discovery of these
systems to Veinott (1968). Veinott establishes that every extreme point of a Leontief
substitution system can be generated by a basis which is Leontief. (It was already known that
every Leontief basis is feasible for every nonnegative right hand side vector, since such
matrices have nonnegative inverses.) We let \(B^*\) be the set of all full, square Leontief
submatrices of a Leontief substitution system. (Note, all such matrices are nonsingular.) We also define $B$ to be the set of all full, square pre-Leontief submatrices of this system which have pre-Leontief transposes (i.e., each row has at most one positive component). Any square Leontief matrix has exactly one positive element in each row and column, and so $B^* \subseteq B$. A Leontief substitution system is called \textit{totally Leontief} if $B = B^*$.

It is straightforward to show that totally dynamic systems are totally Leontief. Recall our definition in Section 3.5 of $T$ as the collection of all sets of columns of a totally dynamic matrix where each set contains exactly one critical column for each row. Since each column of a totally dynamic matrix has exactly one positive component, it is clear that $B = T$. Furthermore, each element of $T$ is Leontief, since each such matrix is upper triangular with a strictly positive diagonal and nonpositive elements above the diagonal. Thus totally dynamic systems are totally Leontief.

Hence Veinott's sets $B^*$ and $B$ are both equivalent to $T$ in the special case of totally dynamic systems. Beyond being totally Leontief, what really sets totally dynamic systems apart is the fact that $B^* = B = T$ can be characterized as the set of full, square, upper triangular matrices with strictly positive diagonals and nonpositive components above the diagonal. This special structure of the generating set $T$ is the reason that LPs over totally dynamic constraint sets can be solved so quickly.

Of course, it is not true that a dynamic Leontief system of Dantzig is necessarily totally Leontief. We have not investigated whether any system which is both dynamic and totally Leontief must be totally dynamic. We venture a speculation that this is not the case.
3.8 The Inventory-Production-Orders Balance Constraint Set of Model (EMRP)

So far, we have limited our attention in this chapter to the properties of the inventory-production balance constraints of (BMRP). In this section we investigate the properties of these constraints when the fixed demands of (BMRP) are generalized to the order variables of model (EMRP). The interesting results of this section are:

1) The inventory-production-order (I-P-O) balance constraint sets are totally dynamic under basically the same conditions that inventory-production balance constraint sets are totally dynamic.

2) If an I-P-O balance constraint set satisfies the totally dynamic conditions, then the order variables are integral in any extreme point of the convex polyhedron formed by the constraint set.

We recall the I-P-O balance constraint set that first appeared as part of our presentation of model (EMRP) in Section 1.2. It is:

\[
I_{t+1} + y_t P_{t+1} - I_t - \sum_{j=1}^{N} a_{ij} P_{jt} = \sum_{z=1}^{Z_{1(t)}} d_{iz} W_{izt} \quad i = 1, \ldots, N \quad t = 1, \ldots, T \tag{3-12}
\]

\[
\sum_{t=1}^{T} W_{izt} = 1 \quad i = 1, \ldots, N \tag{3-13}
\]

to which we add the appropriate nonnegativity constraints

\[
I_{it}, P_{it}, W_{izt} \geq 0 \tag{3-14}
\]
We recast (3-12,13,14) in matrix-vector notation as

\[ Ap + Dv - Ew = 0 \quad (3-15) \]

\[ Fw = 1 \quad (3-16) \]

\[ p,v,w \geq 0 \quad (3-17) \]

where (3-15,16) correspond to (3-12,13).

We begin by assuming a problem with no initial inventory and no holdover production in model (EMRP). In Section 3.4 we saw that these conditions, combined with the condition that there is no exogenous demand for any item through its cumulative lead time, imply that all production and inventory variables through the cumulative lead times of all items are constrained to zero in the I-P balance system of model (BMRP). We called such problems pure production problems. We saw that such problems are feasible, because the resultant system, obtained from eliminating unnecessary constraints and variables preconstrained to zero, is totally dynamic, and totally dynamic systems are always feasible.

To obtain the parallel situation in the (EMRP) model with orders, we must, in addition to assuming no initial inventory and holdover production, preconstrain to zero all order variables \( W_{iLt} \) with \( 1 \leq t \leq L_i^{*} \). Once this is done, the resultant system obtained from eliminating production, inventory, and order variables throughout all cumulative lead times is totally dynamic. This result is obtained directly from the following theorem.

THEOREM 3-8: Consider systems of the form

\[ X = \left\{ x,w : Bx - Ew = 0 ; Fw = 1 ; x,w \geq 0 \right\} \]
where

1) \( w \) is partitioned as \( w = [w^1, w^2, ..., w^n] \);

2) \( F \) is an \( n \times |w| \) matrix and the constraint set \( Fw = 1 \) is of the form

\[
\sum_{i=1}^{|w|} w_i^j = 1 \quad j = 1, ..., n;
\]

3) \( E \) is a nonnegative matrix and \( B \) is a totally dynamic matrix.

Such systems are totally dynamic.

**PROOF:** The five properties of totally dynamic systems (as outlined in Section 3.3) can be verified immediately by writing the constraint system defining \( X \) in the form

\[
\begin{bmatrix}
B & -E^1 & -E^2 & ... & -E^n \\
0 & F^1 & 0 & ... & 0 \\
0 & 0 & F^2 & 0 & ... & 0 \\
& & & & & \\
& & & & & \\
0 & 0 & 0 & ... & F^n \\
\end{bmatrix}
\begin{bmatrix}
x \\
w^1 \\
w^2 \\
\end{bmatrix}
= \begin{bmatrix}
0 \\
1 \\
\end{bmatrix}
\]

with

\[
E = [E^1 \ E^2 \ ... \ E^n] \quad \text{and} \quad F = \begin{bmatrix}
F^1 & 0 & ... & 0 \\
0 & F^2 & 0 & ... & 0 \\
& & & & & \\
& & & & & \\
& & & & & \\
0 & ... & F^n \\
\end{bmatrix}
\]
partitioned column-wise in correspondence to the partitioning of \( w \). Note that each submatrix \( F_j \) is 1 x \( |w| \) and has +1's in all components.

Returning to our pure production problems with orders, it is clear that such problems generate constraint systems that fall within the types of systems addressed in Theorem 3-8. These constraint systems are developed from the I-P-O balance system by removing all constraints corresponding to the cumulative lead time of all items, plus presetting the variables \( l_{it}, p_{it}, w_{it}, \forall i, 1 \leq t \leq L_t^* \) to zero. The resultant constraint matrix for inventory and production variables corresponds to matrix \( B \) in the theorem. Section 3.4 has demonstrated that such matrices are totally dynamic. The correspondence between the resultant constraint matrices for the order variables and matrices \( E \) and \( F \) is apparent.

Also of interest is the following:

**THEOREM 3-9:** In any system of the form outlined in Theorem 3-8, the extreme points of the corresponding feasible region have integral \( w \) vectors. More specifically, the full set of extreme points can be generated by:

1) Taking each \( w_j \) \( j = 1, ..., n \) and setting exactly one component \( w_j \) to one and all others to zero;

2) For each of the \( |w^1| \times |w^2| \times ... \times |w^n| \) vectors that can be created by 1, set \( x \) in all possible ways corresponding to the complete set of extreme points of the totally dynamic system

\[
\begin{bmatrix}
  x : Bx = Ew, x \geq 0
\end{bmatrix}
\]
by employing the generating set of bases outlined in Theorem 3-6.

PROOF: The result is apparent when we consider Theorem 3-8 which established that such systems are totally dynamic and Theorem 3-6 which established a set of bases capable of generating all extreme points. The generating set of bases leads directly to the characterization of extreme points outlined in the theorem. □

It is clear that the presence of initial inventory and/or holdover production destroys the totally dynamic nature of I-P-O balance systems, because their existence creates demand vectors with negative components. Moreover, if initial inventory and holdover production are zero, it is straightforward to show that any order variable \( W_{ist} \) with \( 1 \leq t \leq L_i^* \) must be zero in any feasible solution. This analogous situation in model (BMRP) is that exogenous demand must be zero for all items through their cumulative lead times if there is no initial inventory and holdover production, since otherwise the problem is infeasible.

Define \( X-I \) to be systems similar to the systems \( X \) of Theorem 3-8, with the exception that the \( W_{ist} \) variables are now constrained to be 0-1. Theorem 3-9 implies that any minimization problem with a concave objective function and \( X-I \) defining the feasible region can be solved by relaxing the integrality restrictions on the \( W_{ist} \) variables and selecting an extreme point of the relaxed system \( (X) \) which is minimal among the set of \( X \)'s extreme points. This is due to the fact that \( [X-I]^C = X \) where \( [\cdot]^C \) represents the convex hull of a set. Specifically, LPs over \( X-I \) can be solved by replacing \( X-I \) with \( X \) and using the totally dynamic constraint set algorithm of Section 3.6.
3.9 Solving LPs Over General Inventory-Production Balance Constraints

This section addresses issues raised by the fact that I-P balance constraint sets generally violate the properties of totally dynamic systems. However, the results of this chapter show that, in most practical cases, a subset of the I-P balance constraint set encompassing most of the constraints form a totally dynamic system. The question then is: can this fact be exploited to develop algorithms for solving LPs over general I-P balance systems encompassing initial inventories and holdover production (and hence potentially negative demands) through some initial number of periods?

We first return to the global problem (BMRP) and the nested decomposition strategy outlined in Chapter 2 for a little perspective. Recall our intention of solving the Benders subproblem via dual decomposition to isolate the I-P balance constraints, which were assumed to have an exploitable special structure. It is clear that when only a subset of the I-P balance system has special structure (i.e., forms a totally dynamic system), the remaining I-P balance constraints can simply be priced out along with capacity constraints in the Lagrangian relaxation of the Benders subproblem. This is certainly a viable alternative. The intent of this section, however, is to outline an approach which keeps all I-P balance constraints in the Lagrangian subproblem.

We now develop an impetus for this approach. Consider any LP

\[ \nu(P) = \min \ c^1 x^1 + c^2 x^2 \quad \text{(P)} \]

\[ A^1 x^1 + A^2 x^2 = d \]

\[ Bx^1 = b \]

\[ x^1, x^2 \geq 0 \]
such that

\[ d - A^1 x^1 \geq 0 \quad \forall x^1 \in X = \{ x^1 : B x^1 = b ; x^1 \geq 0 \} \]

and \( A^2 \) is a Leontief matrix. If we employ resource-directive decomposition to this problem and set the vector \( x^1 \) in the master problem, the resulting subproblem is

\[ \nu(SP) = \min \ c^2 x^2 \quad (SP) \]

\[ s.t. \ A^2 x^2 = d - A^1 x^1 \]

\[ x^2 \geq 0. \]

The linear program \( (SP) \) is an optimization problem over a Leontief substitution system \( \forall x^1 \in X \) because of our assumptions. Thus there is a dual vector \( n^* \) which is optimal in the dual of \( (SP) \) for any \( x^1 \in X \). The goal of a feasible subproblem in this decomposition scheme is to form a cut for the master problem. (Note the subproblem is always feasible in this problem.) But \( n^* \) is dual optimal in every subproblem, implying that \( (P) \) can be solved by solving the LP

\[ \nu(MP) = \min \ c^1 x^1 + n^* (d - A^1 x^1) = n^* d + \min (c^1 - n^* A^1)x^1 \quad (MP) \]

\[ B x^1 = b \]

\[ x^1 \geq 0. \]

Thus the usual decomposition requirement of iteratively solving a sequence of master and subproblems is unnecessary here. The vector \( n^* \) is found by assigning any \( x^1 \in X \) in \( (SP) \) and
using any optimal solution to the dual of (SP) for \( r^* \). The optimization problem (MP) is then solved as an LP to achieve global optimality.

Hence if we can isolate a totally dynamic subproblem and a feasible master problem from the I-P balance constraint system by resource-directive decomposition, we can solve LPs over I-P balance systems by

1) Solving a single LP (generally over a majority of the constraints) by the totally dynamic constraint set algorithm,

2) Solving a single generic LP over the (generally relatively few) remaining constraints.

The question is: how do we partition the set of variables? That is, which variables will be preset in the master problem and which will become subproblem variables? We recall our simple example of an I-P balance system in Figure 3-5. We immediately notice that all variables with nonzero coefficients in type 1 constraints (see Section 3.3) must be in the master problem. If this is not the case, then the subproblem will contain constraints containing no variables with positive coefficients, and this is a violation of the properties of totally dynamic systems. Thus, for example, all variables with nonzero coefficients in constraint \( it = 21 \) must be in the master problem. This implies variable \( l_{21} \) is in the master problem. But since \( l_{21} \) is the only critical variable of constraint 22, all variables with nonzero coefficients in constraint 22 (and hence constraint 22) must be in the master problem for the exact same reason. The recursive logic developing here may sound familiar. In fact, we are on a similar track as when we sought the conditions under which constraints violating the totally dynamic properties could be eliminated by inspection without affecting the problem's feasible region. (See Section 3.4.) Following this logic to fruition, it again becomes clear that a necessary condition for a totally dynamic subproblem is that all variables with nonzero coefficients in constraints of the form \( it, 1 \leq t \leq L_i^* \), must be in the master problem. Call this set of variables \( S \), and define \( C \) to be the set
of constraints \( it: 1 \leq t \leq L_i^* \). (We note that first period constraints of items \( i \) with \( L_i = 0 \) can also have negative net demand because of initial stocks. This situation is somewhat of a special case, because such constraints simultaneously receive production and have negative net demands. In the analysis that follows, we will ignore such cases. That is, we assume the net demand \( r_{11} \) is nonnegative for all constraints \( 1 \) where \( L_i = 0 \). The analysis can be readily extended to include cases with \( r_{00} < 0 \) and \( L_i = 0 \), but the notation required is relatively ponderous. Hence our negligence of these cases is strictly to facilitate the ease of exposition.)

Of course, demands must be nonnegative in the totally dynamic subproblem. It is straightforward to show that the only elements of \( S \) that have positive coefficients in the complement of \( C \) (\( C^c \)) are the variables \( I_{i,L_i^*} \) \( \forall i \). To avoid nonnegative demands in the subproblem as currently posed, we must require \( I_{i,L_i^*} \leq d_{i,L_i^*+1} \) \( \forall i \). Such a restriction alters the problem's feasible region, implying the potential loss of global optimality. However, within the scope of "good" feasible solutions, these constraints may not be overly restrictive.

Unfortunately, even this restriction is not necessarily enough to achieve our goal. An additional complication is that the bounding of the inventory variables which straddle the constraint set partition may lead to an infeasible master problem. This is illustrated by the simple example of Figure 3-6 in which the elements of \( C \) are marked with arrows. The bounding of the variable \( I_{11} \leq 2 \) implies \( P_{31} \geq 5 \). This in turn creates a demand of 25 for item 2 in period 1 and hence an infeasible master problem. In general, the bounding of inventory can "force" production of items upstream in the product structure relatively early in the planning horizon and may, as we have seen, lead to infeasibilities. This forced production is also undesirable, from the perspective of inventory holding costs, to the extent that it is more costly to stock items that are further upstream in the product structure.

In response, we develop a procedure that always creates a constraint set partition with a feasible master problem and a totally dynamic subproblem. Moreover, the master problem will not artificially force production in the manner just described. Define
\[ D_{it} = \sum_{m=1}^{t} r_{im} \quad \forall i, t: 1 \leq t \leq T \]

to be the cumulative net demand for an item where \( r_{it} \) is the net demand for \( i \) in period \( t \). Let \( S_i \) and \( P_i \) be the sets of immediate successors and predecessors of \( i \) respectively. We also define

\[ C_{it} = D_{it} \quad \text{if } i \text{ is a finished good (} S_i = \emptyset \) \]

\[ = D_{it} + \sum_{j \in S_i} [a_{ij} \max \{0, C_{jd} / y_j \} / y_j] \quad \text{otherwise} \quad (3-18). \]
Figure 3-5 Inventory-Production Balance Constraints for a Simple Product Structure
| i=1, | t=1 | -1 | -2 | -12 ↔ |
|      | t=2 | 1  | y₁  | -1  | 2  |
|      | t=3 | 1  | y₁  | -1  | 0  |
|      | t=4 | 1  | y₁  |      | 5  |

| i=2, | t=1 | -1          | -5 | -10 ↔ |
|      | t=2 | 1  | -1  | -5  | 3  |
|      | t=3 | 1  | y₂  | -1  | 0  |
|      | t=4 | 1  | y₂  |      | 10 |

| i=3, | t=1 | -1          | 0  | ↔   |
|      | t=2 | 1  | -1  | 5  | ↔   |
|      | t=3 | 1  | y₃  | -1  | 5  | ↔   |
|      | t=4 | 1  | y₃  | 15  | ↔   |

<table>
<thead>
<tr>
<th>level</th>
<th>Product Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>1 2</td>
</tr>
</tbody>
</table>

*a₁₃ = 2  \quad L₁ = 1   \quad a₂₃ = 5  \quad L₂ = L₃ = 2*

**Figure 3-6 An Infeasible Master Problem**
$C_{it}$ consists, in part, of the cumulative net demand $D_{it}$ for $i$. Moreover, consider some (not necessarily immediate) successor $j$ of $i$ that is unable to meet a demand placed on it through holdover production or initial stock. The unmet demand requires production of $j$, which in turn puts additional internally generated demand on all predecessors of $j$. The accumulation of all such internally generated demand on $i$ through period $t$ is reflected in $C_{it}$ through the summation over $S_{ij}$.

If $C_{it}$ is positive for some $1 \leq t \leq L_i$, then the I-P balance constraint system is infeasible, since production of $i$ is required to meet a demand within the minimum lead time, and this is impossible. It is interesting to note that if

$$C_{it} \leq 0 \quad \forall i, t : 1 \leq t \leq L_i \quad (3-19)$$

then the I-P balance system is feasible. This latter fact can be verified recursively by starting at the finished goods level of the product structure and working downstream in the product structure. The relationship in (3-19) ensures that all demands imposed on $i$ within its minimum lead time can be satisfied by holdover production and initial stocks. Beyond the minimum lead time ($t > L_i$), there is time to produce $i$ to meet demands, assuming all immediate predecessors can be made available. But since (3-19) holds for all items, the recursive relationship implies immediate predecessors can be made available as needed. Thus (3-19) represents succinctly the necessary and sufficient conditions for the feasibility of a generic I-P balance constraint system of model (BMRP).

**Theorem 3-10:** The I-P balance system of (BMRP) is feasible if and only if

$$C_{it} \leq 0 \quad \forall i, t : 1 \leq t \leq L_i \quad \Box$$
We define $OH_{it} \equiv -C_{it} \forall i,t$. We think of $OH_{it}$ as being the "on-hand" stock of $i$ in period $t$. The quantity $OH_{it}$ can be either negative or positive. It is positive if and only if the initial stock and cumulative holdover production received through $t$ exceed the cumulative total demand (exogenous and internally generated) for $i$ through $t$. As a function of $t$, $OH_{it}$ has no monotonic properties through period $L_i$ because of the possibility of receiving holdover production through the minimum lead time. It is, however, monotonically nonincreasing thereafter. We plot a sample of $OH_{it}$ as a function of time in Figure 3-7.

![Figure 3-7: $OH_{it}$ as a Function of $t$](image)

Let $r_i$ be the latest period (possibly zero) in which $OH_{it}$ is positive. Hence period $r_i \cdot L_i + 1$ is the earliest period in which production of $i$ must be initiated to meet demands for $i$, and thus any production initiated earlier will create stock of $i$ and/or one of $i$'s successors.
We assume $r_i \geq L_i$, which (via the definition of $r_i$ and Theorem 3-10) are the necessary and sufficient conditions for which the I-P balance system is feasible. We define

$$N_i = r_i \text{ if } P_i = \emptyset$$

$$= \max \{ r_i, \max_{j \in P_i} (N_j + L_j) \} \text{ otherwise.}$$

We use $N_i$ to form a variable set and constraint set partition that establishes the desired master and subproblems:

**Master and Subproblem Formulation**

1. Put all constraints $it, 1 \leq t \leq \min \{ T, N_i \}$, in the master problem constraint set $C$ and all remaining constraints in $C$'s complement, $C^c$.

2. Put all variables with some nonzero coefficient in an element of $C$ in the master problem variable set $S$ and all remaining variables in $S$'s complement, $S^c$.

3. If $N_i < T$, then bound the master problem variables $I_i, N_i$ above by $d_i, N_i + 1 \forall i$.

Observe that $N_i \geq L_i \forall i$, so the decomposition meets the necessary conditions for a totally dynamic subproblem previously outlined.

**THEOREM 3-11:** $P_{it} \in S^c \forall i, t : N_i - L_i + 1 \leq t \leq T - L_i, I_{it} \in S^c \forall i, t : N_i + 1 \leq t \leq T - 1$.

**PROOF:** We need to show that for any $i$ with $N_i < T$, any variable $P_{it}$ with $N_i - L_i + 1 \leq t \leq T - L_i$ has all of its nonzero coefficients in elements of $C^c$. For any $j \in P_i$, $P_{it}$ has a negative component $-a_{ij}$ in constraint $jt$. Note that
\[ t > N_t - L_t \geq N_j + L_i - L_i = N_j \]

where the second inequality is from the definition of \( N_t \). Hence all negative components of \( P_{it} \)'s column are in elements of \( C^c \). \( P_{it} \)'s positive component \( y_i \) is in constraint \( i, t + L_i \) and

\[ t + L_i > N_t - L_i + L_i = N_t, \]

and therefore \( i, t + L_i \) is also in \( C^c \). The second part of the theorem is apparent and the proof is complete. \( \square \)

**THEOREM 3.12:** The subproblem created by the above formulation is totally dynamic.

**PROOF:** \( d_{it} \) is nonnegative for each \( it \in C^c \) where \( T \geq t > N_t + 1 \) because \( N_t \geq L_t \). Furthermore, \( d_{i,N_i + 1} - I_{i,N_i} \geq 0 \) \( \forall i \) such that \( N_i + 1 \leq T \), implying the demand vector of the subproblem is nonnegative. Theorem 3.11 ensures that all critical variables of all elements of \( C^c \), with the exception of constraints \( i, N_i + 1 \) \( \forall i \) such that \( N_i + 1 \leq T \), are in the subproblem. For this latter group of constraints, the critical variables \( I_{i,N_i} \) have been placed in the master problem, but Theorem 3.11 shows that the critical variables \( P_{i,N_i,L_i} + 1 \) remain in the subproblem. These facts combined with a straightforward variation of Theorem 3.1 yield the result. \( \square \)

**THEOREM 3.13:** The master problem created by the above formulation is feasible.

**PROOF:** We establish the theorem by showing that the following solution is feasible:

\[
P_{it} = 0 \quad \forall i,t: 1 \leq t \leq t_i \cdot L_t
\]

\[
= C_{i,t + 1} / y_i \quad \forall i,t: t = t_i \cdot L_t + 1
\]

\[
= (C_{t + 1,t} - C_{t + L_t - 1,t}) / y_i \quad \forall i,t: t = t_i \cdot L_t + 2, ..., N_t \cdot L_t
\]
\[ I_{it} = OH_{it} \quad \forall i, t: 1 \leq t \leq \tau_i \]
\[ = 0 \quad \forall i, t: \tau_i + 1 \leq t \leq N_i. \]

The proposed solution is relatively easy to interpret: \( P_{it} \) is zero through the first \( \tau_i \cdot L_i \) periods because the demand \( C_{it} \) for \( i \) can be met by initial stock and holdover production during the time interval in which the production from these variables becomes available (i.e., periods \( L_i + 1 \) through \( \tau_i \)). The remaining production variables \( P_{it} \) in the master problem produce exactly what is needed to meet each period's outstanding exogenous and internally generated demand for \( i \) in periods \( P_{, t+L_i} \). Simply stated, the proposed master problem solution does not produce anything until it is absolutely necessary, and then it meets demands by producing "just-in-time."

Since \( C_{it} \leq 0 \) for \( t = 1, \ldots, \tau_i \) and \( C_{it} > 0 \) and is monotonically nondecreasing for \( t = \tau_i + 1, \ldots, T \), all variables in the proposed solution are set nonnegatively. Also, the bounds on the inventory variables that straddle the master and subproblem constraints are clearly met. We must, then, verify that the solution satisfies the I-P balance equations

\[ I_{t,t-1} - I_{it} - \sum_{j \in S_i} a_j P_{jt} + y_j P_{t,t-L_i} = d_{it} \quad \forall i, t: 1 \leq t \leq N_i \quad (3-20). \]

For any \( j \in S_i \) in (3-20) we have

\[ P_{jt} = 0 = [0 - 0]/y_j = \max \{ 0, C_{j,t+L_j} - \max \{ 0, C_{j,t+L_j-1} \} \}/y_j \quad 1 \leq t \leq \tau_j - L_j \]

\[ P_{jt} = C_{j,t+L_j}/y_j = [C_{j,t+L_j} - 0]/y_j = \max \{ 0, C_{j,t+L_j} - \max \{ 0, C_{j,t+L_j-1} \} \}/y_j \quad t = \tau_j - L_j + 1 \]

and
\[ P_{jt} = (C_{j,t+L_j} - C_{j,t+L_j-1}) / y_j = \]

\[ [\max\{0, C_{j,t+L_j}\} - \max\{0, C_{j,t+L_j-1}\}] / y_j \quad \forall j, t: 1 \leq t \leq N_j - L_j. \]

Therefore

\[ P_{jt} = [\max\{0, C_{j,t+L_j}\} - \max\{0, C_{j,t+L_j-1}\}] / y_j \quad \forall j, t: 1 \leq t \leq N_j - L_j \quad (3.21). \]

Observe that \( N_j - L_j \geq N_i \quad \forall j \in S_i \). Hence \( P_{jt} \) has the value indicated in (3.21) in every constraint of constraint set (3.20). Now, for item \( i \) in (3.20) we have

\[ I_{t,t-1} - I_{it} = OH_{t,t-1} - OH_{it} = C_{it} - C_{i,t-1} = \]

\[ r_{it} + \sum_{j \in S_i} a_{ij} (\max\{0, C_{j,t+L_j}\} - \max\{0, C_{j,t+L_j-1}\}) / y_j \quad 1 \leq t \leq \tau_i \]

whereas

\[ I_{t,t-1} - I_{it} = OH_{t,\tau_i} = -D_{t,\tau_i} - \sum_{j \in S_i} a_{ij} \max\{0, C_{j,t+L_j}\} / y_j \quad t = \tau_i + 1 \]

\[ I_{t,t-1} - I_{it} = 0 \quad \tau_i + 2 \leq t \leq N_i. \]

Finally,

\[ y_i P_{it,t-L_i} = d_{it} - r_{it} \quad 1 \leq t \leq L_i \]

\[ y_i P_{it,t-L_i} = 0 \quad L_i + 1 \leq t \leq \tau_i \]
\[ y_i P_{i,t-L_i} = C_{i,t+1} = D_{i,t} + 1 + \sum_{j \in S_i} a_{ij} \max\{0, C_{j,t+L_j+1}/y_j\} \quad t = t_i + 1 \]

and

\[ y_i P_{i,t-L_i} = C_{it} - C_{i,t-1} = 
\]

\[ r_{it} + \sum_{j \in S_i} a_{ij} \left( \max\{0, C_{j,t+L_j}\} - \max\{0, C_{j,t+L_j+L_j-1}\} \right) / y_j \quad t_i + 2 \leq t \leq N_i \]

We note that \( r_{it} = d_{it} \forall i,t \) such that \( L_i + 1 \leq t \leq T \). We leave it to the reader to verify that the sum of the values of the three functions \( I_{i,t-1} - I_{it} \cdot \Sigma a_{ij} P_{jt} \cdot \) and \( y_i P_{i,t-L_i} \) equals \( d_{it} \forall i,t \) such that \( 1 \leq t \leq N_i \). This completes the proof of the theorem. \( \square \)

It may be helpful to summarize our progress: Infeasibility in the master problem is possible because bounding the inventory variables in the last period constraints of the master problem may force a demand for some item \( i \) in a period \( t, 1 \leq t \leq L_i \), that is greater than the on-hand stock of \( i \) at time \( t \). Since \( t \) is so early in the planning horizon, this demand cannot be met by producing \( i \). Moreover, forced production generally tends to create stock upstream in the product structure, and this is often undesirable.

This situation is alleviated by incorporating at least the first \( t_i \) constraints in the master problem \( \forall i \). The fact that \( N_i \geq t_i \) constraints are actually in the master problem is a result of the fact that \( t_i + L_i \) can exceed \( t_i \) for some \( j \in P_i \), and hence \( N_i \) must replace \( t_i \) to ensure a totally dynamic subproblem. The variable \( t_i \) is a demarcation point indicating when initial stocks and holdover production are used up by demand for \( i \). By setting up the partition of variables in the prescribed manner, the necessity of forcing production upstream before it is needed to meet
demand is avoided, since the option of letting stocks dwindle and then producing lot-for-lot (i.e., just-in-time) is a feasible master problem option.

If it is always cheaper to stock downstream, which is reflected mathematically by

\[ h_i \geq \sum_{j \in P_i} a_{ji} h_j \quad \forall i, \]

and production costs are constant, then the master problem solution outlined above is optimal. Moreover, under these conditions, the optimal global solution to the LP over the I-P balance system is to produce lot-for-lot, and mathematical programming is not required. In the more general situation, one also has the option of accepting the lot-for-lot solution to the master problem, if it seems a viable option, and thus optimizing only over the totally dynamic subproblem.

We mention that the criteria leading to the master, subproblem partition is strictly a function of the demands \( d_{it} \). Thus, in an optimization problem in which constraints are relaxed, and the I-P balance constraint set forms the feasible region of the corresponding Lagrangian subproblem, the partition outlined in this section remains valid for all dual price vectors on the relaxed constraints. Thus we are assured that the feasible region of the Lagrangian subproblem remains the same if we choose to solve the Lagrangian subproblem by the resource-directive decomposition outlined in this section.

We can use the quantities \( C_{it} \), defined in (3.18), to set the components of the diagonal matrix \( Q \) in the fixed charge constraints of model (BMRP). Assume that we want to set the diagonal components of \( Q \) so that \( p \leq q = Q l \) for every vector \((p, v)\) that is feasible in the I-P balance constraint set. (The reason that \( q \) should be at least this large will be made clear in Chapter 4.) However, for reasons outlined in Chapter 2, we would also like each component of \( q \) to be as small as possible. Recall that \( C_{it} \) is the amount of item \( i \) required by the end of period \( t \). Taking into account the minimum lead time \( L_i \), it follows that
\[ C_{iT} / y_i = \sum_{t=1}^{T-L_i} P_{it} \quad \forall \ i \quad (3-22). \]

That is, \( C_{iT} / y_i \) is the aggregate production quantity of \( i \) initiated throughout the planning horizon. It is also true that

\[ \sum_{k=1}^{t-1} P_{ik} \geq C_{i,t+L_i-1} / y_i \quad \forall \ i,t : 1 \leq t \leq T - L_i \quad (3-23), \]

since otherwise the solution has a negative production variable and/or does not meet some demand, exogenous or internally generated, for \( i \). It follows that

\[ \sum_{k=t}^{T-L_i} P_{ik} \leq (C_{iT} - C_{i,t+L_i-1}) / y_i \quad \forall \ i,t : 1 \leq t \leq T - L_i \quad (3-24), \]

and in particular

\[ P_{it} \leq (C_{iT} - C_{i,t+L_i-1}) / y_i \quad \forall \ i,t : 1 \leq t \leq T - L_i \quad (3-25). \]

Therefore, (3-25) provides us with the bound we seek that will be satisfied by every feasible solution to the inventory-production balance constraint set. These values can thus be used for the diagonal components of matrix \( Q \).
3.10 Alternate Methodologies for Solving LPs Over General Inventory-Production Balance Constraints

Having carried out an analysis of a resource-directive methodology for solving LPs over general I-P balance constraints in the last section, we now mention some problems with this approach and outline some ideas for alternate methodologies. One problem with our resource-directive approach is that the inventory flows \( I_{i,N_i} \) must be bounded above by \( d_{i,N_i+1} \) \( \forall i \). These bounds are quite artificial in that they don't reflect any discernible physical limitations in the manufacturing environment. In essence, these bounds are an artifice for making it easier to obtain feasible schedules. In production environments that are heavily capacitated, it may be the case that most (or all) good schedules call for certain items (i.e., those produced on heavily constrained facilities) to be manufactured early and possibly carried as inventory. The inventory bounds imposed on \( I_{i,N_i} \) \( \forall i \) may prevent such schedules from being realized.

Moreover, the parameters \( N_i \) depend on the parameters \( \tau_i \), which in turn depend on initial stocks, holdover production, and a known schedule for exogenous demands for all items in all periods. Our analysis so far has been carried out for model (BMRP) in which demands are assumed to be known in each period. However, such a schedule is not known for model (EMRP), since it is the responsibility of (EMRP) to determine this schedule by deciding when orders should be shipped. Hence our resource-directive analysis is not directly applicable to (EMRP).

We now broadly outline some ideas on alternate approaches to the problem of solving LPs over general I-P balance constraint systems (that is, constraint systems whose feasible region is not readily characterized by a totally dynamic system because of the presence of initial inventories and/or holdover production). It is not our intention to analyze these alternatives in detail in this work, because this subject is largely an area for future research.

Let \( T_i^* = \{1, ..., L_i\} \). For each item \( i \) in the product structure, let
\[ \sigma_i = \arg \max_t \left\{ t \in T_i^* : P_{i,t-L_i} > 0 \right\} \text{ if } P_{i,t-L_i} > 0 \text{ for some } t \in T_i^* \]

otherwise set \( \sigma_i = 0 \).

Let

\[ \delta_i = 1 \text{ if } I_{i0} > 0 \]

\[ = 0 \text{ otherwise}, \]

and

\[ \beta_i = \max \{ \sigma_i, \beta_i \} \]

for each item \( i \). That is, \( \beta_i \) is the latest time period that receives holdover production or initial inventory of item \( i \). Hence \( \beta_i = 0 \) if there is no holdover production or initial inventory present. However, for ease of exposition, we assume \( \beta_i > 0 \ \forall \ i \). Let

\[ \alpha_i = \beta_i \text{ if } P_i = \emptyset \]

\[ = \max \{ \beta_i, \max_{j \in P_i} \beta_j + L_i \} \text{ otherwise.} \]

Now, instead of the \( N_i \)s of the last section – which were defined in a manner similar to the \( \alpha_i \)s – we let the \( \alpha_i \)s define a partition of the set of variables of the I-P balance constraint system. Specifically, we put all constraints \( it \) with \( 1 \leq t \leq \min \{ \alpha_i, T \} \) in constraint set \( A \), and all variables with a nonzero coefficient in any constraint of constraint set \( A \) in variable set \( A \). The remaining constraints and variables go in constraint set \( B \) and variable set \( B \) respectively. (Note that \( \alpha_i \leq N_i \ \forall \ i \), and, given the manner in which the two parameters are defined, \( \alpha_i \) can be
significantly smaller than \( N_i \).) By definition, any variable with nonzero coefficients in both constraint sets \( A \) and \( B \) is an element of variable set \( A \). It is relatively straightforward to show that any production variable \( P_{it} \) with \( t + L_i \geq \alpha_i + 1 \) is in variable set \( B \). In addition, all other production variables are in variable set \( A \) and have no nonzero coefficients in constraint set \( B \).

Moreover, the only variables of variable set \( A \) with nonzero coefficients in constraint set \( B \) are the inventory variables \( I_{i,a} \). Thus we see that the parameters \( \alpha_i \) ensure a clean partition of the I-P balance constraints and variables. That is, with the exception of the inventory variables \( I_{i,a} \), each variable has all its nonzero coefficients in either constraint set \( A \) or \( B \).

Since the two constraint sets have only the "carry over" inventory variable \( I_{i,a} \) in common, the issue is what to do about them. Our intention here is to "decouple" the inventory flow out of constraint set \( A \), which is represented by the variables \( I_{i,a} \), from the inventory flow into constraint set \( B \). We do this by replacing \( I_{i,a} \) by the new variable \( I_i^A \) in constraint set \( A \) and replacing \( I_{i,a} \) by another new variable \( I_i^B \) in constraint set \( B \). To "preserve" the feasible region, we add the constraints

\[
I_i^A - I_i^B = 0 \quad \forall i. \quad (3-26).
\]

These constraints stipulate that the inventory flow out of constraint set \( A \) must equal the flow into constraint set \( B \) for every item. (Please refer to Figure 3-8.) If we relax the constraint set (3-26), then constraint sets \( A \) and \( B \) are decoupled and completely independent. (See Figure 3-9.) We think of constraint set \( A \) as representing a "transient" problem. That is, constraint set \( A \) incorporates the part of the optimization problem that must deal with the effects of initial stocks and holdover production. In contrast, constraint set \( B \) represents a "pure production" problem. Namely, constraint set \( B \) corresponds to the part of the planning horizon in which the production becoming available for use originated from scheduling decisions made during the planning horizon.
Figure 3-8: In the global problem, \( I_i^A = I_i^B = I_{i,a}, \forall i \).

Note the difference between this approach and the approach of the last section. In the scheme of the last section, the inventory flows from the resource-directive master problem (which corresponds to constraint set \( A \) in this section) serve as direct input to the resource-directive subproblem (which corresponds to constraint set \( B \) in this section). We have noted the limitations that must be placed on the inventory flows in this resource-directive scheme to assure a totally dynamic subproblem. In an attempt to alleviate this restriction, we have
decoupled the inventory flows between the two problems. Because of the decoupling mechanism, the pure production problem is always directly representable by a totally dynamic system. In this scheme, however, it is up to some price-adjusting mechanism to attempt to set prices on the constraints of (3-26) so that the outflow and inflow are equal. In essence, we have migrated from a resource-directive scheme to a price-directive scheme as a mechanism for
dealing with the flow of inventories between the transient problem and the pure production problem.

Because of the decoupling scheme, we are able to redefine the parameters (i.e., we use the \( \alpha \)'s instead of the \( N_i \)'s) that define the separation of the transient problem from the pure production problem. In the decoupling scheme we do not have to wait for on-hand stocks to dwindle to zero before the "pure production" phase begins, as is the case in the resource-directive scheme. Since the \( \alpha \)'s are generally smaller than the \( N_i \)'s, more of the constraints are retained in general in the pure production problem. This is certainly desirable, because the pure production problem is representable as a totally dynamic system and it can be solved very efficiently. It can be shown that the transient problem represented by constraint set \( A \) can be formulated as a generalized linear network flow problem, and this structure may be worth exploiting. (See Steinberg and Napier (1980) for a discussion of the relationship of multistage production problems to generalized linear network models.) Also of importance is the fact that the parameters \( \alpha \) depend only on initial inventories and holdover production. This implies that the decoupling scheme can be readily extended to the inventory-production-orders balance constraints of model (EMRP). We will not provide the details of this extension here.

Recall that Chapter 2 outlined a price-directive decomposition methodology for isolating the I-P balance system in a Lagrangian subproblem by relaxing the facility capacity constraints. This methodology was proposed for the Benders subproblem that resulted from predetermining production setups. Clearly the price-directive decoupling scheme of this section is easily incorporated into the nested decomposition scheme of Chapter 2. We simply relax the inventory-coupling constraints (3-26) along with the facility capacity constraints, and, as we have seen, the Lagrangian subproblem decouples into two independent subproblems that correspond to a transient and a pure production problem. Thus it is up to the price-adjusting mechanism of choice to determine appropriate prices for both capacity utilization of facilities and the inventory flows between the transient and pure production problem. Heuristics can be developed for generating feasible inventory flows from Lagrangian solutions with unequal
inventory flows between the transient and pure production problems. Moreover, these heuristics can be integrated with heuristics for constructing capacity-feasible schedules from the solutions of subproblems that result from pricing out capacity restrictions.

We mention that there are other price-directive schemes that may prove promising for solving LPs over general (i.e., not necessarily totally dynamic) I-P balance systems. The analysis and testing of these schemes is an area for future research and will not be discussed here.
Chapter 4  Analysis and Reintegration of the Benders Subproblem

4.1 Introduction

Chapters 2 and 3 have provided the rationale for employing Benders decomposition as a high-level solution methodology and then relaxing certain constraints of the resultant Benders subproblem. This nested decomposition scheme results in the isolation of special structure constraints (i.e., the inventory-production balance constraints) in a Lagrangian subproblem of the Benders subproblem. The goal of Chapter 4 is to explore in depth the implications and issues, both theoretical and practical, of using this decomposition strategy on our production planning models (BMRP) and (EMRP).

Section 4.2 discusses the characteristics of a dual problem that arises when the facility capacity and fixed charge constraints of our models are relaxed. Since elements of the feasible region of this dual problem will be used for generating the Benders master problem cuts, the implications of forming cuts from these elements, and their effect on the convergence properties of the nested decomposition algorithm, are discussed in detail.

Section 4.3 shows that there are compelling reasons for performing the Lagrangian relaxation differently than originally proposed. Specifically, the alternate relaxation involves only the facility capacity constraints. The reasons why this alternate relaxation leads to efficiencies relative to the original relaxation are outlined. We also begin to discuss in Section 4.3 the theoretical implications of performing Lagrangian relaxation on the Benders subproblem in the alternate fashion and the relationship between the two dual problems that arise from relaxing constraints in the original and the alternate manner.

In Section 4.4 we observe that the relaxation of Benders subproblem constraints in the alternate fashion raises interesting issues concerning the Benders master problem cuts thereby generated. We present a detailed study of the relationship between the two dual regions that are created by our two relaxation proposals. We also discuss the potential
benefits of generating Benders master problem cuts from the feasible region of a dual problem created by Lagrangian relaxation, relative to the cuts generated in the usual manner of solving the Benders subproblem as a monolithic LP via the simplex algorithm.

Our analysis includes the development of an algorithm that moves from a feasible solution of the alternate dual problem to a feasible solution of the original dual problem in such a way that the two solutions have equal objective function value in their respective problems. Section 4.5 revises this algorithm with the intention of making it practical to implement. This section also provides a synopsis of the issues involved in creating a solution to the alternate dual problem — which is generally easier to obtain than a solution to the original dual problem — and transforming it to a solution to the original dual problem.

Section 4.6 reveals the limitations and potential problems of forming Benders master problem cuts from solutions to the alternate dual problem. The section also includes an interpretative synthesis of results of the chapter encountered up to that point.

The final section of Chapter 4, Section 4.7, explores further the structure of the dual region created by performing the original relaxation of both the facility capacity constraints and the fixed charge constraints of the Benders subproblem. Properties of the rays of the convex polytope that define this region are developed. This analysis is used to create a characterization of equi-cost solutions (that is, solutions with the same objective function value) that are available for any feasible solution to the original dual problem. Some of these equi-cost solutions can be shown to be preferable to others, in the sense that they lead to better (or "stronger") Benders master problem cuts. Given this criterion for judging the value of solutions to the original dual problem as generators of master problem cuts, we evaluate the worth of the solutions to the original dual problem that we are capable of creating from the alternate dual region.
4.2 Benders Subproblem Formulations

We have seen that formulations (d-3:x) and (d-4:x) of Section 2.4 are equivalent LPs to (SPx) – the Benders subproblem derived by presetting the setup vector x – for models (BMRP) and (EMRP) respectively. These formulations are interpretable as the "full" Dantzig-Wolfe master problems where all elements of the set of extreme points of the D-W subproblems are known. We will relabel (d,x-3) and (d,x-4) as (SPx) throughout this chapter. Hence for (BMRP) we formulate problem (SPx) as

\[ v(SPx) = f(x) + \min \left| \sum_{i=1}^{L} \alpha_i (h \nu^i) + co \right| \quad (4-1) \quad (SPx) \]

s.t. \[ \sum_{i=1}^{L} \alpha_i (Bp^i) - o \leq a - Sx \quad (4-2) \]

\[ \sum_{i=1}^{L} \alpha_i p^i \leq Qx \quad (4-3) \]

\[ \sum_{i=1}^{L} \alpha_i = 1 \quad (4-4) \]

\[ \alpha_i \geq 0, i = 1, \ldots, L : \quad o \geq 0 \quad (4-5) \]

where, as in Section 2.4,

\[ Y_E = \left\{ (p, \nu^i) \right\}_{i=1}^L \]

is the set of extreme points of the feasible region of the Lagrangian subproblem.
\[ Y_{ip} = \left\{ (p,v) : Ap + Dv = d ; p,v \geq 0 \right\} \]

for model (BMRP). For (EMRP), we have

\[ \nu(SPx) = fx + \min \left[ \sum_{i=1}^{L} a_i (hv^i + bw^i) + co \right] \quad (SPx) \]

s.t. \[ \sum_{i=1}^{L} a_i (Bp^i) - o \leq a - Sx \]

\[ \sum_{i=1}^{L} a_i p^i \leq Qx \]

\[ \sum_{i=1}^{L} a_i = 1 \]

\[ a_i \geq 0, i = 1, ..., L : o \geq 0 \]

with

\[ Y_E = \left[ (p,v,w)^i \right]_{i=1}^{L} \]

and

\[ Y_{ip} = \left\{ (p,v,w) : Ap + Dv - Ew = 0 ; Fw = 1 ; p,v,w \geq 0 \right\} . \]
Note that we have used $L$ to represent the cardinality of $Y_E$ for both models for notational convenience. Recall that $Y_{IP}$ is the feasible region of the inventory-production balance constraint set. The dual of $(SP x)$ for (BMRP) is

$$
u(SP x) = \nu(SP x,D) = f x + \max \left[ \gamma \left( S x - a \right) - \lambda Q x + w \right]$$  \hspace{1cm} (4-6) \hspace{1cm} (SP x,D)$$

$$s.t. \hspace{0.5cm} w \leq h v^i + (\gamma B + \lambda p^i) \hspace{0.5cm} i=1,...,L$$  \hspace{1cm} (4-7)$$

$$0 \leq \gamma_{mt} \leq c_{mt} \hspace{0.5cm} m=1,...,M \hspace{0.5cm} t=1,...,T$$  \hspace{1cm} (4-8)$$

$$\lambda \geq 0$$  \hspace{1cm} (4-9)$$

The dual of $(SP x)$ for (EMRP) is similar with the exception that $hv^i$ is replaced by $hv^i + bw^i$.

For notational convenience, we will work primarily with the formulations for model (BMRP) throughout this chapter, since the majority of the analysis that follows applies to both models. Any real discrepancies between the two will be examined in detail as they arise.

It is instructive to think of $(SP x,D)$ as the dual problem derived from the Lagrangian relaxation of the capacity constraints and the fixed charge constraints of the Benders subproblem. Again, as in Section 2.3, $\gamma$ and $\lambda$ are the dual prices on the capacity and fixed charge constraints respectively, and the Lagrangian subproblem is labelled $(L(\gamma,\lambda;x))$. For any setting of $\gamma$ and $\lambda$, $0 \leq \gamma \leq c$, $\lambda \geq 0$, the bracketed function subject to maximization in (4-6) with scalar $w$ being set to the maximal value satisfying (4-7) – is equal to $\nu(L(\gamma,\lambda;x))$, the optimal objective function value of the Lagrangian $(L(\gamma,\lambda;x))$. For preset $\gamma$ and $\lambda$, $w$ represents the part of the optimal Lagrangian objective function value $\nu(L(\gamma,\lambda;x))$ that involves minimization; that is, $w = \min_{p,v} \gamma_{IP} \left[ hv + (\gamma B + \lambda) p \right]$. Solving $(SP x,D)$ thus solves the dual problem

$$\nu(d;x) = \sup_{\lambda \geq 0, 0 \leq \gamma \leq c} \nu(L(\gamma,\lambda;x)) \hspace{1cm} (d;x)$$
which, given the linear nature of the Benders subproblem, solves the Benders subproblem as well.

One can, then, return the Lagrangian objective function

\[
\nu(L(y, \lambda; x)) = y[Sx - a] - \lambda Qx + w = y[Sx - a] - \lambda Qx + \min_{p, v, \xi} \{ hv + (yB + \lambda)p \}
\]

with \((y, \lambda)\) being an optimal solution to the dual problem \((d'x)\), to the Benders master problem as the following master problem cut:

\[
\nu \geq \nu(L(y, \lambda; x)).
\]

The vector \((y, \lambda, w)\) is optimal in \((SPx,D)\).

Note that \(Y_E\) is independent of \(x\). Moreover, the feasible region \(S\) of \((SPx,D)\) is fully qualified as a region for generating valid Benders master problem cuts, because it is a convex polytope that is independent of \(x\); i.e., only the objective function of \((SPx,D)\) is a function of \(x\). Also, any element of \(S\) can be used to generate a valid cut. This is because any vector that is feasible in a formulation of a Benders LP subproblem, where the feasible region of the formulation is independent of the variables set in the Benders master problem, can be used to create valid Benders master problem cuts.

If D-W decomposition is employed as a solution methodology, the constraints of (4-7) will be generated iteratively as the algorithm progresses. Hence the full constraint set (4-7) is generally not known to us at any given point of the optimization. The dual of the D-W master problem at any algorithm iteration suggests a vector \((y, \lambda, w)\) which is, in general, superoptimal and infeasible in \((SPx,D)\). The D-W subproblem is given the Lagrangian prices \((y, \lambda)\) and performs the necessary optimization to find the true value of \(\nu(L(y, \lambda; x))\). The resulting vector \((y, \lambda, \min_{p, v, \xi} \{ hv + (yB + \lambda)p \})\) is feasible but generally suboptimal in \((SPx,D)\).
The D-W algorithm terminates when the D-W master and subproblems have equal objective function values. This termination may, of course, happen before all elements of constraint set (4-7) are known. On the surface, then, it appears that solving (SPx) via D-W decomposition leads to a dual solution (y,λ,w) to a relaxation of (SPx,D) (specifically, a relaxation of the constraints of (4-7) corresponding to unknown elements of Y_F). However, the fact that the D-W master and subproblems have equal objective function values implies (y,λ,w) is feasible in any region formed by adding any of the constraints corresponding to currently unknown elements of Y_F. Since (y,λ,w) is optimal in the relaxed region with the current objective function, it is optimal in any of the resulting "tightened" regions as well with the same objective function because of its feasibility in the tightened regions. Specifically, it is optimal over S, the feasible region of (SPx,D).

It is clear, then, that our employment of the D-W algorithm is a method for generating valid cuts from a dual region that is a convex polytope. Moreover, the dual region is independent of the variables preset in the Benders master problem. Therefore the nested decomposition algorithm for the global optimization problem will converge finitely to an optimal global solution. It is relatively straightforward to extrapolate to a more general situation and conclude that this nested technique remains valid for any mixed-integer programming problem in which the Lagrangian derived by relaxing Benders LP subproblem constraints has a feasible region that is independent of the master problem variables.

In Section 3.4 we devised conditions under which the I-P balance system could be converted to an equivalent totally dynamic formulation by eliminating, at most, a few unnecessary constraints. For example, we called feasible balance systems with no initial stocks and holdover production pure production problems and observed that such systems satisfy these conditions. For mathematical clarity, we assume throughout this chapter that the I-P balance systems of both models (BMRP) and (EMRP) are pure production problems. Actually, none of the results presented in this section depend on the Lagrangian subproblem within the Benders subproblem possessing this, or any other, special structure. Recall,
however, that the original motivation for employing Lagrangian relaxation in the Benders subproblem was to isolate special structure constraints in a Lagrangian subproblem because of the speed with which LPs over such constraints can be solved.

A method for compensating for situations where the above assumption (i.e., all of the inventory-production constraints can be included in a special structure Lagrangian subproblem) is not true is readily available. The set of priced-out constraints can be expanded to include I-P constraints of types 1 and 2 (see Section 3.3 for constraint type definitions), while the Lagrangian subproblem is shrunk to include only I-P constraints of type 3, thereby ensuring a subproblem with the totally dynamic special structure. Alternatively, the algorithms of Sections 3.9 and 3.10 can be employed on I-P balance systems that are not readily converted to equivalent totally dynamic systems. Such approaches include virtually all constraints of the I-P balance system in the subproblem.
4.3 Alternate Subproblem Formulations

Presetting the setup vector $\mathbf{x}$ in the Benders master problem imposes a clear restriction on the feasibility of production vectors in the Benders subproblem. Specifically, only those components $p_{it}$ corresponding to $X_{it} = 1$ will be allowed positive values.

From $Y_E$ and the preset setup vector $\mathbf{x}$ we formulate the following subset of $Y_E$:

$$Z_{E,x} = \left\{ (p,v) \in Y_E : X_{it} = 0 \Rightarrow p_{it} = 0 \right\}.$$

From constraint set (4-3), $(p,v) \in Y_E \cdot Z_{E,x}$ implies $a_i = 0$, since otherwise some constraint with right hand side $q_{it}X_{it} = 0$ is violated. Hence only subproblem-generated columns that are elements of $Z_{E,x}$ are of use in the D-W master problem, in the sense that only these columns can have their variables set positive in some feasible solution. This observation combined with unbounded overtime implies $(SP\mathbf{x})$ is feasible for model (BMRP) if and only if $Z_{E,x}$ is nonempty. If overtime is bounded, then we temper this last statement to read: constraint sets (4-3) through (4-5) can be satisfied if and only if $Z_{E,x}$ is nonempty. Note that $(SP\mathbf{x})$ is always feasible for (EMRP), even with bounded overtime, because one always has the option of not producing anything in the extended model.

We will assume that $\mathbf{p} \leq q = Q1$ $\forall (p,v) \in Y_{IP}$. The reasons why this assumption is made will become clear as the analysis of this chapter evolves. (A method for setting the positive diagonal components of the $Q$ matrix was presented in Section 3.9.) Hence any convex combination of master problem columns generated from $Y_E$ is feasible relative to constraints of (4-3) with $X_{it} = 1$. Moreover, the complete constraint set (4-3) is not violated by any convex combination of columns generated from elements of $Z_{E,x}$, and it is clear that the set of all convex combinations of $Z_{E,x}$-generated columns defines the set of $\alpha$ vectors that are feasible relative to constraint set (4-3).
Therefore, from the standpoint of computational effort, it would be advantageous to be able to generate only elements of $Z_{E,x}$ in the D-W subproblem $(SSP-DW)$; these columns are the only ones of use in $(SPx)$, and if only such elements are generated the constraint set $(4-3)$ could be dropped from $(SPx)$. The result, then, would be a D-W master problem with the potential for significantly fewer rows and columns.

We define

$$p_x = \text{the subvector of } p \text{ obtained by deleting components of } p \text{ constrained to zero by the setting of } x.$$  

If $B$ is a general matrix which can premultiply $p$, we let

$$B_x = \text{the submatrix of } B \text{ created by eliminating the columns of } B \text{ corresponding to the components of } p \text{ eliminated in creating } p_x.$$  

Also, let

$$L_x = \text{the cardinality of } Z_{E,x}.$$  

Given our observations, an equivalent formulation (with $a_j$ variables that are constrained a priori to zero removed) for $(SPx)$ is
\[ v(SPx) = v(SPx - 2) = fx + \min \left[ \sum_{i=1}^{L_x} a_i \left( hv^i \right) + co \right] \quad (4-10) \quad (SPx-2) \]

\[ s.t. \quad \sum_{i=1}^{L_x} a_i (B_x p^i_x) = a - Sx \quad (4-11) \]

\[ \sum_{i=1}^{L_x} a_i = 1 \quad (4-12) \]

\[ a_i \geq 0, \quad i = 1, \ldots, L_x ; \quad o \geq 0 \quad (4-13) \]

with corresponding dual

\[ v(SPx) = v(SPx - 2, D) = fx + \max \left[ y(Sx - a) + w \right] \quad (4-14) \quad (SPx-2, D) \]

\[ s.t. \quad w \leq hv^i + yB_x p^i_x \quad i = 1, \ldots, L_x \quad (4-15) \]

\[ 0 \leq y_{mt} \leq c_{mt} \quad m = 1, \ldots, M \quad t = 1, \ldots, T \quad (4-16) \]

In this formulation the set of columns (rows) of \((SPx-2, (SPx-2, D))\) is formed from the equivalent set \(Z_{E_x}\). Take note, however, that the D-W subproblem

\[ \min (yB + \Lambda)p + hv \quad ( + bw \quad \text{for model (EMRP)}) \quad (SSP-DW) \]

\[ s.t. (p,v) \in Y_{lp} \]

of master problem \((SPx)\) does not incorporate the information conveyed by the setup vector \(x\).

Hence there is nothing to prevent \((SSP-DW)\) from returning elements of \(Y_E - Z_{E,x}\) to the
master problem \((SPx)\). The return of an element of \(Y_E - Z_{E,x}\) to the master problem in our original formulation implies the corresponding variable \(a_i\) has negative reduced cost in the current basis and thus "should" be pivoted into the basis. It can only be pivoted in at a value of 0, however, implying a degenerate solution. The problem, then, is how to construct a D-W subproblem which returns only elements of \(Z_{E,x}\) as columns with negative reduced cost in the current D-W master problem.

The path to alleviating this immediate problem begins by observing that there is arbitrariness in the way the D-W master and subproblems are formulated from the Benders subproblem. Specifically, we point out that the fixed charge constraint set ((1-4) of (BMRP), (1-9) of (EMRP)) could have been left in the Lagrangian subproblem, which implies the set would then be in the D-W subproblem instead of the D-W master problem. The D-W subproblem corresponding to the alternate relaxation for (BMRP) is

\[
v(SSP - DWx) = \min \left\{ (yB) p + hv \right\} \quad (4 - 17) \quad (SSP - DWx)
\]

\[
s.t. \quad Ap + Dv = d \quad (4 - 18)
\]

\[
p \leq Qx \quad (4 - 19)
\]

\[
p,v \geq 0 \quad (4 - 20).
\]

Recall that the constraint set \(p \leq q = Q1\) is satisfied by all elements of \(Y_{IP}\). Hence \((SSP-DWx)\) can be reformulated as

\[
v(SSP - DWx) = \min \left\{ (yB) p_x + hv \right\} \quad (4 - 21) \quad (SSP - DWx)
\]

\[
s.t. \quad Ap_x + Dv = d \quad (4 - 22)
\]

\[
p_x,v \geq 0 \quad (4 - 23)
\]
where components of \( p \) that are preconstrained to zero by \( x \) are removed. The (EMRP) subproblem can be similarly constructed and reformulated. We define \( Y_{IP,x} \) to be the feasible region of (SSP-DWx) as formulated with the reduced production vector \( p_x \) ((4-22) through (4-23)). We point out that while \( Y_{IP} \) is independent of \( x \), \( Y_{IP,x} \) is not. Also, \( Y_{IP} \) is nonempty by assumption, whereas \( Y_{IP,x} \) may be empty, depending on \( x \).

We define the transformation \( g \) with domain \( Z_{E,x} \) and range \( g(Z_{E,x}) = Q_x \), where \( g \) transforms each element \( (p,v) \in Z_{E,x} \) into the subvector \( (p_x,v) \) of \( (p,v) \) obtained by deleting components of \( p \) preconstrained to zero by the setting of \( x \). The relationship between the Benders subproblem and \( Y_{IP,x} \) for (BMRP) is given by the following theorem:

**Theorem 4-1**: For model (BMRP), \( Y_{IP,x} \) is nonempty if and only if the Benders subproblem is feasible.

**Proof**: If \( Y_{IP,x} \) is nonempty, then there exists \( (p_x,v) \in Y_{IP,x} \) such that \( (p_x,v) \) can be expanded to \( (p,v) \in Y_{IP} \) by adding back the appropriate components (i.e., those preconstrained to zero) with value zero. Furthermore, \( p \leq Qx \) by construction. This implies the existence of some element \( (p,v) \in Y_E \) such that \( p \leq Qx \). Consequently, \( Z_{E,x} \) is nonempty and the subproblem is feasible.

If the subproblem is feasible, there must exist \( (p,v) \in Y_E \) such that \( p \leq Qx \), and so \( (p,v) \in Z_{E,x} \). Clearly then, \( g([p,v]) \) is a well defined element of \( Y_{IP,x} \).

If overtime is bounded in (BMRP), then we must alter the statement of the theorem to read: \( Y_{IP,x} \) is nonempty if and only if constraint set (4-3) through (4-5) can be satisfied. We assume the Benders subproblem is feasible throughout the remainder of this section.

Let \( Y_{E,x} \) be the set of extreme points of \( Y_{IP,x} \) and let \( N_x \) be the cardinality of \( Y_{E,x} \). The D-W master problem of the formulation with this altered Lagrangian subproblem is
\[ \nu(SPx) = \nu(SPx - 3) = fx + \min \left[ \sum_{i=1}^{N_x} a_i (hv^i) + co \right] \quad (4-24) \]

\[ \text{s.t. } \sum_{i=1}^{N_x} a_i (B_x p_x^i) - o \leq a - Sx \quad (4-25) \]

\[ \sum_{i=1}^{N_x} a_i = 1 \quad (4-26) \]

\[ a_i \geq 0, i = 1, \ldots, N_x : \quad o \geq 0 \quad (4-27) \]

with corresponding dual

\[ \nu(SPx) = \nu(SPx - 3, D) = fx + \max \left[ y(Sx - a) + w \right] \quad (4-28) \]

\[ \text{s.t. } w \leq hv^i + yB_x p_x^i \quad i = 1, \ldots, N_x \quad (4-29) \]

\[ 0 \leq y_{mt} \leq c_{mt} \quad m = 1, \ldots, M \quad t = 1, \ldots, T \quad (4-30) \]

Notice that \((SSP-DWx)\) will never generate columns for the master problem \((SPx-3)\) whose variables are constrained to zero \textit{a priori} in any feasible master problem solution. We have seen that this desirable characteristic is not possessed by our original D-W formulation \((SPx)\). In this alternate formulation, the setup information is retained in the Lagrangian subproblem, meaning that information on production variables constrained to zero is now conveyed to the Lagrangian problem. Consequently the number of subproblem variables is reduced in comparison to the original formulation. Moreover, as previously noted, the
master problem formulation has the potential for significantly fewer rows and columns than
the original master problem \((SPx)\).

We now establish that \((SPx-2)\) and \((SPx-3)\) are really equivalent formulations. The
implication of this result is that columns for \((SPx-2)\) can be generated exclusively from the
set \(Z_{E,x}\) by simply leaving the fixed charge constraints in the Lagrangian subproblem. To
demonstrate the equivalence of \((SPx-2)\) and \((SPx-3)\), we must show that \(Z_{E,x}\) and \(Y_{E,x}\) are
equivalent sets. Consider the general nonempty convex polyhedron

\[
X = \left\{ (x_1, x_2) : C^1 x_1 + C^2 x_2 = b : x_1, x_2 \geq 0 \right\}.
\]

Let \(E\) be the extreme points of \(X\). Define \(E_0 \subseteq E\) by

\[
E_0 = \left\{ (x_1, x_2) : (x_1, x_2) \in E, x_2 = 0 \right\}.
\]

Also, let

\[
X_1 = \left\{ C^1 x_1 = b : x_1 \geq 0 \right\}
\]

and \(E_1\) be the extreme points of \(X_1\). Assume \(X_1\) is a nonempty set.

**THEOREM 4-2:** \((x^1,0) \in E_0\) if and only if \(x^1 \in E_1\).

**PROOF:** It is possible that \([C^1, C^2]\) and/or \(C^1\) do not have full row rank. In this case we
expand the definition of a basis to be a maximally independent set of columns, a nonnegative
linear combination of which produces the right hand side vector.

Let \(m\) be the rank of the constraint matrix defining \(X\), and \(m_1 (\leq m)\) the rank of the
constraint matrix defining \(X_1\). Assume \((x^1,0) \in E_0\), implying \(C^1 x^1 = b\). Take any basis \(B\)
which produces \((x^1, 0)\) in \(X\) and let \(B^1\) be the submatrix of \(C^1\) such that \(B^1\) consists of the columns of \(C^1\) which are also in \(B\). The columns of \(B^1\) are linearly independent and \(B^1\) multiplied by \(x^1, B\) – the subvector of basic elements of \(x^1\) relative to basis \(B\) – produces \(b\). If the rank of \(B^1\) is \(m_1\), a basis of \(X_1\) producing extreme point \(x^1 \in E_1\) is at hand; namely \(B^1\). If \(B^1\)'s rank is less than \(m_1\), simply add linearly independent columns of \(X_1\) to \(B^1\) to produce a basis of \(X_1\) that yields \(x^1 \in E_1\).

Now assume \(x^1 \in E_1\) and let \(B^1\) be a basis of \(X_1\) which produces \(x^1\). Expand \(x^1\) to \((x^1, 0) \in X\). If \(\text{rank}(B^1) = m^1 = m\), then \(B^1\) is a feasible basis of \(X\) producing \((x^1, 0)\), implying \((x^1, 0) \in E_0\). Otherwise, \(\text{rank}(B^1) = m_1 < m\), and \(B^1\) can be augmented to a basis \(B\) of \(X\) by adding linearly independent columns of \(X\) to \(B^1\). Basis \(B\) also produces \((x^1, 0)\), and again \((x^1, 0) \in E_0\).

Therefore, the result is shown. □

Recall the matrix \(A\) that premultiplies \(p\) in the I-P balance constraints. Let \(A_{x^0}\) be the columns of \(A\) corresponding to components of \(p\) constrained to zero by \(x\). Therefore each column of \(A\) is either in \(A_x\) or \(A_{x^0}\).

THEOREM 4-3: \(Q_x = Y_{E,x}\). Thus transformation \(g\) can be viewed as a mapping from \(Z_{E,x}\) to \(Y_{E,x}\) that is onto \(Y_{E,x}\). Furthermore, \(g\) is one-to-one. Therefore \(g^{-1}\) exists, and the sets \(Z_{E,x}\) and \(Y_{E,x}\) are equivalent.

PROOF: Since the Benders subproblem is assumed to be feasible, Theorem 4-1 implies \(Y_{IP,x}\) is nonempty regardless of which of the two models we are considering. We use Theorem 4-2 with \(X = Y_{IP}\) and \(C^2 = A_{x^0}\). Hence \(E = Y_{E}, E_0 = Z_{E,x}, X_1 = Y_{IP,x}, \) and \(E_1 = Y_{E,x}\). The "only if" part of Theorem 4-2 implies \(g((p, v)) \in Y_{E,x}\) for any \((p, v) \in Z_{E,x}\), and so \(Q_x \subseteq Y_{E,x}\).

If \((p_x, v) \in Y_{E,x}\), then the "if" part of Theorem 4-2 implies \((p_x, v)\) expanded to a full vector \((p, v) \in Y_{IP}\) by adding appropriate components at value zero is also an element of \(Z_{E,x}\). Furthermore, \(g((p, v)) = (p_x, v) \in Q_x\). Since \((p_x, v)\) is an arbitrary element of \(Y_{E,x}\), \(Y_{E,x} \subseteq Q_x\).
and we conclude \( Q_x = Y_{E,x} \). The proof is completed by observing that \( g \) is one-to-one by definition.

Note that the inverse transformation \( g^{-1} \) is as simple as \( g \). The function \( g^{-1} \) takes any \((p_x,v) \in Y_{E,x}\) and expands it to a full vector \((p,v) \in Z_{E,x}\) by simply adding components eliminated by \( X_{it} = 0 \) and setting these components to zero.

We now discuss the structure of the constraint set defining \( Y_{IP,x} \) for both models (BMRP) and (EMRP). Recall our assumption that \( Y_{IP} \) is defined by a pure production problem. This means \( Y_{IP} \) can be defined by a totally dynamic system consisting of I-P balance constraints with indices of the form \( \{ it : i = 1, \ldots, N ; L_i^* + 1 \leq t \leq T \} \), where \( N \) is the number of items in the product structure. (See sections 3.4 and 3.8.)

This totally dynamic system is altered to form a constraint set defining \( Y_{IP,x} \) by removing columns corresponding to production vectors \( P_{it} \) with \( X_{it} = 0 \). The remaining system satisfies the properties of totally dynamic systems with the possible exception that some constraints may now have no critical columns. For example, \( P_{i,L+i} \) is the only critical variable for constraint \( i,L^* + 1 \forall i \), so the removal of any of these production variables produces such a constraint.

If the altered totally dynamic system still has a critical column for each constraint, then it is obviously still totally dynamic and \( Y_{IP,x} \) is nonempty. If this is not the case, then any constraint with no critical columns and a zero right hand side forces all variables with nonzero coefficients in the constraint to zero in any feasible solution. This may cause a "snowballing" or "cascading" effect of eliminating the critical columns of other constraints, as we've seen several times in Chapter 3. If, at any point in this iterative procedure of eliminating constraints and variables, a constraint is found with no critical columns and a positive right hand side, then \( Y_{IP,x} \) is infeasible. Otherwise \( Y_{IP,x} \) is nonempty, and the resultant system of remaining constraints and variables is totally dynamic. Further details
on this process of finding a totally dynamic formulation that defines the region $Y_{IP,x}$ are given in Chapter 5.

Therefore, it is relatively easy to determine whether $Y_{IP,x}$ (and hence the Benders subproblem) is feasible for model (BMRP) (the Benders' subproblem is always feasible for (EMRP)) when dealing with pure production problems. Actually, it is possible to determine in a relatively straightforward manner whether Benders subproblems of (BMRP) with general (i.e., not necessarily pure production) I-P balance constraint sets are feasible. This involves a redefinition of $C_{it}$, the net cumulative demand for $i$ in period $t$, presented in Section 3.9. Production variables preconstrained to zero are accounted for, and Theorem 3-10 can be applied to the redefined $C_{it}$. We will not present the details of the procedure in this work.

Recall that the vector $q = Q_1$ was assumed to be constructed as an upper bound on the production vector $p$ that is satisfied by all elements of $Y_{IP}$. An interesting question is: what happens when $q$ is constructed so that the constraint $p \leq q$ is not necessarily satisfied by all elements of $Y_{IP}$? Such a $q$ could be found in models where the scheduler wishes to place limits on production beyond those implied by the facility capacity constraints.

Let's examine what happens in such situations when we perform the Lagrangian relaxation on the Benders subproblem so that the fixed charge constraints remain in the Lagrangian subproblem. Observe that the formulation ((4-17) through (4-20)) for the Lagrangian subproblem cannot be reformulated as ((4-21) through (4-23)) as before, specifically because $p \leq q$ alters the region $Y_{IP}$. The correct reformulation that utilizes $p_x$ is

$$v(SSP - DWx) = \min \{ (yB + \lambda) p_x + hv \} \quad (4-31) \quad (SSP - DWx)$$

s.t. $A_x p_x + Dv = d \quad (4-32)$

$$p_x \leq q_x \quad (4-34)$$

$$p_x', v \geq 0 \quad (4-33).$$
Let's assume (4-32) combined with (4-33) define a feasible region, which implies (4-32) can be formulated as an equivalent totally dynamic system. We now consider totally dynamic systems that in addition have simple upper bounds on variables, such as our latest formulation of $(SSP-DWx)$. A straightforward observation is that, in contrast to unbounded totally dynamic systems, the bounded totally dynamic systems may be infeasible.

Moreover, it is well known that the simple bounding variables can be incorporated into the simplex method, without having to add the bounding constraints in the tableau, by using generalized upper bounding (e.g., Dantzig and Van Slyke (1967); Lasdon (1970), Chapter 6). One would hope that such a procedure could be incorporated into the totally dynamic constraint set algorithm of Section 3.6 to handle bounded totally dynamic systems without radically impairing the algorithm's structure and speed.

Unfortunately, such a simple generalization does not appear to be possible. To show why this is so, we recall that our basis for being able to solve totally dynamic systems efficiently is Theorem 3-6 of Section 3.5. This theorem states that any extreme point of such systems can be generated from the set $T$ of upper triangular bases with strictly positive diagonals and nonpositive components above the diagonal (i.e., the set of full, square totally dynamic submatrices). The knowledge that the search for a primal optimal basis can be limited to such a set is the key to the algorithm of Section 3.6.

We state a well known result from the theory of linear programming without proof:

**THEOREM 4-4:** Consider the LP

$$\min cx$$

$$s.t. \quad Ax = b$$

$$\quad 0 \leq x \leq d$$
\( \mathbf{x}^* \) is an extreme point of the convex polyhedron forming the feasible region of this LP if and only if \( \mathbf{x}^* \) can be generated from a basis \( \mathbf{B} \) of constraint matrix \( \mathbf{A} \) with all components of the variable vector \( \mathbf{x} \) whose columns are not in \( \mathbf{B} \) set to either their lower or upper bound.

The reason that the totally dynamic algorithm cannot be readily extended to the bounded variable case is that it is not true in general that every extreme point of such systems can be generated from an element of \( \mathbf{T} \) with nonbasic variables set at upper or lower bounds. Consider, for example, the bounded totally dynamic system

\[
\begin{bmatrix}
1 & 0 & 0 & -1 & 0 \\
0 & 1 & 0 & 0 & -1 \\
0 & 0 & 1 & 1 & 1 \\
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2 \\
x_3 \\
x_4 \\
x_5 \\
\end{bmatrix}
= 
\begin{bmatrix}
0 \\
0 \\
1 \\
\end{bmatrix}
\]

\( 0 \leq x_1, x_2 \leq .5 \)

\( 0 \leq x_3, x_4, x_5 \leq 5. \)

The vector

\[
\begin{bmatrix}
x_1 \\
x_2 \\
x_3 \\
x_4 \\
x_5 \\
\end{bmatrix}
= 
\begin{bmatrix}
.5 \\
.5 \\
0 \\
.5 \\
.5 \\
\end{bmatrix}
\]
is an extreme point generated by the basis consisting of the third through the fifth columns. The vector cannot be generated by any of the (3) elements of $T$ for this system.

Without the help of Theorem 3-6, we are apparently faced with the prospect of reverting to the generalized upper bounding version of the simplex algorithm for solving bounded totally dynamic systems. This prospect obviates our motivation for relaxing only the capacity constraints in the Benders subproblem to create a subproblem that includes the fixed charge constraints. It is possible that another approach may yield a special structure algorithm for bounded totally dynamic systems, but we will not attempt to find it in this work. We leave this as a topic for possible future research. It can also be shown that the equivalence between the extreme point sets $Z_{E,x}$ and $Y_{E,x}$ no longer holds if $Y_{E,x}$ represents the extreme points of (SSP-DW$x$) when $p \leq q$ is not satisfied by all elements of $Y_{IP}$.

We will assume that all elements of $Y_{IP}$ satisfy $p \leq q$ throughout the remainder of this chapter. Note that any simple bounds on production that would not be satisfied by all elements of $Y_{IP}$ could be placed in the capacity constraints instead of being incorporated into the fixed charge constraints. This is accomplished by creating a dummy machine for items with bounds on production that have no setups, no overtime variables, and capacities set to the desired upper bounds.
4.4 Analysis of Subproblem Duals

An important observation is to note the dependence of the feasible region of \((SPx-2,D)\) on \(x\). This dependence raises some fundamental questions: Can we solve the primal, dual pair \((SPx-2),(SPx-2,D)\), return extreme points of \((SPx-2,D)\) (which are \(x\)-dependent) to the master problem, and be guaranteed immunity from incorrect results? That is, can we run the nested decomposition algorithm to termination by creating such Benders cuts and be guaranteed convergence to a global optimal solution in a finite number of iterations? Recall that the standard proof of the finite convergence of Benders' algorithm hinges on a dual feasible region to the LP subproblem that is independent of the variables set in the master problem. If such a guarantee is not possible, can we still solve the more efficient \((SPx-2)\) as the subproblem and somehow use the results to obtain, relatively quickly, a dual extreme point from the feasible region \(S\) of \((SPx,D)\) to return to the Benders master problem?

We will show the answers to these questions are no and yes respectively. In Section 4.6 we construct a specific example within our general model context in which returning \(x\)-dependent extreme points of \((SPx-2,D)\) leads to a nonoptimal primal feasible solution to (BMRP) being reported as optimal by the nested decomposition algorithm. In this section we analyze the relationship between the two dual regions \((SPx,D)\) and \((SPx-2,D)\). We show how a feasible solution to \((SPx,D)\) is always obtainable from any feasible solution to \((SPx-2,D)\) such that the two solutions have equal objective function value.

We mention that once either of these formulations is solved, yielding an optimal vector \((p,v,o)\) to the Benders subproblem, this vector could then be used to obtain an optimal solution to the dual of the standard (i.e., no D-W decomposition) Benders subproblem, as given by (2-7) through (2-11). Assuming \((p,v,o)\) is a basic solution, and letting \(B, e^B\) be a basis which produces \((p,v,o)\) and its corresponding cost vector respectively, the task at hand is equivalent to finding \(\eta = B^{-1}e^B\).
Whereas this option is open, we recall that one of our primary motivators in this research is to establish the theoretical groundwork for finding "good" but not necessarily optimal solutions to our models. Therefore it may be advantageous to be able to generate valid Benders cuts without having to find an optimal dual solution to the Benders subproblem. The employment of the simplex method in solving the Benders subproblem requires us to solve the subproblem to optimality. This is because the simplex algorithm maintains primal feasibility and complementary slackness throughout, and dual feasibility is attained only at the last algorithm iteration. Since we need at least a dual feasible vector to return to the Benders master problem, the simplex method, if employed, must be run to completion. We could solve the dual as the primal, thereby generating valid dual solutions for potential Benders cuts at every simplex iteration. But by doing this, we would forgo generating global feasible solutions (given a feasible Benders subproblem) unless we solve the Benders subproblem to optimality. However, in Chapter 1 we discussed how the ability to generate global feasible solutions relatively frequently is an asset given our solution goals.

It is clear that the Dantzig-Wolfe formulation (SPx) not only solves a restriction of the general Benders subproblem in the primal D-W master problem at any iteration (and thus obtains a feasible solution for a feasible Benders subproblem), but the formulation also yields dual feasible solutions (elements of $S$) as well at each iteration via the D-W subproblem. By testing the objective function values of these primal and dual solutions, we may well find it worthwhile to stop the Benders subproblem short of optimality.

More generally, any valid approach for solving the dual problem implied by the relaxation of the facility capacity constraints and the fixed charge constraints of the Benders subproblem can be employed to generate Benders cuts. Besides Dantzig-Wolfe decomposition, some of the more well known dual algorithms include subgradient optimization (see Polyak (1967, 1969) and Demyanov (1966)) and the generalized primal-dual ascent algorithm (see Grinold (1972) and Fisher et al. (1975)). A solution to any of the
Lagrangian subproblems solved by these algorithms can readily be used to generate a valid Benders cut.

At any time the Lagrangian solutions can be used to construct D-W master problem columns, and solving this master problem will create feasible solutions. Heuristics can also be employed to construct a feasible solution strictly from the Lagrangian solution at hand. For example, in models with unbounded overtime for every facility in every period, simply set any overtime variable equal to the difference between the utilization of the facility in the period and the capacity of the facility in the period if this difference is positive, and zero otherwise. When overtime is bounded, the same procedure can be applied and the overtime assignments can be checked to see if they are feasible. As we have noted, tight bounds on capacities and/or very expensive overtime may make it difficult to generate good feasible solutions from Lagrangian solutions, because it may prove difficult to generate capacity-feasible solutions from Lagrangian solutions that violate capacities in multistage production problems. (We observe that this topic of how to generate such solutions is another possible area for future research.) In any event, the objective function value of the "best" feasible solution found may be used to judge the worth of the "best" Lagrangian-based Benders cut encountered so far.

Finally, beyond the possible practical applications of forming Benders master problem cuts from solutions to Lagrangian relaxations of the Benders subproblem, the technique opens up an interesting area of theoretical research.

Note the relationship between the variable vectors of problems \((SP_x,D)\) and \((SP_x-2,D)\). The former is obtained from the latter by the addition of the \(\lambda\) vector corresponding to constraint set (4-3). The feasible region \(S\) of \((SP_x,D)\) is unbounded. We can verify this fact by rewriting (4-7) as

\[
\omega - \gamma B p^i - \lambda p^i \leq h v^i \quad i = 1, \ldots, L \tag{4-17}
\]
The vector \((y, \lambda, w) = (0, 0, 0)\) is feasible because \(hvi \geq 0\). Since \(\lambda p_i \geq 0\) \(\forall \lambda \geq 0\), and \(w\) is unconstrained except for (4-17), any \((0, \lambda, w)\) with \(\lambda \geq 0, w \leq 0\) is a ray of the feasible region of \((SP, D)\). Hence \(S\) is unbounded. In a similar fashion, we can show that the feasible region \(S_x\) of \((SP_x-2, D)\) is unbounded.

We now clarify the relationship between \((SP_x, D)\) and \((SP_x-2, D)\). Let \((y, w)\) be any feasible solution to \((SP_x-2, D)\). We expand \((y, w)\) to \((y, 0, w)\), the zero vector being of the same dimension as \(\lambda\) in \((SP_x, D)\). For any \((v, p_x)^i \in Y_{E,x}\), we have \(B_xp_x^i = Bp^i\) where \((v, p)^i = g^{-1}[(v, p_x)^i]\). Also, since \((y, w)\) satisfies all of constraint set (4-15), it follows that \((y, 0, w)\) satisfies the subset of constraint set (4-7) consisting of constraints formed from the elements of \(Z_{E,x}\). (This follows from the equivalence of the sets \(Z_{E,x}\) and \(Y_{E,x}\).)

There is no guarantee of \((y, 0, w)\) satisfying the remaining constraints of (4-7); i.e., those corresponding to elements of \(Y_E - Z_{E,x}\). Consequently, \((y, 0, w)\) is not necessarily feasible in \((SP_x, D)\). We can use \((y, 0, w)\) as the initial candidate in an algorithm for finding a feasible solution to \((SP_x, D)\), however. Within this algorithm, \((y, \lambda, w)^s\) represents the incumbent solution - i.e., the current proposed solution to \((SP_x, D)\) - while \(V_x\) represents the set of elements of \(Y_E\) for which the incumbent is not feasible in the corresponding constraints of (4-7).

### Feasibility Algorithm One

0) **Initialization**: \((y, \lambda, w)^s \leftarrow (y, 0, w)\). \(V_x \leftarrow Y_E - Z_{E,x}\). Reduce \(V_x\) by eliminating all elements corresponding to constraints of (4-7) satisfied by \((y, \lambda, w)^s\).

1) **Test for end of problem**: If \(V_x\) is empty, \((y, \lambda, w)^s\) is feasible in \((SP_x, D)\) and we're done.

 Otherwise go to step 2.
2) **Incumbent revision**: Choose \((p,v) \in V_x\) and find item \(i\) and period \(t\) such that \(P_{it} > 0\) and \(X_{it} = 0\). Set \(v_e = \omega^s - hv - \gamma^s Bp - \lambda^s p\). Update the incumbent by altering only \(\lambda_{it}^s\) by \(\lambda_{it}^s \leftarrow \lambda_{it}^s + (v_e / P_{it})\).

3) **Reduction of \(V_x\)**: \(V_x = V_x - \{(p,v)\}\). Check elements of \(V_x\) to see if there are any corresponding to constraints of (4-7) now satisfied by incumbent \((\gamma, \lambda, \omega)^s\). Eliminate all such elements from \(V_x\). Go to step 1. 

Steps 2 and 3 of the algorithm require some explanation. We know there exists some item \(i\) and period \(t\) such that \(P_{it} > 0\) and \(X_{it} = 0\) because \((p,v) \in Y_E - Z_{E,x}\). By reviewing (4-7), we observe that \(v_e\) is positive since the constraint of (4-7) corresponding to \((p,v)\) is violated by incumbent \((\gamma, \lambda, \omega)^s\). The incumbent alteration increases \(\lambda_{it}^s\) to the point where \(\omega^s - \gamma^s Bp - \lambda^s p = hv\), implying the new incumbent does not violate the constraint of \((SPx,D)\) corresponding to \((p,v)\). Hence \((p,v)\) is removed from \(V_x\) in step three. The new incumbent is readily seen to satisfy any constraints corresponding to elements of \(Y_E - V_x\) before \((p,v)\) is removed from \(V_x\). This is because the incumbent before the revision satisfied these constraints, and from (4-17) it is apparent that the revision cannot increase the left hand side of any of the \(L\) inequalities. This nonincreasing property means elements of \(V_x\) after the removal of \((p,v)\) can also have their corresponding constraints of \((SPx,D)\) satisfied by the revised incumbent, hence the second procedure in step three to remove unnecessary elements of \(V_x\).

Constraint set (4-9) will always be satisfied by any incumbent; this is true of our initial incumbent, and any incumbent differs from its predecessor by an increase in exactly one component of \(\lambda\). The algorithm is guaranteed to terminate with a feasible solution to \((SPx,D)\) in a finite number of steps. To see this, note that at each step at least one additional
constraint of (4-7) is satisfied by the incumbent, and any constraint satisfied by an incumbent is satisfied by all succeeding incumbents. Since each constraint of (4-7) corresponds to exactly one element of \( Y_E \), and \( Y_E \) is finite, the result is shown. Of course the algorithm as stated is of limited practical value because the set \( Y_E \) is not known in all likelihood. The algorithm will be revised later to compensate for this deficiency. Also of interest is the following theorem:

**THEOREM 4-6**: The feasible solution to \((SPx,D)\) found by the algorithm has the same objective function value in \((SPx,D)\) as the solution of \((SPx-2,D)\), used to initiate the algorithm, has in \((SPx-2,D)\).

**PROOF**: It is clear that \((y,w)\) and \((y,0,w)\) do not differ in objective function value. To prove the theorem, it suffices to show that the objective value does not change from incumbent to incumbent. Note in step 2 of the algorithm that the \( \lambda_{it} \) chosen to increase always corresponds to an \( X_{it} = 0 \). Since the contribution of \( \lambda_{it} \) to the objective function is \( \lambda_{it} q_{it} X_{it} = 0 \), and no other components change between successive incumbents, the result is shown.

Given our initial incumbent \((y,0,w)\) and utilizing the proof of Theorem 4-6, the following corollary is also clear.

**COROLLARY 4-1**: For any \((y,w)\) feasible in \((SPx-2,D)\) there exists a \((y,\lambda,w)\) feasible in \((SPx,D)\) such that the two solutions have equal objective function values in their respective problems; i.e., \( \lambda x = 0 \).

Recall our assumption that \( p \leq q = Q1 \forall (p,v) \in Y_{fp} \) implies any constraint of constraint set (4-3) with \( X_{it} = 1 \) is satisfied for any convex combination of elements of \( Y_E \). Hence all such constraints can be eliminated from \((SPx)\) without affecting its feasible region. The elimination of these constraints is equivalent to pricing them out in a Lagrangian relaxation.
with the Lagrangian prices equal to zero. Since \((SPx)\) and \((SPx,D)\) are the LP duals of each other, the elimination of the constraints of (4-3) with \(X_{it} = 1\) is equivalent to presetting the corresponding \(\lambda_{it} = 0\) in \((SPx,D)\). Since \(v(SPx) = v(SPx,D)\) is not altered by this process, we conclude that there always exists an optimal solution \((\gamma, \lambda, \omega)\) to \((SPx,D)\) with \(\lambda x = 0\) when the Benders subproblem is feasible. (We assume the Benders subproblem is feasible whenever we refer to an optimal solution to either of the dual problems \((SPx,D)\) or \((SPx-2,D)\) throughout the remainder of this section.)

This result is also attainable by noting that \(v(SPx) = v(SPx-2) = v(SPx,D) = v(SPx-2,D)\) and employing Corollary 4-1 with any optimal solution \((\gamma, \omega)\) to \((SPx-2,D)\). Therefore,

**COROLLARY 4-2:** If \((\gamma, \omega)\) is optimal in \((SPx-2,D)\), then there exists \(\lambda\) such that \((\gamma, \lambda, \omega)\) is optimal in \((SPx,D)\) with \(\lambda x = 0\). Hence there is always an optimal solution \((\gamma, \lambda, \omega)\) to \((SPx,D)\) such that \(\lambda x = 0\). □

If, for every constraint of (4-3) with \(X_{it} = 1\), there is an optimal solution to \((SPx)\) with slackness in the constraint, then linear programming complementary slackness implies \(\lambda_{it} = 0\), and hence \(\lambda x = 0\), in every optimal solution to \((SPx,D)\). On the other hand, if every optimal solution to \((SPx)\) has no slackness in some constraint \(it\) of (4-3) with \(X_{it} = 1\), then it is possible that an optimal solution \((\gamma, \lambda, \omega)\) to \((SPx,D)\) exists with \(\lambda_{it} > 0\), and hence \(\lambda x > 0\).

The diagonal matrix \(Q\) can always be constructed to guarantee slackness in constraints of (4-3) when \(x = 1\). However, we observed in Section 1.2 that it is better from a modelling perspective to have the \(q_{it} s\) as small as possible. From a practical standpoint, we have limited information on how to set the \(q_{it} s\) as small as possible such that \(p \leq q\) is satisfied for all elements of \(Y_{fp}\). The implication is that the \(q_{it} s\) will not be tight bounds for the vast majority of the elements of \(Y_{fp}\). Hence slackness will hold virtually all the time for constraints of (4-3) with \(X_{it} = 1\), and \(\lambda x\) will be zero in every optimal solution to \((SPx,D)\).
THEOREM 4-7: If \((y,A,w)\) is feasible in \((SPx,D)\), \(Ax = 0\), and \((p,v) \in Z_{E,x}\), then \(Ap = 0\).

PROOF: Since \(Ax = 0\), \(X_{xt} = 1\) implies \(A_{tt} = 0\) and therefore \(A_{tt}p_{tt} = 0\). If \(X_{tt} = 0\), \(P_{tt} = 0\) by the definition of \(Z_{E,x}\), and again \(A_{tt}p_{tt}\) is zero. □

Corollary 4-1 shows that any feasible solution to \((SPx-2,D)\) can be expanded to a feasible solution to \((SPx,D)\) by the addition of some nonnegative \(A\) vector. We next show the counterpart to this result.

THEOREM 4-8: If \((y,A,w)\) is feasible in \((SPx,D)\) and \(Ax = 0\), then \((y,w)\) is feasible in \((SPx-2,D)\).

PROOF: Since \((y,A,w)\) is feasible in \((SPx,D)\), it satisfies constraint set (4-7). Also, \(Ap \geq 0\) ∀ \((p,v) \in Y_E\) while \(Ap = 0\) ∀ \((p,v) \in Z_{E,x}\) by Theorem 4-7. For any \((p,v) \in Z_{E,x}\) we therefore have

\[ w \leq hv + yBp + Ap = hv + yBp \] (4-18)

where the inequality follows from (4-7) and the equality from Theorem 4-7. Since \((p,v)\) is an arbitrary element of \(Z_{E,x}\) and \(Bp = Bxp\) where \((p,v) = g((p,v))\), we see that constraint set (4-15) of \((SPx-2,D)\) is satisfied by \((y,w)\). Since constraint sets (4-8) and (4-16) are equivalent, \((y,w)\) is feasible in \((SPx-2,D)\). □

Observe that the \(Ax = 0\) property implies the two vectors of Theorem 4-8 have equal objective function values in their respective problems. In particular, for any optimal solution \((y,A,w)\) to \((SPx,D)\) with \(Ax = 0\), which always exists, we have \((y,w)\) as an optimal solution to \((SPx-2,D)\). Note that Theorem 4-8 does not necessarily hold when we omit the \(Ax = 0\) condition in the statement of the theorem. This is because \(Ap\) can be positive for some feasible solution \((y,A,w)\) to \((SPx,D)\) and some \((p,v) \in Z_{E,x}\). If \(Ap > 0\), then \(yBx\) can be strictly less than \(w\), and this implies \((y,w)\) is infeasible in \((SPx-2,D)\).
For each \((y, w) \in S_x\), we define

\[ S_{(y, w), x} = \left\{ (y, \lambda, w) : (y, \lambda, w) \in S, \lambda x = 0 \right\}. \]

From the analysis done so far in this chapter, we can state the following: \(S_{(y, w), x}\) is nonempty \(\forall (y, w) \in S_x\). Clearly the sets \(S_{(y, w), x}\) are mutually exclusive. Moreover, it is not necessarily true that

\[ \bigcup_{(y, w) \in S_x} S_{(y, w), x} = S. \]

Also, for each \((y, w) \in S_x\), \((y, w)\) has the same objective function value in \((SPx, 2, D)\) as each element of \(S_{(y, w), x}\) has in \((SPx, D)\).
4.5 A Revised Feasibility Algorithm

We have observed that feasibility algorithm 1 for deriving a feasible solution to \((SPx,D)\) from a feasible solution to \((SPx-2,\mu)\) assumes an explicit knowledge of \(Y_E\). In all likelihood we have no such knowledge. Therefore, from a practical standpoint the algorithm must be revised. The revision entails using mathematical programming to find \((p,v) \in V_x\) of step 2 of feasibility algorithm 1. Specifically, we search for the element \((p,v)\) of \(V_x\) whose corresponding constraint in (4-7) gives the maximum variation from feasibility. To find this \((p,v)\) for the incumbent \(s=(\gamma,\lambda,\omega)^s\), we solve the LP

\[
\nu(SP-s) = \min hv + (\gamma^sB + \lambda^s)p \quad (SP-s)
\]

\[
s.t. (p,v) \in Y_{IP}.
\]

The problem \((SP-s)\) can be solved by the special structure algorithm of Section 3-6, because \(Y_{IP}\) is assumed to be defined by a totally dynamic system. If \(\nu(SP-s) \geq \omega^s\) then, from (4-7), \(V_x\) is empty and \((\gamma,\lambda,\omega)^s\) is feasible and optimal in \((SPx,D)\). If \(\nu(SP-s) < \omega^s\), \((\gamma,\lambda,\omega)^s\) is not feasible for \((SPx,D)\), and the optimal solution \((p,v)\) to \((SP-s)\) is an element of \(V_x\) for which the feasibility violation \(v_s = \omega^s - \nu(SP-s)\) in (4-7) is greatest.

Note that the revised incumbent which corrects this maximum feasibility violation – and ensures the feasibility of the new incumbent for the constraint of (4-7) corresponding to the element generated by \((SP-s)\) – is not guaranteed to be feasible in all constraints of (4-7). One might erroneously assume that the revised incumbent is feasible, because the element of \(V_x\) with the maximum feasibility violation has been made feasible. Note, however, that \(\lambda_{it}\) is increased by \(v_s/P_{it}\) in step 2 of the algorithm for some \(t\) where \(P_{it} > 0\) and \(X_{it} = 0\). Therefore, in light of (4-7), only those elements \((p,v)^m \in V_x\) where the feasibility violation of the constraint
corresponding to \((p,v)^m\) is less than or equal to \(v_{ij} P_{it}^m / P_{it}\) will be eliminated from \(V_x\) by the new incumbent.

If \(V_x\) is unknown to us, the second procedure of step 3 which eliminates unnecessary elements of \(V_x\) due to the creation of a new incumbent is not possible. Moreover, the procedure is unnecessary, since we are now using \((SP-s)\) to find the maximum variation from feasibility; \((SP-s)\) will determine whether or not \(V_x\) is an empty set. We present a revision of feasibility algorithm 1 for handling the case where \(Y_E\) and \(V_x\) are unknown.

**Feasibility Algorithm 2**

1. **Initialization**: \((y,\lambda,\omega) \leftarrow (y,0,\omega)\).

2. **Test for end of problem**: Solve \((SP-s)\). If \(v((SP-s)) \geq \omega_s, V_x\) is empty and we're done. Otherwise go to step 2.

3. **Incumbent revision**: Use the optimal solution \((p,v)\) to \((SP-s)\) as an element of \(V_x\). Set \(v_{st}\) and alter \(\lambda_{st}\) as before to create a new incumbent. Go to step 1.

We have shown how any feasible solution to \((SPx-2,D)\) can be augmented by some \(\lambda \geq 0\) vector to a feasible solution of \((SPx,D)\) with identical objective function value. Hence, to generate cuts for the Benders master problem of the nested decomposition algorithm, the following strategy is available: Set up the Lagrangian problem of the Benders subproblem by relaxing only the facility capacity constraints in the Benders subproblem. Hence the Lagrangian problem is

\[
v(SSP: x, y) = \min \left[ h v + yB_x p_x \right] \quad (SSP: x, y)
\]

\[s.t. \quad (p_x, v) \in Y_{fp,x}.
\]
Use the technique deemed most appropriate for generating "good" Lagrangian price vectors $\mathbf{y}$. When a suitable $\mathbf{y}$ vector is found, use feasibility algorithm 2 to expand $(\mathbf{y}, w)$ to an equi-cost element of $S$, a feasible region that is $\mathbf{x}$-independent. The expanded vector $(\mathbf{y}, \lambda, w)$ can then be used to generate a master problem cut.

Therefore, the original Lagrangian problem ($SSP: \mathbf{x}, \mathbf{y}$) looks for suitable Lagrangian vectors over the region $S_\mathbf{x}$. The vectors $(\mathbf{y}, w)$ generated are always feasible relative to the constraints of (4-7) corresponding to the elements of $Z_{E, \mathbf{x}}$, without any components of $\lambda$ having to be positive, because $Z_{E, \mathbf{x}}$ is equivalent to the set of extreme points $Y_{E, \mathbf{x}}$ of $S_\mathbf{x}$. The possibility of $(\mathbf{y}, 0, w)$ being infeasible relative to the remaining constraints of (4-7) is handled by ($SP-s$). This routine searches for extreme points of $S$ from the set $Y_E \cdot Z_{E, \mathbf{x}}$ that have their corresponding constraint in (4-7) violated by the incumbent and adjusts the dual prices on the fixed charge constraints ($\lambda$) accordingly. Again, both routines can employ the special structure algorithm for totally dynamic systems in carrying out their search for extreme points of $Y_{IP, \mathbf{x}}$ and $Y_{IP}$. Note that the updating of the incumbent $(\mathbf{y}, \lambda, w)_e$ involves only one simple division and addition.
4.6 Cuts for the Benders Master Problem

An important question is whether the procedure developed in the last two sections is necessary; that is, must we extend a feasible solution of \((SP\mathbf{x},2,D)\) to a feasible solution of \((SP\mathbf{x},D)\) in order to generate a valid cut for the Benders master problem? Is there a reason why the optimal solution to \((SP\mathbf{x},2,D)\) cannot be used directly to generate this cut? We now resolve this issue.

No matter which of the two formulations is used to generate a master problem cut, the cut, as is always the case in Benders' decomposition, is derived from the Benders subproblem's objective function. If \((SP\mathbf{x},D)\) is used, the solution \((\mathbf{y},\lambda,\omega)^k\) found by the Benders subproblem at iteration \(k\) of the Benders algorithm is embedded in the objective function

\[
f\mathbf{x} + \mathbf{y}^k \left[ \mathbf{S}\mathbf{x} - \mathbf{a} \right] - \lambda^k \mathbf{Q}\mathbf{x} + \omega^k
\]

(4–24).

The function is then used to generate the cut

\[
v \geq f\mathbf{x} + \mathbf{y}^k \left[ \mathbf{S}\mathbf{x} - \mathbf{a} \right] - \lambda^k \mathbf{Q}\mathbf{x} + \omega^k
\]

(4–25).

The cut is added to the constraint set of the Benders master problem. Notice \(\mathbf{x}\) is the variable vector in (4–25). This contrasts with (4–6), the objective function of the Benders subproblem from which it was derived, in which \((\mathbf{y},\lambda,\omega)\) is the variable vector and \(\mathbf{x}\) is fixed. Assume \((\mathbf{y},\lambda,\omega)^k \in S_{(\mathbf{y},\omega)^k}\) where \((\mathbf{y},\omega)^k \in S_{\mathbf{x}^k}\) and \(\mathbf{x}^k\) is the vector of setups for the \(k\)th iteration of the Benders algorithm. If \((SP\mathbf{x},2,D)\) is used to generate cuts, then \((\mathbf{y},\omega)^k\) can be embedded in the function

\[
f\mathbf{x} + \mathbf{y}^k \left[ \mathbf{S}\mathbf{x} - \mathbf{a} \right] + \omega^k
\]

(4–26),
and the cut generated by the solution \((\gamma, \omega)^k\) to \((SPx^k-2,D)\) is

\[
v \geq f^k + \gamma^k \left[ Sx - a \right] + \omega^k \tag{4-27}\]

The functions (4-24) and (4-26) have equal values when both are evaluated at \(x^k\), because \((\gamma, \lambda, \omega)^k \in S_{(\gamma, \omega)^k}\) implies \(\lambda^k x^k = 0\). In general, there is no guarantee that the two constraints have equal values for arbitrary 0-1 \(x\) vectors. In fact, we see (4-24) must be less than or equal to (4-26) because \(\lambda^k x \geq 0\) for any 0-1 \(x\) vector.

We now consider the Benders master problem at iteration \(k\) of the nested decomposition algorithm. For simplicity, we ignore constraints generated from the extreme rays of the feasible region of the dual to the Benders subproblem LP. The role of these rays is to prevent previously generated, infeasible subproblems from being generated again. This is accomplished by making the setup vector \(x\) that generated the subproblem infeasible in a master problem constraint constructed from one of the rays. Thus, for purposes of clarity, we assume that the \(k\) previously generated Benders subproblems are all feasible.

We begin by assuming the Lagrangian subproblem of the Benders subproblem includes only the facility capacity constraints. If \(x^i, i = 1,\ldots,k\) represents the setup vector at the \(i\)th Benders iteration, then the master problem contains a cut generated from an element \((\gamma, \omega)^i\) of each of the regions \(S_{x^i}, i = 1,\ldots,k\). The master problem is thus written as

\[
\nu(MP - 2^k) = \min \nu^k \tag{MP - 2^k}
\]

\[
s.t. \quad \nu^k \geq f^i + \gamma^i \left[ Sx - a \right] + \omega^i \quad i = 1,\ldots,k
\]

\[X_{it} \in \{0,1\}.
\]
Now, for each \((y, w)^t\), \(i = 1, \ldots, k\), let \((y, \lambda, w)^t\) be any element of \(S_{(y,w)^t} \subseteq S\). We can therefore create the alternate master problem

\[
v(MP^k) = \min v^k \quad (MP^k)
\]

\[s.t. \quad v^k \geq fx + y^i \left[ Sx - a \right] - \lambda^i Qx + w^i \quad i = 1, \ldots, k
\]

\[X_{it} \in \{0, 1\}.
\]

For either master problem, the task at hand is to find the \(x\) vector that minimizes the maximum of the right-hand-side values of the \(k\) cuts. For \((MP^k)\), the objective function value \(v(MP^k)\) of this "minimax" problem is known to be a lower bound on the optimal objective function value of the global problem. The value of the function (4-24) is never greater than (4-26) for any pair of vectors \((y,w)^t \in S_x\) and \((y,\lambda, w)^t \in S_{(y,w)^t}\), and setup vector \(x\). Hence the maximum of the right hand side of the \(k\) cuts must be at least as great for \((MP^k)\) as it is for \((MP^k)\), for any fixed \(x\). Consequently, \(v(MP^k)\) is at least as great as \(v(MP^k)\).

The question is: does this matter? Can erroneous results be generated by using \((MP^k)\) as the Benders master problem? Is it true that \(v(MP^k)\) is not necessarily a lower bound on the global optimal objective function value? To answer these questions, we consider a relatively simple example that illustrates how erroneous results can be obtained when \((MP^k)\) is used. The implication is that generating cuts from the "efficient" dual \((SPx^k,D)\) is not sufficient for generating valid Benders master problem constraints. Further work, such as using feasibility algorithm 2, is necessary to extend \((y, w)^k\) to \((y, \lambda, w)^k\). This latter vector can then be used to create (4-25) as a master problem cut.

Our example has only one item in the product structure. We have a three period planning horizon and one facility that produces the item. The minimum lead time for production of the
item is one period. All avoidable costs are period-independent. Third period variables are eliminated because they are unnecessary. The relevant data is as follows:

1) Fixed setup cost to produce the item is $50.
2) Inventory holding cost is $8 per unit.
3) Overtime cost is $30 per unit.
4) Demands for the item in periods 1, 2, 3 are 0, 5, 10 respectively.
5) A setup uses 2 units of facility capacity in any time period.
6) A unit of item production uses up 3 units of facility capacity in any time period.
7) Facility capacity is 100 units in any time period.

Hence the mixed-integer programming formulation of the problem is

\[ \min \left[ 8I_1 + 8I_2 + 30O_1 + 30O_2 + 50X_1 + 50X_2 \right] \]

\[-I_1 = 0 \]

\[ I_1 + P_1 - I_2 = 5 \]

\[ I_2 + P_2 = 10 \]

\[ 3P_1 + 2X_1 - O_1 \leq 100 \]

\[ 3P_2 + 2X_2 - O_2 \leq 100 \]

\[ P_1 - 15X_1 \leq 0 \]

\[ P_2 - 10X_2 \leq 0 \]
\[ I_1, I_2, P_1, P_2, O_1, O_2 \geq 0 \quad X_1, X_2 \in \{0,1\}. \]

Note that the \( q \) vector \((15, 10)\) is constructed with each component as small as possible such that \( p \leq q \ \forall \ (p,v) \in Y_{IP} \). The optimal strategy is to produce the item "just in time" and avoid any inventory holding cost. No overtime cost is incurred by this strategy. Thus the optimal solution is \((P_1, P_2, I_1, I_2, O_1, O_2, X_1, X_2) = (5, 10, 0, 0, 0, 0, 1, 1)\) with a cost of $100.

We now follow the sequence of steps resulting from the use of nested decomposition in solving our simple problem. First, we use \((SPx)\) as the master problem in the Dantzig-Wolfe algorithm for solving the Benders subproblem. Assume \(x^1 = (1,1)\), which means production is allowed in both time periods. In the Dantzig-Wolfe subproblem, the generation of the extreme point \((p,v)^1 = (P_1, P_2, I_1, I_2)^1 = (5, 10, 0, 0)\) of \(Y_{IP}\) allows the D-W algorithm to determine that no further extreme points of \(Y_{IP}\) are necessary to solve the Benders subproblem. The capacity availability, adjusted for \(x^1\) (i.e., \(a - Sx^1\)), is \((98, 98)\), and the per-unit capacity usage of \((p,v)^1\) is \((15, 30)\). The D-W master problem with \((P_1, P_2, I_1, I_2)^1\) as the only known (and needed) element of \(Y_E\) is:

\[
100 + \min \left| 0a_1 + 30O_1 + 30O_2 \right| \quad (SPx^1)
\]

\[ s.t. \quad 15a_1 - O_1 \leq 98 \]

\[ 30a_1 - O_2 \leq 98 \]

\[ 5a_1 \leq 15 \]

\[ 10a_1 \leq 10 \]

\[ a_1 = 1. \]
The optimal solution \((a_1, O_1, O_2) = (1, 0, 0)\) has cost \$100 and an optimal dual solution 
\((y, \lambda, w)^1 = (y_1, y_2, \lambda_1, \lambda_2, w)^1 = (0, 0, 0, 0, 0)\). The vector \((y, \lambda, w)^1\) is used to generate the first constraint for the Benders master problem, namely \(v \geq 50X_1 + 50X_2\).

We now run through the integer vectors \(x^k\) and dual extreme points \((y, \lambda, w)^k\) generated by the nested decomposition algorithm:

**Iteration 2:** \(x^2 = (0, 0)\). This setup vector produces an infeasible subproblem. We assume a constraint is generated for the Benders master problem which makes \(x = (0, 0)\) infeasible in any future master problems.

**Iteration 3:** Either \(x = (1, 0)\) or \(x = (0, 1)\) is optimal. The latter produces an infeasible subproblem. Without loss of generality assume \(x^3 = (1, 0)\). The Dantzig-Wolfe subproblem generates \((P_1, P_2, I_1, I_2)^2 = (15, 0, 0, 10)\) as an extreme point of \(Y_{IP}\) to be used in the D-W master problem. It is the only D-W master problem column generated in solving the Benders subproblem. The Benders subproblem returns \((y_1, y_2, \lambda_1, \lambda_2, w)^2 = (0, 0, 0, 8, 80)\) as optimal in \((SPx^3, D)\) with cost \$130. The new constraint generated for the Benders master problem is \(v \geq 50X_1 + 50X_2 - 80X_2 + 80\).

**Iteration 4:** \(x^4 = (1, 1)\). The Benders subproblem and master problem have identical objective values. \((P_1, P_2, I_1, I_2, O_1, O_2, X_1, X_2) = (5, 10, 0, 0, 0, 1, 1)\) is determined optimal.

Now we follow the sequence of steps resulting from the use of the nested decomposition algorithm in solving the simple problem when \((SPx-2), (SPx-2, D)\) are used as the master problem primal, dual pairs in the Dantzig-Wolfe algorithm. Also, the optimal solutions obtained for \((SPx-2, D)\) are not extended to optimal solutions to \((SPx, D)\) but are used directly in forming Benders master problem constraints.
**Iteration 1:** Again, \( \mathbf{x}^1 = (1, 1) \) and \((p,v)^1 = (P_1, P_2, I_1, I_2)^1 = (5, 10, 0, 0)\) is generated by the Dantzig-Wolfe subproblem to confirm the optimality of \((\mathbf{y}, w)^1 = (\gamma_1, \gamma_2, w)^1 = (0, 0, 0)\) in \((SPx^1, D)\). Note the shortened dual vector due to the removal of \(\lambda\). The optimal cost is again \$100 and the same constraint for the Benders master problem as before is generated.

**Iteration 2:** Same as before.

**Iteration 3:** Basically the same as before. The only difference is that \((p,v)^1\), generated by the D-W subproblem in **Iteration 1**, is not an element of \(Y_{IP,x}\). Therefore \((p,v)^1\) does not correspond to a column in the current D-W master problem. Again, only \((P_1, P_2, I_1, I_2)^2 = (15, 0, 0, 10)\) is generated in solving the Benders subproblem. The Benders subproblem returns \((\gamma_1, \gamma_2, w)^3 = (0, 0, 80)\) as optimal in \((SPx^3, 2, D)\), again with cost \$130. The new constraint generated for the Benders master problem is \(v \geq 50X_1 + 50X_2 + 80\). Note how this cut differs from the previous **Iteration 3** cut because of the lack of the \(\lambda^3Qx\) term here. Also, if we expand \((\gamma_1, \gamma_2, w)^3\) to \((\gamma_1, \gamma_2, \lambda_1, \lambda_2, w)^3 = (0, 0, 0, 0, 80)\), the latter vector is infeasible in \((SPx, D)\). Specifically, it violates the constraint

\[
\quad w \leq h v^1 + (\gamma B + \lambda) p^1
\]

where \((p,v)^1\) is the D-W subproblem solution generated in iteration 1. Observe that \((p,v)^1 \in Y_{E-Z_{E,x^3}}\).

**Iteration 4:** \(x^4 = (1, 0)\). The Benders subproblem and master problem have identical objective values. \((P_1, P_2, I_1, I_2, O_1, O_2, X_1, X_2) = (15, 0, 0, 10, 0, 0, 1, 0)\) is erroneously reported as optimal with cost \$130.
We have noted that using \((MP-2^k)\) in place of \((MP^k)\) implies \(\nu(MP-2^k) \geq \nu(MP^k)\). We also speculated that this phenomenon could lead to problems if \((SPx-2,D)\) is used to generate master problem cuts. Our simple example illustrates that blatant errors are indeed possible. We refer the reader to Figures 4-1 and 4-2. What happens in our example is that the right-hand-side value of the cut generated during iteration 3 increases in value from 100 to 180 when \(x = (1,1)\) with the elimination of the \(-AQx\) term in the \(h\) function. As a result, the Benders algorithm, if \((MP-2^k)\) is used as the master problem, concludes that the minimum production schedule attainable with \(x = (1,1)\) can have a cost no less than $180. In fact, the minimum cost attainable with this vector is $100, which also happens to be optimal in the global problem. The problem is, the increase in \(h\) gives the impression that \((y,\lambda,w)\) does not maximize \(h\) given \(x=(1,1)\). The $180 value is well above the actual minimum value obtainable in the primal Benders subproblem with \(x = (1,1)\). As a result, \(x=(1,0)\) gives the appearance of being preferrable to \(x=(1,1)\) in the Benders master problem.

Hence, expansion to an optimal solution of a dual region which is independent of the variables fixed in the Benders master problem is necessary. The work that established feasibility algorithms 1 and 2 is therefore of practical as well as theoretical interest.

We have shown the equivalence of the sets \(Z_{E,x}\) and \(Y_{E,x}\) in Section 4.3. We have also seen that only \(a_i\) variables corresponding to elements of \(Z_{E,x}\) can be positive in the primal D-W formulation \((SPx)\). Perhaps, then, we can generate elements of \(Z_{E,x}\) exclusively to return to \((SPx)\) by minimizing the objective function \((y^kB + \lambda^k)x_p + hv\) over the region \(Y_{IP,x}\). Optimal extreme points of \(Y_{E,x}\) are readily extendable to their counterparts in \(Z_{E,x}\). The problem \((SPx)\) is thus solved by generating only “usable” elements of \(Y_E\). Since the fixed charge constraints are retained in the D-W master problem, we would thus be generating dual vectors of the form \((y,\lambda,w)\).

While employing this strategy certainly solves the Benders subproblem and obtains a vector of the right dimensions to return to the Benders master problem, the generated vector \((y,\lambda,w)\) is not guaranteed to be feasible in a region which is independent of \(x\); that is, the vector
Figure 4-1: Cut values using $(MP^k)$

$(y,\lambda,w)$ is not necessarily an element of $S$. This proposed scheme disallows a priori the generation of elements of $Y_{E,Z_{E,x}}$ and thus effectively presets the corresponding $a_i$ variables in the primal D-W master problem to zero. That is, the D-W "full" master problem (i.e., the master problem in which all columns are known) contains only elements of $Z_{E,x}$. Presetting primal variables is equivalent to pricing out the corresponding constraints in the dual via Lagrangian relaxation: the price on each constraint is the preset value of the corresponding primal variable. Thus, presetting primal variables at value zero is equivalent to simply eliminating the corresponding dual constraints. The resultant dual optimal vector $(y,\lambda,w)$ is optimal in the relaxed dual region, but is not necessarily feasible in $S$, the full dual feasible region. The vector $\lambda$ can always be preset to zero without affecting optimality in this scheme. Moreover, it is clear
that employing such a strategy leaves us open to the types of errors, discussed in this section, that can occur when Benders cuts are generated from x-dependent regions.

We can interpret the use of the alternate D-W formulation followed by the feasibility algorithm in the following way: we have seen how the original D-W formulation is "inefficient" because the subproblem will return columns to the primal master problem that can never be positive. The alternate formulation alleviates this situation and also eliminates the fixed charge constraints from the D-W formulation.

The feasibility algorithm begins by expanding the dual optimal solution (y,λ) of the alternate formulation (SPx-2,D) to (y,0,λ). The vector (y,0,λ) has the following interpretation: if only constraints corresponding to elements of Z_{E,x} are to be considered in the original D-W master problem formulation, then the shadow prices on the fixed charge constraints (the λ_{it}'s)
can all be set to zero. This is because \( X_{it} = 0 \) implies \( P_{it} = 0 \) \( \forall (p,v) \in Z_{E,X} \), and so there is no "economic incentive" to increase the right hand side \( q_{it}X_{it} \) of the corresponding fixed charge constraint beyond zero.

We have seen, however, that \((\gamma,0,\lambda)\) is not necessarily feasible in \((SPx,D)\). Infeasibility results from \((\gamma,0,\lambda)\) violating a constraint of the form \( w \leq hv^i + (\gamma B + \lambda)p^i \) where \((p,v)^i \in Y_E - Z_{E,X} \). This infeasibility implies the corresponding variable \( a_i \) of \((SPx)\) has negative reduced cost when only variables corresponding to elements of \( Z_{E,X} \) are allowed in the primal basis. This negative reduced cost implies the optimal objective function value could be lowered if \( a_i \) could be pivoted into the current basis at a positive value. But \((p,v)^i\)'s inclusion in \( Y_E - Z_{E,X} \) prevents this, since there exists some \( X_{it} = 0 \) and \( P_{it} > 0 \). If, for all \( it \) such that \( X_{it} = 0 \) and \( P_{it} > 0 \), \( q_{it}X_{it} \) is raised incrementally by some \( \varepsilon > 0 \), \( a_i \) could (at least if overtime is unbounded) be pivoted into the basis at positive value, thereby reducing the objective function value. Loosely speaking, it may therefore be inappropriate that shadow prices on all fixed charge constraints are zero when some constraint corresponding to an element of \( Y_E - Z_{E,X} \) has a negative reduced cost, since there is a possible economic advantage to be gained by raising certain right hand side elements \( q_{it}X_{it} \) from zero. The feasibility algorithm, thus adjusts shadow prices on certain fixed charge constraints with \( X_{it} = 0 \) to account for this situation.

Clearly the original D-W formulation \((SPx)\) determines the appropriate shadow price vector \( \lambda \) on the fixed charge constraints. The relative advantage of our alternate approach is that it attains primal optimality by using only needed data (i.e., columns generated from the set \( Z_{E,X} \)) and then adjusts the \( \lambda \) vector to account for elements of \( Y_E - Z_{E,X} \). This adjustment is done in an efficient manner using the totally dynamic constraint set algorithm and thus circumvents relying on simplex pivots in the large \((SPx)\) formulation to attain dual feasibility.

In general, no matter what method is used to attack the dual problem implied by relaxing constraints in the Benders subproblem, it is clear that relaxing only the facility capacity constraints is a better method for getting good dual solutions when one is not concerned with generating valid Benders master problem cuts. Retaining the setup information in the
Lagrangian subproblem allows us, in practice, to generate Lagrangian solutions very efficiently when $Y_{IP}$ can be represented by a totally dynamic system. In Chapter 5 we discuss in detail how the number of constraints and variables can potentially be greatly reduced when the fixed charge constraints are retained in the Lagrangian subproblem.

If the setup information is not present in the Lagrangian subproblem, then we are faced with constantly solving Lagrangian subproblems over the full region $Y_{IP}$, even though many of the production variables may not be allowed to be positive because of setup restrictions. In short, we leave it to the $\lambda_{it}$ prices to attempt to “force” production variables $P_{it}$ to zero when $X_{it} = 0$.

Hence it may prove to be more computationally efficient to generate a solution to $(SPx-2,D)$ — with an acceptable objective function value — by employing the dual procedure of choice, and then generating the $\lambda$ vector needed to form a valid Benders cut by solving a sequence of LPs over totally dynamic systems that represent the region $Y_{IP}$. 
4.7 Equi-Cost Dual Solutions Along Rays and Strong Benders Cuts

It is possible to generate an equi-cost solution \((\mathbf{y}, \mathbf{\lambda}, w)\) to \((SPx,D)\) from a solution \((\mathbf{y}, w)\) to \((SPx-2,D)\) by methods other than feasibility algorithm 2. Later in this section we present one such method. The question then arises: are some of the solutions to \((SPx,D)\) that are generated from solutions to \((SPx-2,D)\) "better," in some sense, as generators of Benders master problem cuts? In fact, there are criteria for judging the quality of such cuts. In this section we explore the quality of the cuts we are capable of generating in relation to these criteria. To facilitate this endeavor, it is necessary to develop some properties of the rays of the convex polytope \(S\) that represents the feasible region of problem \((SPx,D)\).

It is possible to explicitly characterize the set \(R\) of rays of the convex polytope \(S\). The constraints defining \(S\) are found in constraint sets (4-7), (4-8), and (4-9). Clearly a vector \((\mathbf{y}, \mathbf{\lambda}, w)^r \in R\) must satisfy \(\mathbf{\lambda}^r \geq \mathbf{0}\) if \((\mathbf{y}, \mathbf{\lambda}, w)^r \in R\). Observe that any \((\mathbf{y}, \mathbf{\lambda}, w)^r \in R\) has \(y^r = 0\), because \(y\) is bounded in \(S\) by (4-8). We noted in Chapter 2 that if overtime is bounded in the global optimization problem, then the dual problem that arises from relaxing the facility capacity constraints and the fixed charge constraints in the Benders subproblem will have \(y\) unbounded. It is straightforward to show, however, that if any component \(y_{nt}\) of \(y\) approaches \(+\infty\) monotonically, then the corresponding sequence of Lagrangian subproblems over the I-P balance constraints produces a sequence of objective function values that converges to \(-\infty\), even when overtime is bounded. Hence, it is always possible to bound \(y\) in the dual problem without eliminating any optimal solution to the dual problem, and therefore \(S\) can always be constructed with upper bounds on \(y\) without eliminating any optimal solution.

These facts combined with (4-7) imply \((\mathbf{y}, \mathbf{\lambda}, w)^r \in R\) must also satisfy

\[ w^r \leq \min_{i \in \{1, \ldots, L\}} \mathbf{\lambda}^r \mathbf{p}^i \]

where \(Y_E = \left\{ (\mathbf{p}, \mathbf{y})^i \right\}_{i=1}^L \) (4-28).
Moreover, the necessary conditions we have outlined are sufficient, and hence any vector 
\((\mathbf{y}, \lambda, \mathbf{w})^r\) satisfying these conditions is an element of \(R\). For any \(\lambda \geq 0\), let \(v_\lambda = \min_{i \in \{1, \ldots, L\}} \lambda p_i\). For a given vector \(\lambda \geq 0\), it follows that \((\mathbf{y}, \lambda, \mathbf{w})^r = (0, \lambda, \mathbf{w})\) is a ray of \(S\) if and only if \(\mathbf{w} \leq v_\lambda\). Hence an explicit characterization of the set of (nonzero) rays \(R\) is given as follows:

**Theorem 4.9:** \(R = \{(\mathbf{y}, \lambda, \mathbf{w})^r : \mathbf{y}^r = 0, \lambda^r \geq 0, \mathbf{w}^r \leq v_\lambda\} \cup \{(\mathbf{y}, \lambda, \mathbf{w})^r : (\mathbf{y}, \lambda, \mathbf{w})^r = 0\}\). □

\(R\) is clearly a nonempty set.

Let \(h((\mathbf{y}, \lambda, \mathbf{w})^r : \mathbf{x})\) represent the objective function (4.6) of \((SP_x, D)\). There exists \((\mathbf{y}, \lambda, \mathbf{w})^r \in R\) such that \(h((\mathbf{y}, \lambda, \mathbf{w})^r : \mathbf{x}) > 0\) if and only if there exists a sequence of elements of \(S\) with objective function values unbounded from above in \((SP_x, D)\). Such a sequence exists, according to LP strong duality, if and only if \((SP_x)\) is infeasible.

We have observed that \((SP_x)\) is always feasible for model (EMRP), because we always have the option of not producing anything in the extended model. Therefore it must be that \(h((\mathbf{y}, \lambda, \mathbf{w})^r : \mathbf{x}) \leq 0 \ \forall \ (\mathbf{y}, \lambda, \mathbf{w})^r \in R\) and all 0-1 \(\mathbf{x}\) vectors in (EMRP). We can also verify this fact algebraically: The vector \((\mathbf{p}, \mathbf{v}, \mathbf{w}) = (0, 0, \mathbf{w}^T + 1)\) – where \(\mathbf{w}^T + 1\) is a 0-1 vector, and a component \(W_{it}\) of \(\mathbf{w}^T + 1\) is 1 if and only if \(t = T + 1\) – is a solution of \(Y_E\) in (EMRP). In fact, \((0, 0, \mathbf{w}^T + 1)\) is the solution that represents the decision not to produce anything. Hence (4.28) implies \(\mathbf{w}^r \leq 0\) for any \((\mathbf{y}, \lambda, \mathbf{w})^r \in R\) in (EMRP), because \(\lambda^r p = 0\) when \((\mathbf{p}, \mathbf{v}, \mathbf{w}) = (0, 0, \mathbf{w}^T + 1)\). Moreover,

\[
h((\mathbf{y}, \lambda, \mathbf{w})^r : \mathbf{x}) = \mathbf{w}^r - \lambda^r \mathbf{Q} \mathbf{x} \quad \forall \ (\mathbf{y}, \lambda, \mathbf{w})^r \in R \quad (4-29),
\]

and so \(h((\mathbf{y}, \lambda, \mathbf{w})^r : \mathbf{x}) \leq 0 \ \forall \ (\mathbf{y}, \lambda, \mathbf{w})^r \in R\) and all 0-1 \(\mathbf{x}\) vectors in (EMRP).

For a given \(\mathbf{x}\), we call \((\mathbf{y}, \lambda, \mathbf{w})^r \in R\) an equi-cost ray of \(S\) if \(h((\mathbf{y}, \lambda, \mathbf{w})^r : \mathbf{x}) = 0\), and we let \(R_{0x}\) be the set of all equi-cost rays. If \(R_{0x}\) is nonempty, it is clear that we can “go out on” any ray in
\( R_{0x} \) from any element \((y,\lambda,w)\) of \( S \), and each solution to \((SPx,D)\) generated in this fashion has the same objective function value as \((y,\lambda,w)\).

For any vector \( y \) that has the same length as \( x \), let \( y = [y_{x1},y_{x0}] \) be the partition of the components of \( y \) into the subvectors \( y_{x1} \) and \( y_{x0} \) such that a component \( Y_{ii} \) of \( y \) is in \( y_{x1} \) if and only if \( X_{ii} = 1 \).

**THEOREM 4-10:** \((y,\lambda,w)^r \in R_{0x}\) if and only if

1) \( y_r = 0 \)

2) \( \lambda_r \geq 0 \)

3) \( w_r = \lambda_{x1}^r q_{x1} \leq \nu_{\lambda_r} \).

**PROOF:** \((y,\lambda,w)^r \in R_{0x}\) if and only if it is an element of \( R \) with zero objective function value. From (4-29), a ray has zero objective function value if and only if \( w_r = \lambda^r Q x \). The theorem is proven by combining \( \lambda^r Q x = \lambda_{x1}^r q_{x1} \) with Theorem 4-9. \( \square \)

Note that \((y,\lambda,w)^r \in R_{0x}\) implies \( w_r \geq 0 \). Let \( V_{0x} = \{ (y,\lambda,w)^r : y_r = 0, \lambda_{x1}^r = 0, \lambda_{x0}^r \geq 0, w_r = 0 \} \).

**COROLLARY 4-3:** \( V_{0x} \subseteq R_{0x} \).

**PROOF:** For any \((y,\lambda,w)^r \in V_{0x}, w_r = \lambda_{x1}^r q_{x1} = 0\), while \( \nu_{\lambda_r} \geq 0 \) for any \( \lambda_r \geq 0 \). The result is thus shown in light of Theorem 4-10. \( \square \)

If \( x = 1 \), then it is straightforward to show that \( S_{(y,w),x} = \{ (y,0,w) \} \forall (y,w) \in S_x \) and that \( V_{0x} \) is empty. Otherwise, \( V_{0x} \) — and hence \( R_{0x} \) — are nonempty and

\[(y,\lambda,w) + \theta(y,\lambda,w)^r \in S_{(y,w),x} \forall (y,\lambda,w) \in S_{(y,w),x}, (y,\lambda,w)^r \in V_{0x}, \text{ and } \theta \geq 0 \quad (4-30).\]
Therefore, when \( x \neq 1 \) there exists an equi-cost ray to "go out on" from each \((y, \lambda, w) \in S\), because \( V_{0x} \) is guaranteed to be nonempty. Moreover, for each \((y, w) \in S_x\) the set \( S_{(y, w), x} \) is uncountably infinite and unbounded. In particular, there are always alternate optimal solution to \((SPx, D)\) along rays of \( S \). Observe that the feasibility algorithms of Sections 4.4 and 4.5 always generate a vector \((y, \lambda, w)^r = (0, \lambda^r, 0) \in V_{0x}\) in constructing \((y, \lambda, w) \in S_{(y, w), x} \) from \((y, w) \in S_x\).

**Corollary 4.4:** If \((y, \lambda, w)^r \in R_{0x}\), then \( w^r = \lambda_{x_1}^r p_{x_1} = \lambda_{x_1}^r q_{x_1} \forall (p, v) \in Z_{E,x}\).

**Proof:** Recall that \( p \leq q \forall (p, v) \in Y_{fp} \). Utilizing this fact and Theorem 4.10, we have

\[
\lambda_{x_1}^r p_{x_1} \leq \lambda_{x_1}^r q_{x_1} = w^r \leq \lambda_{x_1}^r p = \lambda_{x_1}^r p_{x_1} \quad \forall (p, v) \in Z_{E,x}
\]

(4.31)

where the last equality holds because \((p, v) \in Z_{E,x}\) implies \( P_{it} = 0 \) whenever \( X_{it} = 0 \). Therefore all inequalities can be replaced by equalities in (4.31) and the result is shown. □

**Corollary 4.5:** If \( \forall it \) such that \( X_{it} = 1 \) there exists \((p, v) \in Z_{E,x}\) such that \( P_{it} < q_{it} \), then \( V_{0x} = R_{0x}\).

**Proof:** Consider any \( it \) where \( X_{it} = 1 \). The conditions of the corollary implies there is some \((p, v)^r \in Z_{E,x}\) such that \( P_{it}^+ < q_{it}\). Let \( j(it) \) be the index of the component of \( \lambda_{x_1} \) that corresponds to the component of \( \lambda \) indexed by \( it \). Corollary 4.4 then forces \( \lambda_{x_1} \mid_{j(it)}^r = 0 \forall (y, \lambda, w)^r \in R_{0x}\), since otherwise \( \lambda_{x_1}^r p_{x_1} = \lambda_{x_1}^r q_{x_1}\). Therefore, \( \lambda_{x_1}^r = 0 \forall (y, \lambda, w)^r \in R_{0x}\), and Theorem 4.10 guarantees \( w^r = 0 \) as well for all elements of \( R_{0x}\). It follows that \( R_{0x} \subseteq V_{0x}\), and since \( V_{0x} \subseteq R_{0x}\), the two sets are equal. The corollary is therefore proven. □

**Theorem 4.11:** If \( \lambda x = 0 \) in every optimal solution to \((SPx, D)\), then \( V_{0x} = R_{0x}\).

**Proof:** If \((y, \lambda, w)\) is optimal in \((SPx, D)\) and \((y, \lambda, w)^r \in R_{0x}\), then \((y, \lambda, w) + \theta(y, \lambda, w)^r\) is optimal \( \forall \theta \geq 0 \). Hence, by utilizing the condition of the theorem, it follows that

167
\[(\lambda + \theta \Lambda^r) \mathbf{x} = \theta \Lambda^r \mathbf{x} = \theta \Lambda^r_{x^1} \mathbf{1} = 0\]

for any \(\theta > 0\). Since \(\lambda_{x^1} \geq 0\), we conclude that \(\lambda_{x^1} \rho = 0\) and \(w^r = 0\ \forall (\mathbf{y}, \lambda, w)^r \in R_{0x}\). Thus \(R_{0x} \subseteq V_{0x}\), and since \(V_{0x} \subseteq R_{0x}\), the two sets are equal. \(\square\)

We saw the following in Section 4.4: if, for every fixed charge constraint in (4-3) with \(X_{il} = 1\), there exists an optimal solution to \((SPx)\) with slackness in the constraint, then \(\lambda \mathbf{x} = 0\) in every optimal solution to \((SPx,D)\). Theorem 4-11 implies this condition is sufficient to guarantee \(V_{0x} = R_{0x}\). This condition of Section 4.4 implies, but is not implied by, the condition of Corollary 4-5. Hence Corollary 4-5 gives a less restrictive set of sufficient conditions (relative to the condition of Section 4.4) under which \(V_{0x} = R_{0x}\).

We call a ray \((\mathbf{y}, \lambda, w)^r\) of \(S\) negative indefinite if \(h((\mathbf{y}, \lambda, w)^r : \mathbf{x}) = w^r - \lambda^r \mathbf{Qx}\) is nonpositive for all 0-1 \(\mathbf{x}\) vectors, and we let \(N\) represent the set of all negative indefinite rays of \(S\). If \(N \neq R\), then any \((\mathbf{y}, \lambda, w)^r \in R - N\) has positive objective function value for some \(\mathbf{x}\) vector \(\mathbf{x}^r\). This implies \((SPx^r)\) is infeasible. Since \((SPx)\) is always feasible for model \((EMRP)\) for any setup vector, \(N = R\) in the extended model.

Note that \(V_{0x} \subseteq N\) for any \(\mathbf{x}\) because \(w^r = 0\) for any \((\mathbf{y}, \lambda, w)^r \in V_{0x}\). Hence

\[
\bigcup_{\mathbf{x} : X_{il} \in \{0,1\}} V_{0x} \subseteq N.
\]

It is straightforward to show that \((\mathbf{y}, \lambda, w)^r \in N\) if and only if \(w^r \leq 0\), and any \((\mathbf{y}, \lambda, w)^r \in N\) has a strictly negative objective function value for some 0-1 \(\mathbf{x}\) vector. Clearly \(N\) is always a nonempty set. If \(N \neq R\), then it is possible that a ray of \(S\) may have positive, zero, or negative objective function value, depending on the setup vector \(\mathbf{x}\).
We say that \((\nu, \lambda, w) \in S\) is *dominated* if there exists \((\nu, \lambda, w)^1 \in S\) that is distinct from \((\nu, \lambda, w)\) and \(h((\nu, \lambda, w)^1; x) \geq h((\nu, \lambda, w); x)\) \(\forall\) 0-1 \(x\) vectors. We call \((\nu, \lambda, w) \in S\) *efficient* if it is not dominated. From the standpoint of generating Benders master problem cuts, it is always preferable to generate a cut from an efficient element of \(S\). A cut generated from an efficient vector is "stronger" in the Benders master problem than cuts generated from the vectors the efficient vector dominates. That is, cuts generated from efficient solutions lead to better lower bounds on the optimal, global objective function value from the Benders master problems, and quicker convergence of the objective function values of the Benders master problems to the optimal, global objective function value. (See Magnanti and Wong (1981).)

An element \((\nu, \lambda, w) \in S\) is dominated if and only if there exists \((\nu, \lambda, w)^1 \in S\) such that \(h((\nu, \lambda, w) - (\nu, \lambda, w)^1; x) \leq 0\) \(\forall\) 0-1 \(x\) vectors. Clearly, if

\[
(\nu, \lambda, w) = (\nu, \lambda, w)^1 + \theta (\nu, \lambda, w)^r \text{ for some } (\nu, \lambda, w)^1 \in S, (\nu, \lambda, w)^r \in N, \text{ and } \theta > 0
\]

then \((\nu, \lambda, w)\) is an element of \(S\) that is dominated by \((\nu, \lambda, w)^1\). When (4-32) holds, we say that \((\nu, \lambda, w)\) is *dominated along a ray*. If \((\nu, \lambda, w)\) is in the relative interior of \(S\), then \(\forall (\nu, \lambda, w)^r \in N\) (4-32) holds for some \(\theta = \theta_r > 0\) because \(- (\nu, \lambda, w)^r / \| (\nu, \lambda, w)^r \|\) is a feasible direction of \(S\) at every interior point of \(S\). Since \(N\) is nonempty, every interior point of \(S\) is seen to be dominated. Hence the set of efficient points will be found on the boundary of \(S\). Since \(S\) is a convex polytope, this means that all efficient points of \(S\) lie on some facet of the polytope. If \(N = R\) (as is the case in model (EMRP)), then any boundary point that lies on a facet containing a ray of the polytope is dominated. Hence, in this case, the set of efficient points will only be found on bounded facets of the polytope. That is, any efficient point is a boundary point that can be written strictly as a convex combination of extreme points of \(S\).

We now ask whether any vector \((\nu, \lambda, w)\) generated by feasibility algorithm 2 of Section 4.5 can possibly be dominated along a ray. If we can show that this is not possible for all such
vectors, then we can at least guarantee that these vectors are boundary points of the polytope $S$ that are not dominated along a ray of $N$.

We observe that constraint set (4-7) always contains a constraint with no slack when $(\gamma, \lambda, w)$ is generated by feasibility algorithm 2. This is because the constraint considered by the algorithm at each iteration is always violated by the current incumbent before the incumbent revision, but the constraint is satisfied with no slack by the revised incumbent.

Consider any $(\gamma, \lambda, w) \in S$ generated by the feasibility algorithm. Assume there exists some $\lambda_{it} > 0$ such that there is slack in all constraints of (4-7) corresponding to the elements of $Y_F$ with $P_{it} > 0$. Let $Y_{it} \subseteq Y_F$ be the nonempty set of points that have $P_{it} > 0$. (We know that $Y_{it}$ is nonempty, because otherwise the feasibility algorithm could not have set $\lambda_{it}$ positive.) For any $(p, v) \in Y_{it}$, let $v_{(p,v)}$ be the positive slack in the constraint of (4-7) corresponding to $(p, v)$. We create a vector $\lambda^r$ by setting

$$\lambda_{it}^r = \min \{ \min_{(p,v) \in Y_{it}} \{ v_{(p,v)} / P_{it} \}, \lambda_{it} \} \text{ and all remaining components of } \lambda^r \text{ to zero.}$$

Note that $\lambda_{it}^r > 0$, so the vector $(0, \lambda^r, 0)$ is an element of $N$. Moreover,

$$(\gamma, \lambda, w) = (\gamma, \lambda - \lambda^r, w) + (0, \lambda^r, 0) \text{ and } (\gamma, \lambda - \lambda^r, w) \in S \quad (4-33).$$

The fact that $(\gamma, \lambda - \lambda^r, w)$ is feasible in $(SPx,D)$ is seen by observing that $\lambda_{it}^r$ has been set so that $\lambda_{it} - \lambda_{it}^r \geq 0$, and all constraints of (4-7) remain satisfied when $\lambda_{it}$ is reduced by $\lambda_{it}^r$. Therefore $(\gamma, \lambda, w)$ is dominated by $(\gamma, \lambda - \lambda^r, w)$ along the ray $(0, \lambda^r, 0)$.

Unfortunately, it is possible that a solution $(\gamma, \lambda, w)$ generated by feasibility algorithm 2, as the algorithm currently exists, satisfies (4-33) for some $\lambda^r$ where $(0, \lambda^r, 0) \in N$. As we have mentioned, the constraint considered by the algorithm at any iteration $k$ (corresponding to some $(p, v)^k \in Y_F$) is satisfied with no slack by the revised incumbent at the termination of iteration $k$.

Let $i_k$ be the index of the unique component of $\lambda$ that is updated at iteration $k$. It is possible that
at some iteration \( j > k \) find \( i_j \neq i_k \) and \( P_{ij} > 0 \). Hence the constraint corresponding to \((p,v)^k\) will be satisfied with slack after iteration \( j \). Moreover, it is possible that all elements of \( Y_E \) with \( P_{ik} > 0 \) are satisfied with slack after iteration \( j \), and thus will remain satisfied with slack throughout the remainder of the algorithm. We have seen that this last condition implies that (4-33) is satisfied for some \( \lambda^r \) where \((0,\lambda^r,0)\) is an element of \( N \).

We can alter feasibility algorithm 2 so that for any \( \lambda_{it} > 0 \) in the generated solution, there will always be an element \((p,v)\) of \( Y_E \) such that \( P_{it} > 0 \), and there is no slack in the corresponding constraint of (4-7). In such a situation, it is impossible to create \((\mathbf{y}, \boldsymbol{\lambda} - \lambda^r, \mathbf{w} - w^r) \in S\) from any \((0,\lambda, \mathbf{w})^r \in N \). Recall that \((0,\lambda, \mathbf{w})^r \in N\) means \( w^r \leq 0 \), so \( \mathbf{w} - w^r \geq \mathbf{w} \). In addition, the reduction of any of \( \lambda \)'s components either creates a negative component or a constraint of (4-7) that is violated when \( \mathbf{w} \) is unaltered. If \( \mathbf{w} \) is actually raised (i.e., \( w^r < 0 \)), then the violations can only become worse. We conclude that any vector created by this altered feasibility algorithm is not dominated along a ray and is therefore a boundary point of \( S \). Note that this statement includes any element of \( S \) generated by feasibility algorithm 2. It is not limited to the case where \((SPx-2,D)\) is solved to optimality and feasibility algorithm 2 creates an optimal solution to \((SPx,D)\).

We will not give the details of the necessary alterations to the feasibility algorithm that make this situation possible. We mention that the heaviest price one must pay is that the positive components of the subvector \( p_{x0} \) of all elements \((p,v) \in Y_E - Z_{E,x} \) found by the algorithm must be retained. Moreover, the algorithm may require more iterations than previously, although it will still always converge in a finite number of steps that is bounded above by the cardinality of the set \( Y_E - Z_{E,x} \). Whether this increased computational burden is worth the benefit received is an issue that can be addressed by future empirical research. Moreover, we feel future research may improve this revised algorithm or yield a superior one that performs the same task.

Consider any \((\mathbf{y}, \mathbf{w}) \in S_\mathbf{x} \) such that \((\mathbf{y}, \mathbf{w})\) creates no slack in some constraint of (4-15). Naturally, all solutions of interest to \((SPx-2,D)\) have this property. Since the right hand side of
all constraints of (4-15) are nonnegative, we therefore have $w \geq 0$. We now show that it is possible to calculate very quickly a vector $(y, \lambda, w)^r \in V_0^x$ such that $(y, 0, w) + (y, \lambda, w)^r \in S(y, w)^x$.

Recall that $(y, 0, w)$ may be infeasible for some constraint of (4-7). The goal of the scheme is to determine a vector $\lambda = [\lambda_x, \lambda_0] = [0, \lambda_0]$ such that $\lambda_x \geq 0$ and we can guarantee $(y, \lambda, w) \in S(y, w)^x$ without ever generating any element of $Y_E - Z_{E,x}$.

We begin by observing that $w$ is an upper bound on the feasibility violation

$$w - hv^i - (yB)p^i > 0$$

for any $(p, v)^i \in Y_E - Z_{E,x}$ whose corresponding constraint in (4-7) is violated by $(y, 0, w)$. This observation is true because $hv^i + (yB)p^i$ is nonnegative.

We are also interested in finding, if possible, a positive lower bound on any positive component of $p \forall (p, v) \in Y_E$. Fortunately, such a bound is possible to compute: For any item $i$ that is a finished product (i.e., $i$'s set of immediate successors $S_i$ is empty), set

$$p_i^* = \min_{i \in \{1, \ldots, n\}} \{d_{ii} : d_{ii} > 0\}$$

for model (BMRP). For (EMRP), set $p_i^*$ equal to the quantity of the order for item $i$ with the least demand. Now we define

$$p^* = \min_{i : S_i = \emptyset} \{ p_i^* \}.$$ 

It is relatively straightforward to see that $p^* > 0$ is a lower bound on any positive production variable found in any extreme point solution to a pure production problem. This is because any extreme point solution has the property that any positive production of an item is exactly enough to meet the demand for that item for some integral number of time periods. (See Chapter 3.) This observation – combined with the fact that the yield factors (the $y_i$s) are
between 0 and 1, and the \( a_{ij} \)s are positive integers – implies any demand for any item in any period is at least as great as \( p^* \). Therefore \( p^* \) is a lower bound on any production variable that is positive in an extreme point solution. Note our assumption that there is no exogenous demand for any item besides finished goods in deriving \( p^* \). Clearly the definition of \( p^* \) can be extended in a straightforward manner to the general situation in which exogenous demands are allowed throughout the product structure.

We define

\[
\lambda^* = w / p^*, \lambda_{x_0}^* = \lambda^* 1, \text{ and } \Lambda^* = [\lambda_{x_1}, \lambda_{x_0}] = [0, \lambda_{x_0}^*]
\]

where 1 is a vector of 1s with the same length as \( \lambda_{x_0} \). For any \((p,v) \in Y_E - Z_{E,x}\), there is at least one \( p_{it} > 0 \) such that \( X_{it} = 0 \). Therefore,

\[
p_{x_0} \lambda_{x_0}^* = \lambda^* p_{x_0} 1 \geq \lambda^* p^* = w.
\]

Hence adding \((y, \lambda, w)^T = (0, \lambda^*, 0)\) to \((y, 0, w)\) produces a vector \((y, \lambda^*, w)\) that is guaranteed to be feasible relative to constraint set (4-7). This is because \( p \lambda^* \) is at least as great as the feasibility violation of any constraint of (4-7) – corresponding to \((p, v) \in Y_E - Z_{E,x}\) – that is violated by the vector \((y, 0, w)\). We conclude, then, that the vector \((0, \lambda^*, 0)\) is an element of \( V_{e,x} \) such that \((y, \lambda^*, w) \in S_{(y, w), x}\).

Since we can find a vector \((y, \lambda, w) \in S_{(y, w), x}\) so quickly, we might ask why one would ever bother trying to calculate such a vector via, say, feasibility algorithm 2. We have seen that feasibility algorithm 2 must solve a sequence of LPs over totally dynamic systems in order to calculate the required vector, and therefore it must require, in general, more computational effort than the quick scheme we have just outlined.

The answer is the following: The element of \( S_{(y, w), x} \) calculated by the quick scheme sets each component of \( \lambda_{x_0}^* \) high enough to theoretically guarantee that it is unnecessary to check...
the resulting vector \((y, \lambda^*, \omega)\) against any elements of \(Y_E - Z_{E,*}\) for feasibility, relative to constraint set (4-7). In other words, we set each component high enough to cover all theoretical worst cases. In reality, it is clear that the existence of some \((0, \lambda, 0)^T \in V_{0X}\) such that \((y, \lambda^r, \omega) \in S_{y, \omega, \lambda, x}\) and \(\lambda^r \leq \lambda^*\) is quite likely. In fact, such a \(\lambda^r\) probably exists where certain (or even most) components of \(\lambda^r\) are considerably less than their counterparts in \(\lambda^*\).

The implication of this, in light of the analysis of this section, is that \((y, \lambda^*, \omega)\) is quite likely to be dominated along a ray of \(S\). Moreover, it is probably "very dominated." Hence the tradeoff is that while we can easily calculate an element of \(S_{y, \omega, \lambda, x}\), the element is likely to be a poorer Benders master problem cut than a cut generated by expending more effort, such as using feasibility algorithm 2.

Finally, we have not addressed the issue of finding solutions to \((SPx, D)\) of guaranteed to be efficient. An interesting area of future research would be to attempt to identify sufficient (and perhaps even necessary) conditions under which an element of \(S\) is efficient. Also, it would be useful to know if feasibility algorithm 2, or some variation of it, can be used to generate efficient solutions. Perhaps Magnanti and Wong's work (1981) on efficient cuts for Benders master problems can be useful in this regard.
Chapter 5 Implementation Issues

5.1 Introduction

In this chapter, we discuss the issues raised by solving totally dynamic LP subproblems of our production scheduling models, created by mathematical programming decomposition, on computers. We are specifically concerned with determining the data structures and detailed algorithms to be used in solving these LP subproblems.

The analysis of Chapter 3 shows that the general algorithm for solving LPs over totally dynamic systems is so specialized that it bears little resemblance to the pivot-oriented simplex algorithm. Our intention, then, is to combine this algorithm with the specific characteristics of our production scheduling models and produce efficient computer code and data structures for the LPs over the totally dynamic systems we isolate via decomposition. That is, we seek code and structures that are relatively compact and simple, and code that executes very quickly.

In this chapter we discuss the data structures and algorithms that we have implemented on an Apollo DN320 minicomputer in the C programming language. The specific problem addressed is a pure production problem for model (EMRP). (A numerical example is given in Section 5.2. See Section 3.4 for a discussion of pure production problems.) We assume that orders are present for finished goods only.

Hence we are concerned with solving LPs over totally dynamic systems constructed from the constraints sets (1-7) and (1-10) of model (EMRP) in Section 1.2. In addition, we are working with the subproblem created by a relaxation of only the facility capacity constraint set (1-8). In the terminology of Chapter 4, this means that we are working with the Lagrangian subproblem of the "alternate" dual problem (SPx-2,D). Since the setup information is retained in the Lagrangian subproblem, the subproblem can be formulated as
\[ \nu(\text{SSP} - \text{DW}_x) = \min \left( (\gamma B)_x p_x + hv + bw \right) \quad (5-1) \quad (\text{SSP} - \text{DW}_x) \]

\[ s.t. \quad A_x p_x + Dv - Ew = 0 \quad (5-2) \]

\[ Fw = 1 \quad (5-3) \]

\[ p_x, v, w \geq 0 \quad (5-4). \]

Constraint sets (5-2), (5-3) correspond to (1-7), (1-10) respectively. Constraint set (5-2) is altered in the usual manner by eliminating all production variables \( P_{it} \) where \( X_{it} = 0 \).

The computer implementation presented herein is designed to fully exploit this particular problem. We are especially interested in demonstrating the efficiencies to be realized by incorporating the setup information in the Lagrangian subproblem. We show how this information can be used not only to reduce the number of variables in the subproblem (i.e., the \( P_{it} \)s where \( X_{it} = 0 \) are eliminated) but also to reduce the number of constraints. Our subproblems are feasible for any set of setup decisions, because we are dealing with the extended model (EMRP) where any order can potentially be pushed out of the planning horizon.

Section 5.2 describes a numerical problem that has been used to test our computer implementation. Section 5.3 outlines a data structure found repeatedly in the implementation. Section 5.4 shows how constraints, as well as variables, can be eliminated when the the setup information is retained in the Lagrangian subproblem. The actual data structures used in the implementation are explicitly defined in Section 5.5, whereas the actual computer algorithms are presented in Section 5.6.
5.2 A Numerical Example

In this section we describe a specific numerical example that we have been using for testing purposes. As we mentioned in the introduction of the chapter, this implementation is for a pure production version of model (EMRP). Figure 5-1 describes the product structure of the test problem. The dashed-line boxes surrounding one or more production items represent production facilities that are required in the manufacturing of the enclosed items. For example, facility 2 is required for the manufacturing of items 2 and 3, while facility 3 is used to produce item 4. Figure 5-2 displays the parameters $a_{ij}$ (i.e., the number of $i$ is required to produce one item $j$).

The test problem has orders for finished goods only. There are seven orders each for items 1 and 2 and four orders for item 3. Figure 5-3 shows the periods in which each order can possibly be shipped. Observe that the problem has a twelve-period planning horizon. Also, for each order, it is possible that the order may not be shipped at all. In each period that an order can be shipped, there is a specified profit that will be gained if the order is shipped in that period. Some shipment periods are preferrable to others for a given order, because the profit realized by shipping an order can vary from period to period.

For each production facility, there is a finite amount of overtime that can be purchased in each time period. These limitations on production capacity are generally binding for the problem: There is approximately 75 percent of the required capacity needed to produce all orders available without purchasing any overtime. The overtime availability brings this figure up to approximately 88 percent. However, the problem has been constructed so that it may not be worth purchasing overtime to meet a specific order, unless the order can be shipped in the periods that realize the maximum possible profit. This condition represents, in effect, an additional constraint on production capacity.

The I-P-O balance constraint set of the test problem has approximately 115 rows and 270 variables. Solving LPs to optimality over this constraint system is currently being
accomplished with an average expenditure of .1 CPU seconds on the Apollo DN320 minicomputer. Empirical research on generating production schedules for this problem is currently in progress. Run time statistics can be found in Calamaro and Chapman (1985).
Figure 5-2: The $a_{ij}$ parameters
5.3 A Simple Data Structure

We now describe a simple data structure that is used repeatedly in this implementation. We clarify the nature of this structure by describing one of its applications in our pure production problem.

Consider any allowable product structure, such as the structure illustrated in Figure 5-4. (As we saw in Section 5.2, this is the product structure we are currently using for testing purposes.) The convention for indexing items shown in the figure (i.e., increasing indices moving left to right across levels, starting at the lowest level and working downstream) is the convention we assume is present in every product structure. The lines linking numbered boxes in the figure represent predecessor-successor relations, and this information is vital in our pursuit of an optimization procedure. In general, we need methods for representing the information contained in product structures in our computer data structures.

For example, we will need to know, at various times, the list of predecessors of a given item \( i \), if any, and the parameters \( a_{ji} \) for every item \( j \in P_i \). (Recall that \( P_i \), \( S_i \) are the sets of predecessors, successors of item \( i \) respectively.) We construct a simple list, or array, as a data structure that is a list of all predecessors for all items in the product structure. That is, we start the array by listing all the predecessors of item 1, followed immediately by all predecessors of item 2, etc. This array, generally known as the predecessor array, is illustrated in Figure 5-5 for the product structure found in Figure 5-4. Note that there are no entries for items with no predecessors. It is clear that a similar array with exactly the same number of entries, called the \( A_{ji} \) array, can be constructed giving the parameters \( a_{ji} \). Hence while a certain entry in the predecessor array gives the index \( j \) of a predecessor of some item \( i \), the corresponding entry in the \( A_{ji} \) array gives the parameter \( a_{ji} \).
Figure 5-4: A Test Product Structure

Figure 5-5: A Predecessor Array

Now, in order to access the predecessor data for any item, we construct a third array, known as the predecessor-pointer array. There is one component for each item in the product structure in this array, sequenced by increasing item indices, and the $i$th component (the component for item $i$) gives the index of the first component in the predecessor and $A_{ji}$ arrays that pertains to $i$. The predecessor-pointer array for the product structure of Figure 5-4 is shown in Figure 5-6. Notice that we can determine the number of predecessors of an item $i$ by subtracting the component value of $i$ from the component value of $i+1$ in the predecessor-pointer array. In
particular, if \( i \) has no predecessors, then \( i + 1 \) has the same component value as \( i \). For simplicity of presentation, we include an extra entry in the array to allow us to determine the number of predecessors of the last item (item 8) in the product structure. Of course, it is technically unnecessary to include pointers for items on the highest level, because our indexing convention implies that all items on this level are raw materials and thus have no predecessors.

![Figure 5-6: A Predecessor-Pointer Array](image)

It is clear that a similar simple construction of three arrays can be created to capture the successor relations in the product structure. We will also represent the relations of items to facilities in a similar manner. Namely, for each item we need to know what facilities are required for an item's production and the capacity requirement for producing one unit of the item on each of these facilities. This construction is also used to store data on when orders can possibly be shipped and the prices for shipping orders in allowable periods. We explicitly describe all data structures we use in our computer code in Section 5.5. Therein the pointer arrays in the constructions just described will always be identified with names ending in "ptr." In addition, the descriptions of these pointer arrays indicate the arrays of data to which the pointer arrays refer.
5.4 First and Last Constraints

As we mentioned in Section 5.1, one of the efficiencies to be realized in incorporating the setup information in the Lagrangian subproblem is a reduction in the number of constraints that must be considered in the optimization of the Lagrangian subproblem. We saw in Section 3.4 that, for pure production problems, all constraints if where $1 \leq t \leq L_i^*$ can be eliminated without affecting the subproblem's feasible region, and the resultant constraint system is totally dynamic. We now show that further elimination of constraints is possible with the incorporation of setup information in the Lagrangian subproblem, and the resultant system is still totally dynamic and defines the same feasible region.

Recall that the physical interpretation of the cumulative lead time $L_i^*$ is that $L_i^*$ is the number of periods required to produce item $i$ if all predecessors of $i$ must be manufactured; that is, no stocks of any predecessor of $i$ are available. Hence period $L_i^* + 1$ is the earliest period in which item $i$ can be available for use in the planning horizon, and all earlier period constraints for $i$ are unnecessary in the optimization.

Now consider any item $i$ with $P_i = \emptyset$. Given the current setup decision vector $x$, let $\tau_{xi}$ be the first period in which production is allowed for $i$ given the setup vector $x$. Hence period $\tau_{xi} + L_i \geq L_i + 1 = L_i^* + 1$ is the first period that can receive production of $i$. If $j \in S_i$, it follows that $j$ cannot be available earlier than period $\tau_{xi} + L_i + L_j$. If we recursively pursue this logic, moving upstream in the product structure, the parameter $L_{ixi}$, known as the first constraint of production item $i$, emerges:

If $P_i = \emptyset$ and $X_{it} = 0 \ \forall \ t \in \{1, \ldots, T-L_i\}$, then $L_{ixi} = T + 1$.

For all other items $i$ with $P_i = \emptyset$, $L_{ixi} = \arg \min_t \{t \in \{L_i + 1, \ldots, T\} : X_{i,t} = 1\}$. 

184
If \( P_i \neq \emptyset \) and \( \max_{j \in P_i} L_{ji}^f + L_i > T \), or if \( P_i = \emptyset \) and \( \max_{j \in P_i} L_{ji}^f + L_i \leq T \) and \( X_{it} = 0 \ \forall \ t \in \{ \max_{j \in P_i} L_{ji}^f, ..., T - L_i \} \), then \( L_{ix}^f = T + 1 \).

For all other items \( i \) with \( P_i \neq \emptyset \), \( L_{ix}^f = \arg \min \ t \{ t \in \{ \max_{j \in P_i} L_{ji}^f + L_i, ..., T \} : X_{it}L_i = 1 \} \).

\( L_{ix}^f \) is, therefore, the first period in which production of \( i \) can possibly become available in a pure production problem, given setup vector \( x \), if production of \( i \) can be received during the planning horizon. Otherwise, \( L_{ix}^f \) is set to \( T + 1 \).

It is clear, then, that the constraints \( it \) where \( 1 \leq t \leq L_{ix}^f - 1 \) can always be eliminated from the Lagrangian subproblem that incorporates the setup information, since no stock of item \( i \) becomes available until the period \( L_{ix}^f \). It is straightforward to show that the resultant constraint system is totally dynamic in light of the analysis of Chapter 3.

We can potentially eliminate constraints from the end of the planning horizon, as well as from the beginning. Consider any item \( i \) on Level 1 of the product structure and any \( j \in S_i \). (Item \( j \) is a finished good on Level 0.) Observe that any production of \( i \) received after period \( T - L_j \) cannot be used in the production of \( j \), because such \( js \) could not be finished within the planning horizon. Given our assumption that orders are possible for finished goods only, the pursuit of this line of reasoning results in the definition of the last constraint \( L_i^l \) of a production item:

If \( S_i = \emptyset \), then \( L_i^l = T \).

For all other items \( i \), \( L_i^l = \max_{j \in S_i} [ L_j^l - L_j ] \).

The parameters \( L_i^l \) are interpretable as the last period of the planning horizon in which production of \( i \) can be received and subsequently used to satisfy an order for a finished good within the planning horizon. Observe that the last constraints do not depend on \( x \). The
implication is that the constraints \( it \) where \( L_i^l + 1 \leq t \leq T \) can be eliminated from the Lagrangian subproblem as well without affecting the feasible region of the subproblem. Again, the resultant system remains totally dynamic.

We can combine the first and last constraint parameters, then, to eliminate constraints from both the beginning and end of the planning horizon. Since first and last constraints are defined on considerations that are independent of each other, it is possible that \( L_{ix} > L_i \) for an item \( i \). In such a case, it will do us no good to receive production after \( L_i \), and consequently we redefine the first constraint parameters so that \( L_{ix} = L_i + 1 \) if production of \( i \) cannot be received within the first \( L_i \) periods.

If \( P_i = \emptyset \) and \( X_{it} = 0 \) \( \forall t \in \{1, \ldots, L_i - L_i\} \), then \( L_{ix} = L_i + 1 \).

For all other items \( i \) with \( P_i = \emptyset \), \( L_{ix} = \arg \min_t \{t \in \{L_i + 1, \ldots, L_i\} : X_{t-L_i} = 1\} \).

If \( P_i \neq \emptyset \) and \( \max_{j \in P_i} L_{ij} + L_i > L_i \), or if \( P_i \neq \emptyset \) and \( \max_{j \in P_i} L_{ij} + L_i \leq L_i \) and \( X_{it} = 0 \) \( \forall t \in \{\max_{j \in P_i} L_{ij}, \ldots, L_i - L_i\} \), then \( L_{ix} = L_i + 1 \).

For all other items \( i \) with \( P_i \neq \emptyset \), \( L_{ix} = \arg \min_t \{t \in \{\max_{j \in P_i} L_{ij} + L_i, \ldots, L_i\} : X_{t-L_i} = 1\} \).
5.5 Data Structures

In this section we define the data structures that are used for solving LPs over the totally dynamic constraint sets that remain after employing the analysis of Chapter 3 and the preceding sections of this chapter. We present the computer code in a rather generic language that resembles many high-level computer languages (PL-I in particular). This code should be readily understood by anyone with an exposure to computer algorithms.

All data structures defined here are either simple variables, one-dimensional arrays, or two-dimensional arrays. The dimensions of the arrays are defined by the values of simple variables. For example, the length \( n \) of the array \( A \) is indicated in the definition of \( A \) by labelling the array as \( A(n) \).

Following the verbal definition of a data structure, we indicate whether the structure contains real, integer, or boolean values, and whether the values of the structure are input to the program (\( \rightarrow \)) or output from the program (\( \leftarrow \)).

**Basic Variables**

Item__# = The number of items in the product structure. (Integer, \( \rightarrow \)).

Period__# = The number of periods in the planning horizon. (Integer, \( \rightarrow \)).

Fac__# = The number of production facilities. (Integer, \( \rightarrow \)).

**Item-related Arrays**

Yield__#(Item__#) = The parameters \( y_i \). (Real, \( \rightarrow \)).
\text{Min\_lead(Item\_#)} = \text{The parameters } L_i. \text{ (Integer, } \rightarrow \text{).}

\text{First\_constraint(Item\_#)} = \text{The parameters } L_{ix}^f. \text{ } L_{ix}^f \text{ represents the first period that is capable of receiving production of } i \text{ given the current setup decisions. (Integer, } \leftarrow \text{).}

\text{Last\_constraint(Item\_#)} = \text{The parameters } L_i^l. \text{ } L_i^l \text{ represents the last period that may need to receive production of } i \text{ to meet demands for finished goods within the planning horizon. (Integer, } \leftarrow \text{).}

\textbf{Product Structure Arrays}

\text{Pred\_succ\_#} = \text{The number of predecessor-successor relations in the product structure.}
\text{ (Integer, } \rightarrow \text{).}

\text{Pred\_ptr(Item\_# + 1)} = \text{Pred\_ptr(I) indexes the beginning of the data on item I in relation to its immediate predecessors. This data is found in the arrays Pred and } A_{ji} \text{ (Integer, } \rightarrow \text{).}

\text{Pred(Pred\_succ\_#)} = \text{For all } J \text{ such that } \text{Pred\_ptr(I)} \leq J < \text{Pred\_ptr(I + 1)}, \text{Pred(J) represents the index of the } (J - \text{Pred\_ptr(I)} + 1)\text{th immediate predecessor of I. (Integer, } \rightarrow \text{).}

\text{A_{ji}(Pred\_succ\_#)} = \text{Similar to the Pred array except } A_{ji(J)}, \text{ where } J \text{ satisfies } \text{Pred\_ptr(I)} \leq J < \text{Pred\_ptr(I + 1)}, \text{ is the parameter } a_{\text{Pred}(J, l)}. \text{ That is, } A_{ji}(J) \text{ is the number of units of item Pred(J) required to produce one unit of item I. (Integer, } \rightarrow \text{).}
Succ_ptr(Item__# + 1), Succ(Pred_succ__#), A_ij(Pred_succ__#) = These three arrays correspond in function to Pred_ptr, Pred, and A_ji respectively but represent the data on an item's relation to its immediate successors instead of its immediate predecessors. (Integer,→).

Item-Facility Relations

Item_fac__# = The number of item-facility relations. (Integer,→).

Fac_price(Fac__#, Period__#) = The vector γ of Lagrangian prices on the facility capacity constraints. (Real,→).

Cap_util(Fac__#, Period__#) = The utilization of facility capacity by the solution generated by the totally dynamic algorithm for each facility in each time period. That is, Cap_util represents the vector Bp where p is the production vector of the solution. (Real,←→).

Fac_util_ptr(Item__# + 1) = Fac_util_ptr(I) indexes the beginning of the data on the facilities required for the production of item I. This data is found in the arrays Facility and Unit_use. (Integer,→).

Facility(Item_fac__#) = For all J such that Fac_util_ptr(I) ≤ J < Fac_util_ptr(I + 1), Facility(J) is the index of the (J - Fac_util_ptr(I) + 1)th facility required for the production of item I. (Integer,→).

Unit_use(Item_fac__#) = Similar to the Facility array except Unit_use(J), where J satisfies Fac_util_ptr(I) ≤ J < Fac_util_ptr(I + 1), is the parameter bI,Facility,J. That is,
Unit\_use(J) is the requirement of the capacity of facility Facility(J) needed to produce one unit of item I. (Real,\rightarrow).
Orders

\( \text{Fin\_goods\_#} = \) The number of finished goods items. (Integer,\( \rightarrow \)).

\( Z(\text{Fin\_goods\_#}) = Z(I) \) is the number of orders for finished good I. (Integer,\( \rightarrow \)).

\( \text{Order\_#} = \) The total number of orders over all items. (Integer,\( \rightarrow \)).

\( \text{Period/order\_#(Order\_#)} = \) The number of possible shipment periods (not including the "no shipment" period) for a given order. (Integer,\( \rightarrow \)).

\( \text{Order\_period\_#} = \sum_{i=1,...,\text{Order\_#}} \text{Period/order\_#(i)}. \) (Integer,\( \rightarrow \)).

\( \text{Order\_period\_ptr(Order\_# + 1)} = \text{Order\_period\_ptr(I)} \) indexes the beginning of the data on the order variables for order I. This data includes all periods in which the order can be shipped and the cost of shipping in each of the possible periods. This data is found in the arrays \( \text{Order\_period} \) and \( \text{Order\_cost} \) respectively. Please note that data on the option of not shipping an order is not stored in the \( \text{Order\_period} \) and \( \text{Order\_cost} \) arrays. We assume that the "cost" of not shipping an order is zero. The entries in the array \( \text{Order\_cost} \) are all negative. (Integer,\( \rightarrow \)).

\( \text{Order\_period(Order\_period\_#)} = \) For all \( J \) such that \( \text{Order\_period\_ptr(I)} \leq J < \text{Order\_period\_ptr(I + 1)} \), \( \text{Order\_period}(J) \) is the period to ship order I represented by the \((J - \text{Order\_period\_ptr(I)} + 1)\)th order variable for order I. (Integer,\( \rightarrow \)).
Order_cost(Order_period__) = Similar to the Order_period array except
Order_cost(J), where J satisfies Order_period_ptr(I) ≤ J < Order_period_ptr(I + 1), is
the cost incurred by shipping order I in period Order_period(J). (Real,→).

Order_demand(Order__) = Order_demand(I) is the demand for the Ith order. (Real,→).

Production-Inventory

Prod_on(Item__,Period__) = The variable Prod_on(I,T) = '1'B if the setup variable
X_{IT} = 1, otherwise Prod_on(I,T) = '0'B. (Boolean,→).

Inv_cost(Item__) = Inv_cost(I) is the cost of holding one unit of item I in inventory from
one period to the next. (Real,→).

Demand(Item__,Period__ + 1) = Demand(I,T) is the demand for item I in period T
created by the optimal assignment of orders to delivery periods. Note that the demand (I,
Period__ + 1) for finished good I is the cumulative demand of all orders pushed out of the
planning horizon in the optimal solution. (Real,←).

Dual Solution Variables

Dual_IP(Item__,Period__ + 1) = Dual(I,T) is the dual price implied by the optimal
primal basis on the inventory-production-orders balance constraint set (5-2) for item I in
period T. (Real,←).
Primal Solution Variables

Prod_crit(Item_, Period_) ⃝ The variable Prod_crit(I,T) = '1'B if the critical variable in the optimal primal basis for the I-P-O balance constraint of item I in period T is a production variable. Otherwise Prod_crit(I,T) = '0'B. (Boolean, ↔).

Deliver(Order_) ⃝ Deliver(I) indicates the period in which the Ith order is to be shipped. (Integer, ↔).

Cost_coeff ⃝ Cost_coeff represents the portion of the optimal objective function value derived from the optimal inventory and orders vector (hv + hw). Hence Cost_coeff holds the optimal objective function value minus the contribution to this value from the optimal production vector. If the solution to the totally dynamic problem is to be used to create a new variable in a Dantzig-Wolfe master problem, then Cost_coeff would form the objective function coefficient. (See formulation (d-4:x) of Section 2.4.) (Real, ↔).

Prod_cost ⃝ The contribution to the optimal objective function value from the optimal production vector ((yB)p). Prod_cost plus Cost_coeff equals the optimal objective function value. (Real, ↔).
5.6 Computer Algorithms

In Section 5.6 we present the computer code for solving LPs over the totally dynamic subproblem of our pure production problem with orders. We will break the code up into major categories and document extensively the purpose and function of each section of code. (We will not document the calculation of the first and last constraint parameters quite so extensively, since we already devoted Section 5.4 to this topic.)

Calculation of the last constraints \( L_i \).

Logic: The \( L_i \)'s are calculated from the recursive definition of Section 5.4.

Code:

/* Initialize \( L_i \)'s to the final period of the planning horizon. */

Do I = 1 to Item__#;

    Last__constraint(I) = Period__#;

end:

/* Set the \( L_i \)'s */

Do I = 1 to Item__#;

    /* Last constraint of item I is already determined Update the last constraint for all predecessors of item I. */

    Do J = Pred__ptr(I) to (Pred__ptr(I + 1) - 1);

    If Last__constraint(Pred(J)) = Period__#

        then Last__constraint(Pred(J)) = Last__constraint(I) - Min__lead(I);
else Last__constraint(Pred(J)) =

max [ Last__constraint(Pred(J)), Last__constraint(I) - Min__lead(I)];

end; /* Do J = */

end; /* Do I = */
Calculation of the first constraints $L_{ix}$

Logic: Set the first constraint parameters according to the recursive definition of Section 5.4.

Code:

/* Initialize the $L_{ix}$'s */

Do I = 1 to Item_#;
    First_constraint(I) = Min_lead(I) + 1;
end;

/* Set the $L_{ix}$'s */

Do I = Item_# to 1 by -1;
    /* $L_{ix}$ is now set to $\max_{j \in P} L_{ix} + L_i$ if $P_i \neq \emptyset$, and $L_i + 1$ if $P_i = \emptyset$. Update $L_{ix}$ to its correct value by taking into consideration the eliminated production variables (i.e., the $P_{it}$s where $X_{it} = 0$) and the fact that $L_{ix}$ must be less than or equal to $L_i + 1$. */
    Do T = First_constraint(I) to Last_constraint(I) while (¬Prod_on(I,T - Min_lead(I)));
end;
    First_constraint(I) = min(T,Last_constraint(I) + 1);

/* Update first constraint of I's successors based on the correct value of $L_{ix}$. */

Do J = Succ_ptr(I) to (Succ_ptr(I + 1) - 1);
    First_constraint(Succ(J)) =
        max [ First_constraint(Succ(J)), First_constraint(I) + Min_lead(Succ(J)) ];
end;
end; /* Do I = */
Calculation of the Optimal Dual Variables on the Inventory-Production-Order Constraints

Logic: The optimal dual variables and critical columns for the optimal primal basis are calculated for the I-P-O balance constraint set (5-2). The calculations are based directly on the analysis of Section 3.6. The optimal dual variable and critical column is found for one constraint at a time, beginning with the constraints representing the items at the highest level of the product structure and working upstream. The dual variables are calculated by proceeding upstream in the product structure because dual prices for an item depend on prices on constraints of the item's immediate predecessors.

For the pure production problem at hand, the first constraint for each item has a production variable as its only critical variable. For the remaining constraints, either only an inventory variable is critical or the constraint has both inventory and production critical variables. In the latter case, it is decided which of the two will be critical in the optimal primal basis.

Local variables:

T = The current time period.

T_lag = The current time period minus the minimum lead time of the item.

Prev_sum = The contribution to the reduced cost of a production variable by the immediate predecessors of the item under consideration.

Prd_cost = The cost coefficient (yB)_{it} of P_{it} in the objective function when P_{it} is under consideration for the critical variable of I-P-O balance constraint i,t+L_i in the optimal basis. These cost coefficients clearly depend on the facility price vector y and are needed
only once throughout the totally dynamic algorithm. Specifically, they are needed when deciding whether or not \( P_{it} \) should be in the optimal primal basis when \( X_{it} = 1 \). Therefore, they are calculated as needed and stored temporarily in Prd__cost.

\[ \text{Dual__temp} = \text{Dual price on a constraint when an inventory variable is critical for the constraint in the primal basis.} \]

Code:

/* Determine optimal critical variables and dual prices on the I-P-O constraint set (5-2). */

Do I = Item__# to 1 by -1;

Do T = First__constraint(I) to Last__constraint(I);

    /* Determine critical variables and dual price on constraint \( it \). */
    T__lag = T - Min__lead(I);
    Dual__IP(I,T) = + \infty;

    If Prod__on(I,T__lag)
        then do: /* Constraint \( it \) has a critical production variable. */

    /* Calculate contribution to reduced cost of immediate predecessors. */
    Prev__sum = 0;
    Do J = Pred__ptr(I) to (Pred__ptr(I + 1) - 1);
        Prev__sum = Prev__sum + A__ji(J) \times
                    Dual__IP(Pred(J),T__lag);
    end;
/* Calculate cost coefficient \((\gamma B)_{i,t-L_t}\) of \(P_{i,t-L_t}\). */

Prd_cost = 0;

Do J = Fac_util_ptr(I) to (Fac_util_ptr(I + 1) - 1);

Prd_cost = Prd_cost + Fac_price(Facility(J), T_lag) x
Unit_use(J);
end;

/* Calculate dual price with production variable being critical. */

Dual_IP(I,T) = (Prod_cost + Prev_sum) / Yield(I);

Prod_crit(I,T) = '1'B;
end; /* then do */

If T > First_constraint(I)
then do:

/* Constraint has a critical inventory variable. Calculate dual price
with inventory variable being critical. */

Dual_temp = Inv_cost(I) + Dual_IP(I,T-1);
If Dual_temp < Dual_IP(I,T)
then do; /* Inventory variable should be in optimal basis. */

Dual_IP(I,T) = Dual_temp;

Prod_crit(I,T) = '0'B;
end;
end; /* then do */

end: /* Do T = */
end; /* Do I = */
Calculation of the Critical Variables for Each Order Constraint and the Demand Per Period for Finished Products

Logic: For each order constraint \( ij, j \in Z(i) \) of constraint set (5-3), the variable \( W_{ijt} \) that should be critical in the optimal primal basis is determined. For each constraint \( ij \) of (5-3), the dual price on the order constraint \( ij \) that gives a zero reduced cost is determined for each critical variable \( W_{ijt} \) of the constraint. The \( W_{ijt} \) with the lowest of these prices is the desired critical variable, since it creates nonnegative reduced costs on all of the \( W_{ijt} \) variables that are critical for constraint \( ij \). (See Section 3.6.)

Once the critical variable \( W_{ijt} \) for order constraint \( ij \) is determined, there is a demand \( d_{ij} \) created for finished good \( i \) in period \( t \). These demands are saved so that the values of the critical production and inventory variables may be calculated later.

Local Variables:

\[ M = \text{Index of order under consideration.} \]

\[ I, J = J\text{th order of item } I \text{ is under consideration.} \]

\[ K = \text{Index of the first } W_{ijt} \text{ variable such that } t \geq L_{ix} / \]

\[ N = \text{Index of the order variable } W_{ijt} \text{ under consideration.} \]

\[ \text{Dual__lowest = We calculate the dual price that yields a zero reduced cost for each critical order variable of an order constraint. As we calculate these prices sequentially, Dual__lowest contains the minimal price found so far.} \]
Ship_order = The shipment period implied by the order variable that produces Dual_lowest.

Ship_cost = The shipment cost implied by the order variable that produces Dual_lowest.

Dual_temp = Dual price on the order constraint that yields a zero reduced cost for the order variable under consideration.

Code:

M = 1;
Do I = 1 to Fin_goods__#:
  Do J = 1 to Z(I);
    /* Determine critical variable for order j of item i. */

    /* Order variables $W_{ij}$ where $t < L_{ij}$ are eliminated from the problem. Set K to reference the first order variable that has not been eliminated. */
    Do K = Order_period__ptr(M) to (Order_period__ptr(M + 1) - 1) while (Order_period(K) < First_constrant(I));
    end;

    /* Find critical variable from remaining variables. */
    Dual_lowest = $+\infty$;

    Do N = K to (Order_period__ptr(M + 1) - 1);
/* Calculate dual price on order constraint \( ij \) when the order variable \( W_{ij} \) referenced by \( N \) is critical. */

\[
\text{Dual\_temp} = \text{Order\_cost}(N) + \text{Dual\_IP}(I, \text{Order\_period}(N)) \times \text{Order\_demand}(M);
\]

If \( \text{Dual\_temp} < \text{Dual\_lowest} \)

then do:

/* Current order variable is the best candidate found so far for the critical variable of constraint \( ij \). */

\[
\text{Dual\_Lowest} = \text{Dual\_temp};
\]

\[
\text{Ship\_order} = \text{Order\_period}(N);
\]

\[
\text{Ship\_cost} = \text{Order\_cost}(N);
\]

end;

end; /* Do \( N = \) */

If \( \text{Dual\_lowest} > 0 \)

/* For order \( j \) of item \( i \), it is optimal to push the order out of the planning horizon. */

then do:

\[
\text{Ship\_order} = \text{Period\_#} + 1;
\]

\[
\text{Ship\_cost} = 0;
\]

end:

/* Record shipment period for order \( j \) of item \( i \). */

\[
\text{Deliver}(M) = \text{Ship\_order};
\]

/* Record demand created by the choice of critical variables. */
Demand(I,Ship__order) = Demand(I,Ship__order) + Order__demand(M);

/* Calculate the contribution to the cost of the optimal solution implied by the
critical order variable for order j of item i. */
Coeff__cost = Coeff__cost + Ship__cost;

/* Reference next order. */
M = M + 1;

end; /* Do J = */
end; /* Do I = */
Calculation of the Capacity Utilization Vector

Logic: Demands for each finished good in each period are now known. Moreover, the periods in which production will take place are known for all items in the product structure. Herein the production and inventory quantities for each production period are calculated, and the facility capacity utilization vector $B_p$, implied by the optimal production vector $p$, is calculated as output from the totally dynamic algorithm. The contribution to the optimal objective function value $(\gamma B)p$ made by the optimal production vector $p$ is calculated, as well as the contribution $(hv)$ of the optimal inventory vector $v$.

Observe that the optimal solution vector $(p,v,w)$ is not stored by the program. Whereas the option of storing this vector is always open, it is sufficient to store the arrays Prod_crit and Deliver. The former array tells which production variables are critical, and the latter indicates when each order should be shipped. From this information, the optimal vector $(p,v,w)$ can be quickly created as need by simple backward substitution.

Local Variables:

$T_{\text{first}}$ = The time period under consideration for item $i$ that receives production of $i$ initiated $L$, periods ago.

$T_{\text{last}}$ = The next period following period $T_{\text{first}}$ that receives production of $i$. Hence the demands for $i$ in periods $T_{\text{first}}$ through $(T_{\text{last}} - 1)$ are all met by the production of $i$ received in period $T_{\text{first}}$.

Cum_demand = The cumulative demand for item $i$ in periods $T_{\text{first}}$ through $(T_{\text{last}} - 1)$.  

205
T\_lag = T\_first - Min\_lead(i).

Cap\_use = The capacity utilization of a given facility implied by Cum\_demand.

Code:

Do I = 1 to Item\_#;
    T\_first = First\_constraint(I);
    Do While (T\_first ≤ Last\_constraint(I));

    /* Establish the periods and the cumulative demand covered by the production of I
       received in period T\_first. */
    Cum\_demand = Demand(I,T\_first);
    Do T\_last = (T\_first + 1) to Last\_constraint(I) while ¬(Prod\_crit(I,T\_last));
        Cum\_demand = Cum\_demand + Demand(I,T\_last);
    end;

    /* Calculate the contribution of the inventory held between the production points
       T\_first and T\_last to the solution cost. */
    Do T = (T\_first + 1) to (T\_last - 1);
        Cost\_coeff = Cost\_coeff + Inv\_cost(I) x Demand(I,T) x (T - T\_first);
    end;

T\_lag = T\_first - Min\_lead(I);
/* Calculate the contribution of the cumulative demand covered by the production of I received in period T_first to the facility capacity utilization vector and the solution cost. */

    Do J = Fac__util__ptr(I) to (Fac__util__ptr(I + 1) - 1);

    Cap__use = Cum__demand x Unit__use(J) / Yield(I);

    Cap__util(Facility(J), T__lag) = Cap__util(Facility(J), T__lag) + Cap__use;

    Prod__cost = Prod__cost + Fac__price(Facility(J), T__lag) x Cap__use;

    end;

/* Calculate the dependent demand for I's immediate predecessors created by the cumulative demand covered by the production of I received in period T_first. */

    Do J = Pred__ptr(I) to (Pred__ptr(I + 1) - 1);

    Demand(Pred(J), T__lag) = Demand(Pred(J), T__lag) + A_ji(J) x Cum__demand / Yield(I);

    end;

/* Set T_first for the next period that receives production of I. */

    T_first = T_last;

end; /* Do While */

end; /* Do I = */
Chapter 6 Extensions of the Research

In this closing chapter we broadly outline some ideas on possible extensions of the research in this dissertation. In particular, we discuss the areas that we feel may prove the most fruitful as future research topics, based on our experiences with this research area. Section 6.1 deals with issues of reformulating models, while Section 6.2 summarizes other possible research areas.

6.1 Model Reformulations

Whereas we feel the two models considered so far ((BMRP) and (EMRP)) are a significant step in the right direction, we also believe these models are not the final word in implementable models. For example, an examination of our models reveals that an item cannot have any positive production in any period without incurring a setup. This precludes the possibility of starting a production run of an item on a machine in one period and continuing the run through subsequent time periods. For example, it is not even possible to start a run late in period $t$ and end it early in period $t + 1$ without being charged for a setup in both periods. In fact, there is no concept of "early in a period" or "late in a period" in the optimization models because of the way the models have discretized time. That is, it is as if a week is discretized to a point and all production within a week occurs at this point. This concept is illustrated in Figure 6.1.

Thus when we say an item $i$ has a minimum lead time of one week ($L_i = 1$), we are certain, within our existing models, that production initiated in week $t$ is available in week $t + 1$. The models' assumption is that production of $i$ occurs "instantaneously" at production point $t$, "waits" one week, and is then available for use at "point" $t + 1$. 
In reality, we are producing items sequentially on machines or, more generally, at production facilities. That is, production jobs are run one after the other. Whereas it is probably necessary to discretize time when formulating production scheduling problems as mathematical programs, what we would like as the end result of our efforts is a Gantt chart (refer to Figure 6.2) for each machine indicating start times and run durations for all jobs scheduled on each machine over the duration of the planning horizon. These start times and run durations occur continuously throughout time, not at discrete moments. And, of course, we would like these schedules to be both low in cost and feasible. Clearly the optimization models fall short of producing such schedules. They do, however, have the potential for pointing us in the right direction by laying down a foundation for detailed scheduling. We will now outline some thoughts on this issue as an indication of possible future research areas.

There are potential difficulties in moving from the output of our current optimization models to detailed machine schedules. For example, consider again item \( i \) produced in period \( t \) that becomes available for use as a component in item \( j \) in period \( t + 1 \), as Figure 6.3 illustrates. Recall the assumption that \( L_j = 1 \). If the minimum lead time of item \( i \) is really one week, and we assume \( i \) is not available for shipment to other work stations until the batch is complete, then, in reality (verses the assumption of the current model), \( i \) will not be available for the production of \( j \) until one week after the production run of \( i \) is terminated. (See Figure 6.4.)
Now assume $i$ is produced on machine $m_i$ while $j$ is produced on $m_j$. If we use the optimization output to derive machine schedules, we must avoid situations like the one illustrated in Figure 6.5.
This is a simple example of the potential hazards one must account for while trying to derive detailed machine schedules from the output of the optimization models. These problems arise as a result of moving from discretized time of the current models, where all production occurs instantaneously, to the realities of continuous time, where jobs must be scheduled sequentially. We feel that formulating the mathematical programming models with the understanding that
the output of these models will serve as input to machine scheduling algorithms is an interesting topic. In particular, one must address finding a workable interface between the output of the discretized mathematical programming models and the detailed scheduling algorithms.

We have mentioned that production runs cannot "carry over" from one period to the next in our optimization models. One can alleviate this situation by introducing a new set of 0-1 variables $V_{it}$, known as "carry over" variables. We add the constraints

$$V_{it} - V_{i,t-1} \leq X_{it} \quad \forall i, t$$

to our models with the stipulation that $V_{i0} = 0 \quad \forall i$. We also let $V_{it}$ replace $X_{it}$ in the fixed charge constraints. The implication of this model enhancement is that we are now "charged" for setup utilization if and only if a production run starts anew, and this is modelled by setting $X_{it} = 1$.

Recalling that production facilities potentially produce several items, we can additionally require that at most one of the items produced on the facility in a given period can "carry over" to the next period. If $F_k$ is the set of items produced on facility $k$, this is accomplished by adding the constraints

$$\sum_{i \in F_k} V_{it} \leq 1 \quad \forall k, t.$$ 

These formulations with "carry over" 0-1 variables are interesting and are worth investigating as a modelling option. However, they involve a substantial increase in the number of binary decision variables in the model. In the models (BMRP) and (EMRP) used throughout this work, there is a binary decision variable for each item in each time period. Thus the number of decision variables present for reasonably large problems implies these problems are probably
beyond our ability to solve to optimality. Perhaps these problems even tax our ability to generate good feasible solutions. (Recall that Benders' decomposition requires the solution of a pure integer program to determine the values of the setup variables.)

The prospect, then, of increasing the number of 0-1 variables substantially has its drawbacks. One direction of possible future research is to move in the opposite direction: that is, reduce the size of our models judiciously through aggregation so that the number of decision variables is decreased while the relevancy of the models is retained. It may also be possible to carry out these reformulations so that some of the aforementioned shortcomings of the models are accounted for. Namely, one can attempt to formulate models that are cognizant of the fact that their output must serve as input to machine scheduling routines. We now discuss one potential way of reformulating our models with these goals in mind.

There is a distinct possibility that it is unnecessary to have a setup decision variable in each time period for many production items. Consider, for example, items that exist at a relatively high level in the product structure (i.e., close to raw materials). If there is relatively little value added in such items, the implication is that the items are cheap to stock. (An important principle of operations management is to "stock before the high value-added step" in the production process.) In addition, consider items produced on very expensive, capital-intensive machines. In this case the cost incurred in lost production due to setups is substantial. Therefore, intelligent production runs on such machines will tend to be relatively long. So, on the one hand, we have items in multistage systems that have little value added and/or that are produced on expensive machines. Consider, on the other hand, simple single-stage systems with constant demand. The items we are describing in the former system are analogous to the items of the latter system with high economic order quantities (EOQs). Clearly, for such items of the multistage systems, the fact that production runs may be somewhat long implies they may cross period boundaries. Also, these runs may cover the demand for the item for a number of time periods. The question is: why necessarily create a setup variable in each time period for these items in our production scheduling models? The multiple setup variables prevent the
model from allowing longer, perhaps multi-period, runs. Furthermore, the numerous binary variables for these items add a great deal of complexity to the optimization without returning any substantial benefit.

In an attempt to evolve our thinking about lot sizing, we consider models in which each potential production run will cover item demands for a contiguous block of time periods. As we are about to see, the length of this block can be determined \textit{a priori} by the characteristics of the production item and the production environment in question. Also, the potential production run that is designed to cover the demand in this block of periods can naturally, if the run becomes an actuality, cross time period boundaries without incurring additional setups. Moreover, it is possible to model such situations by reducing the number of integer variables in relation to models where positive production in any period implies a setup charge.

MRP must deal with similar lot-sizing issues, because MRP requires lot sizing rules as input. (As mentioned in Chapter 2, research on the effect of lot-sizing rules on MRP systems includes Berry (1972), New (1974), Biggs, Hahn, and Pinto (1980), and Collier (1980).) These rules are applied to every item in the product structure during parts explosion. The most popular of these rules include (see Berry (1972) and Orlicky (1975) for detailed discussions):

1) Lot-for-lot: Always produce exactly what is needed.
2) EOQ: Always produce the maximum of the item's EOQ and its immediate requirements.
3) Production order quantity (POQ): always produce enough to cover requirements of the next POQ time interval.
4) Part-period balancing (PPB): Produce enough to cover demands for a variable number of future periods. The number of periods the production will cover is determined by finding the maximum number of weeks for which the cumulative inventory holding cost is less than the cost of a setup.
5) Optimization: Find the (locally) optimal lot size by an optimization algorithm, such as the Wagner-Whitin algorithm.
For example, consider the situation in Figure 6.6 in which the requirements are met by lots derived by using PPB. (In this example lead times are assumed to be zero.) Note how both the length of the time frame covered by a production run and the lot sizes vary according to the requirements pattern.
<table>
<thead>
<tr>
<th>Week Number</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements</td>
<td>10</td>
<td>10</td>
<td>15</td>
<td>20</td>
<td>70</td>
<td>40</td>
<td>40</td>
<td>270</td>
<td>230</td>
<td>40</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>Order Quantity</td>
<td>55</td>
<td>110</td>
<td>40</td>
<td>270</td>
<td>270</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Beginning Invnt.</td>
<td>55</td>
<td>45</td>
<td>35</td>
<td>20</td>
<td>110</td>
<td>40</td>
<td>40</td>
<td>270</td>
<td>270</td>
<td>40</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Ending Invnt.</td>
<td>45</td>
<td>35</td>
<td>20</td>
<td>0</td>
<td>40</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>40</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Setup cost: $300  
Per-unit inventory holding cost: $2

Figure 6.6: Part Period Balancing Example
Assume we are able to obtain a rough, but reasonable, estimate of the requirements of an item throughout the planning horizon. (We have some suggestions for obtaining such estimates, but we don't want to confuse the general discussion by presenting them in this work.) The point in obtaining this estimate is to develop an idea of major requirements fluctuations. We then apply, say, PPB to this estimate. What we want from this exercise is not lot sizes per se, but contiguous blocks of periods whose aggregate requirements are to be covered by the various production runs. In the example of Figure 6.4, these blocks are 1-4, 5-6, 7, 8, 9-11, and 12.

A possible scenario for the reformulation of our models is the following: The contiguous period-aggregated blocks, called requirement blocks, replace the single periods of the current models. The requirement blocks are matched one-to-one with run blocks. The run blocks are also contiguous, period-aggregated blocks. The idea is that the aggregate demand for an item over the time frame covered by each requirement block is met by a production run during its corresponding run block, inventory, or a combination of the two.

Within each run block, there is the potential for exactly one production run. The optimization model determines whether this run actually occurs and, if so, how long it is. Because run blocks can encompass more than one period, the actual runs may naturally cross period boundaries. A setup is charged exactly once for every run block in which a production run actually occurs.

Consider Figure 6.7. The second level indicates the predetermined requirement blocks for item $i$. We assume $L_i = 1$. We must determine when production will take place to meet the requirements in each requirement block while simultaneously considering lead times. Anything produced after period 2 cannot be guaranteed to arrive in time to meet the requirements of requirement block 1. The block (labelled 1) encompassing periods 1 and 2 on Level 1 of the figure is the run block for requirement block 1 and indicates when the production run may take place to meet the requirements of requirement block 1. Level 1 gives a possible set of run blocks that match one-for-one (by numbers) with the requirement blocks of Level 2. Note
Figure 6.7: Level 1 represents potential production runs while Level 2 represents requirement blocks.

how lead time displacement is handled differently than in the (BMRP) and (EMRP) models. For example, no matter when the production covering requirement block 1 takes place in run block 1, it will meet the requirements of any period within requirement block 1. Such conventions alleviate the "overlap" problem illustrated in Figure 6.5 and hence lead to potentially smoother interfaces with machine scheduling algorithms.

The PPB exercise not only gives us an estimate of how to create requirement blocks, but it also gives us an estimate of the requirements covered by each run block; that is, how much needs to be produced in each run. Hence we can estimate the amount of time required for each potential run, which implies we can estimate the number of contiguous periods that should be allocated for the corresponding run block. We take note of the fact that in determining the run block periods, one would probably want to allow for the fact that a particular run may cover the requirements in more than one contiguous set of requirement blocks. For example, we may want run block 1 in the figure to cover the requirements of both requirement blocks 1 and 2, etc.
Whether a run actually covers multiple requirement blocks or not will be determined by the optimization algorithm as it resolves the tradeoffs between costs, capacities, and requirements.

Since we are dealing with multistage production systems, we must also consider the effects of internally generated demand. Hence we add the run and requirement block of item \( j \), a successor of \( i \), in Figure 6.8. There must be \( a_{ij} \) components of type \( i \) available at production initiation, for every unit of \( j \) being produced. It is natural, within our modeling framework, to have an arbitrary run block of successor item \( j \) receive its supply of \( is \) from a particular requirement block of \( i \). An interesting question is: how should we match run blocks of an item to the requirement blocks of its predecessors?

We need to make a series of definitions:

\[
\alpha(i,k) = \text{The first period covered by the } k\text{-th run block of item } i.
\]
\[
\beta(i,k) = \text{The last period covered by the } k\text{-th run block of item } i.
\]
\[
\rho(i,k) = \text{The first period covered by the } k\text{-th requirement block of item } i.
\]
\[
\nu(i,k) = \text{The last period covered by the } k\text{-th requirement block of item } i.
\]
\[
S_i = \text{The set of successor items of item } i.
\]
\[
\lambda(i,j,k) = \text{For any } j \in S_i, \text{ the requirement block of item } i \text{ which furnishes the } k\text{-th run block of item } j \text{ with its requirement for } is.
\]
\[
\Gamma(i,j,n) = \{ k : \lambda(i,j,k) = n \} \forall j \in S_i.
\]
\[
A_i = \text{The index set of the blocks of item } i.
\]

In the example of Figure 6.8, \( A_i = \{1,2,3,4\}, A_j = \{1,2,3,4\} \), while \( \alpha(i,2) = 4 \), \( \beta(i,2) = 4 \), \( \rho(j,1) = 7 \), and \( \nu(j,1) = 10 \).

Perhaps the most natural way to make the assignment \( \lambda(i,j,k) \) is:

\[
\lambda(i,j,k) = l^* \text{ where } l^* \text{ satisfies } l^* = \arg \max_{l \in A_i} \{ l : \rho(i,l) \leq \alpha(j,k) \}.
\]
Figure 6.8: Item \( j \) is a successor of item \( i \) in the product structure.

That is, we assign the \( k \)-th run block of item \( j \) to the latest requirement block of \( i \) that begins at least as early as the \( k \)-th run block of \( j \). Thus, for the example of Figure 6.8, \( M(i,j,1) = 1 \) while \( M(i,j,2) = M(i,j,3) = 3 \) and \( M(i,j,4) = 4 \). Also, \( \Gamma(i,j,1) = \{1\} \), \( \Gamma(i,j,2) = \emptyset \), \( \Gamma(i,j,3) = \{2,3\} \), and \( \Gamma(i,j,4) = \{4\} \). It is clear that the \( k \)-th run block of \( j \) cannot be assigned to a later requirement block \( l > l' \) of \( i \), since it is then possible that a production run of \( j \) in the \( k \)-th block can begin prior to the
beginning of block \( l \). On the other hand, there is no reason to make an assignment to an earlier block and carry inventory of item \( i \) for no good reason.

We can now set up the constraints of a mathematical program corresponding to this period-aggregated model. We begin with the inventory-production balance constraints of the \( k \)-th block of item \( i \).

\[
I_{i,k-1} + \sum_{t=\alpha(i,k)}^{\beta(i,k)} P_{it} - I_{ik} = \sum_{t=\gamma(i,k)}^{\nu(i,k)} d_{it} + \sum_{j \in S_i} \sum_{m \in \Gamma(j,i,k)} \sum_{t=\alpha(j,m)}^{\beta(j,m)} P_{jt}.
\]

The \( k \)-th requirement block has inventory \( I_{i,k-1} \) held over from the previous requirement block. It also receives the production from the \( k \)-th run block which is represented by the first summation. On the right hand side, the first summation represents the exogenous demand for \( i \) in requirement block \( k \). The triple summation represents the requirements for \( i \) generated by the run blocks of \( i \)'s successors which are assigned to the \( k \)-th requirement block of item \( i \). Any remaining inventory is assigned to \( I_{ik} \).

We require a setup variable for each run block, in contrast to each time period, in this model. The capacity constraints for machine \( m \) in period \( t \) are:

\[
\sum_{i=1}^{N} \sum_{k=1}^{A_i} (h_{im} P_{it} + s_{ikm} X_{ik}) - O_{mt} \leq CAP_{mt}
\]

while the fixed charge constraints are

\[
\sum_{t=\alpha(i,k)}^{\beta(i,k)} P_{it} - q_{ik} X_{ik} \leq 0 \quad \forall i,k : k \notin A_i
\]

or
\[ P_{it} - q_{ik}X_{ik} \leq 0 \ \forall i,k,t: k \in A_i, t \in \{a(i,k), a(i,k)+1, \ldots, \beta(i,k)\} \]

\( X_{ik} \) is the decision variable indicating if a run of item \( i \) will be made in run block \( k \). The parameter \( s_{ikm} \) will be positive only if:

1) \( i \) is produced on machine \( m \), and

2) \( t \in \{a(i,k),a(i,k)+1,\ldots,\beta(i,k)\} \).

That is, period \( t \) must fall within the \( k \)-th run block of item \( i \). If 1 and 2 are satisfied, then

\[ s_{ikm} = s_{im} / ((\beta(i,k) - a(i,k) + 1) \]

where \( s_{im} \) is the time required to set up item \( i \) on machine \( m \). Note that the production run of \( i \) in run block \( k \) is generally spread among the production variables

\[ \left[ P_{it} \right]_{t=a(i,k)}^{\beta(i,k)} \]

There appears to be no easy way to force production allocation to contiguous time periods. We have also “spread” the setup time of item \( i \) on machine \( m \) evenly among the periods of the run block. This decision is somewhat arbitrary; for example, we could alternately assign the full setup time to the first period of the run block.

This optimization model provides, if you will, “intelligent” input to a machine scheduling routine. The model performs considerable preliminary work in establishing the framework for finding machine schedules that satisfy requirements, account for lead times, and are capacity-feasible in general, multistage systems. It also moves in the direction of allowing for production runs that cross period boundaries without introducing additional 0-1 variables. In fact, it
reduces the number of binary variables. This model can be viewed as a time-aggregated model, relative to our original formulations (BMRP) and (EMRP). We notice that while inventory and setup variables are aggregated into run blocks, the production variables remain unaggregated. This is because items potentially share machines (facilities) with other items. Hence the "common denominator" of single periods is retained. The inventory-production balance constraints have been aggregated as well from time periods to blocks.

This rather high-level exposition is just one example of the type of modelling evolution we would like to see take place. In general, the issues that need to be addressed are:

1) New model formulations: what are their advantages and disadvantages?
2) The mathematical structure of these models in relation to the prototype models (BMRP) and (EMRP). What special structure is retained, gained, or lost by the new formulations relative to the prototype formulations?
3) The interface to detailed machine scheduling problems. How should the machine scheduling be accomplished given the mathematical programming inputs? What types of problems can develop and how should they be resolved?
6.2 Other Research Areas

It is apparent that any research advocating the use of mathematical programming decomposition as a solution methodology is unfulfilled unless a significant amount of effort is exerted in implementation and empirical research. As we discussed in Chapter 5, we are still in the early stages of testing the strategies outlined in this work, and much work remains in this area.

In the immediate future, we plan to devote attention to routines that will generate feasible schedules relatively frequently. Also, we will experiment with generating a "good" set of cuts for a Benders master problem by observing the effects of various setup decisions we generate, and, as a result, learning about the characteristics of the test problems at hand. Eventually, we will let the Benders master problem take over the task of making setup decisions. Also, the knowledge of the test problems gained may well provide a direction for model reformulations of the type discussed in the previous section. That is, our familiarity with the production environment may make it clear how to "intelligently" formulate requirement and run blocks. As we have seen, such model reformulations make it possible to model production runs that cross period boundaries without incurring a setup in every period.

We have mentioned various possible future research topics throughout this work. We now summarize these topics.

From a theoretical perspective, we know that the general capacitated multistage lot-sizing problem is NP-hard. However, to our knowledge, it has not been determined that the general uncapacitated multistage lot-sizing problem is NP-hard. This problem merits investigation.

In Sections 3.9 and 3.10 we investigated the issue of what to do when the special structure of inventory-production-balance-constraints is eliminated; because, in essence, the scheduling algorithms must cope with production and inventory at the beginning of the planning horizon that were not created by the scheduling algorithms. We feel this is an important issue that has
been largely ignored in the literature, and there is potentially much to be done beyond the results presented in this thesis.

Recall that we propose to generate Benders cuts by employing Lagrangian relaxation on the facility capacity constraints in the Benders subproblem. When employing Lagrangian relaxation in general, it is common to attempt to generate feasible solutions from the solutions to Lagrangian subproblems, usually via a heuristic. In the context of our problems, this means attempting to generate capacity-feasible solutions from the Lagrangian subproblems with priced-out facility capacity constraints. If the amount of overtime that can be purchased is bounded, then it is generally far from trivial to generate such capacity-feasible solutions in a multistage production environment. This follows directly from the "ripple" effect that results from changing the production runs of a particular item. That is, the production runs of an item cannot generally be changed in isolation: If production of item i is moved earlier in time, then the supplies of i's predecessors required for this production of i may arrive too late. Similarly, if production of i is moved later in time, then this production run may be completed too late to meet requirements of i created by production of i's successors. It would be interesting to investigate seriously the possibility of developing heuristics for generating capacity-feasible schedules from relaxations that account for these dependencies.

We have proposed several procedures of generating equi-cost feasible solutions to the dual problem (SPx,D) from feasible solutions to the dual problem (SPx-2,D). Generally, we found that more effort spent in generating cuts results in "stronger" cuts. It would be helpful to have some empirical results on the tradeoffs between the amount of time spent on generating these solutions and the quality of the Benders cuts that are generated. The quality of these cuts is most naturally measured by the computational effort required to solve Benders master problems resulting from the cuts. In addition, we are beginning to investigate how to create Benders cuts from a different pair of dual regions (that is, different from (SPx,D) and (SPx-2,D)). It appears that much of the theory of Chapter 4 will apply to this new pair of regions.

Moreover, it may be true that "strong" Benders cuts can be generated with greater ease with
this new pair of dual regions than is possible with \((SPx,D)\) and \((SPx-2,D)\). Finally, we would like develop a theoretical framework for generating, in the terminology of Chapter 4, "efficient" Benders cuts.
Bibliography


Berry, W.L. (1972), "Lot sizing procedures for requirements planning systems: a framework for analysis," Production and Inventory Management, 2nd Quarter, 19-34


Orlicky, J.A. (1973), "Net change material requirements planning," IBM Systems Journal, 12, 2-29


Veinott, A.F. Jr. (1968), "Extreme points of Leontief substitution systems," *Linear Algebra and Its Applications, 188-194*


