Mathematical Programming Models and Methods for Production Planning and Scheduling*

by

Jeremy F. Shapiro**

OR 191-89 January 1989

* Draft of a chapter to appear in The Handbook of Operations Research, forthcoming from North-Holland.

** This review was supported in part by a grant from the Leaders in Manufacturing Program at MIT.

MATHEMATICAL PROGRAMMING MODELS AND METHODS FOR PRODUCTION PLANNING AND SCHEDULING

by Jeremy F. Shapiro

January, 1989

Table of Contents

1. OVERVIEW

Mathematical programming is a rich formalism from which powerful optimization models for production planning and scheduling can be constructed and solved. Imbedded in easy-to-use decision support systems, the models can provide production managers with the means to more globally analyze their problems, thereby increasing net revenues or reducing costs.

We have chosen to distinguish between production planning problems, which are tactical in nature, and scheduling problems, which are operational in nature. Models for production planning tend to be aggregate, seek to describe large segments of the production environment, and typically consider planning horizons of one month to one year. By contrast, models for production scheduling tend to be detailed, seek to describe smaller segments of the production environment, and consider planning horizons of a few hours to several days or a few weeks.

Of course, the scope of the models and the analyses they can provide depends on the application. In some instances, the complexity and size of the application may be such that an effective micro-based model and optimizer can be developed. In other instances, the complexity and size may require optimization by a supercomputer. More experience with the practical use of mathematical programming models in solving a variety of production planning and scheduling problems is required before we will fully understand how to choose effective models.

At present, two compelling reasons are stimulating new interest in applications of mathematical programming to production planning and scheduling. First, the information revolution has progressed to a point that an increasing number of managers are actively seeking decision support systems to analyze their problems. Although accurate and complete data bases are a pre-requisite for better production management, managers have come to realize that new tools are needed to unravel the complex interactions and ripple effects that make production problems difficult _and_ important. Second, computer technologies that have facilitated the recent breakthroughs in data base construction and management can also be harnessed to the tasks of efficient model generation and optimization.

A major concern throughout this chapter is to present models and solution methods that either have already been applied to actual production planning and scheduling problems, or which show reasonable promise for practical application in the near future. In this regard, we will attempt to convey how the art as well as the science of modeling building and algorithm construction should be employed in specific problem solving situations. The formalism of mathematical programming is often only the point of departure for analyzing large scale problems. To it we add approximation, aggregation and heuristic methods designed to foster the computation of a "good" or "good enough" solution in an acceptable length of time.

A major objective when creating a mathematical programming model for production planning or scheduling is to develop effective approaches for coordinating or integrating a broad range of production activities. The result may be a large scale model that is difficult to assemble, optimize or interpret as a single, monolithic entity. Thus, an important focus of this chapter will be an exposition of mathematical programming decomposition methods that allow large scale models to be broken down into manageable sub-models that can be implicitly reassembled. These methods show considerable promise for time critical scheduling applications, especially when the methods have been adapted for and implemented on parallel computers. In any event, beyond their demonstrated and potential practical importance, decomposition methods are extremely useful as pedagogical devices for explaining model construction and analysis.

The mathematical programming models we will consider fall into several categories: linear programming, network optimization, mixed integer programming, nonlinear programming, dynamic programming, and stochastic programming. The form of these models will be implicitly reviewed as we develop specific production planning and scheduling applications throughout the chapter. The assumption is that the reader has some acquaintance with the field; for background, the reader is referred to Schrage (1986), Shapiro (1979a), and Williams (1985).

The plan of this chapter is as follows. In section 2, we discuss mixed integer programming, dynamic programming and traveling salesman models for lot-sizing and one machine scheduling problems. These relatively simple models represent building blocks that we employ in later sections to construct larger, more comprehensive models. Lagrange multiplier and decomposition methods are presented in section 3. These methods are used throughout the remainder of the chapter to analyze and optimize large-scale models.

The following three sections, sections 4, 5, and 6, are devoted to mathematical programming models for three distinct types of production planning and scheduling problems: process manufacturing, discrete parts manufacturing, and job-shop scheduling. In practice, of course, one cannot always classify production problems as fitting cleanly in one of these categories. At a paper mill, for example, the production of pulp and the operations of the paper machines are characterized by process manufacturing activities, but conversion of the paper to meet the specifications of specific orders (sizes, coatings, etc.) is characterized by job-shop scheduling activities related to specific orders.

Moreover, space and time constraints prevent us from considering production planning and scheduling environments that are not adequately described by these three main model types. In particular, we will not cover models for flexible manufacturing systems and refer the reader to Buzacott and Yao (1986); see also

Stecke (1983). Nor will we cover the growing body of literature on the application of graph and network optimization techniques to the production of printed circuit boards (PCB) and very large-scale integrated (VSLI) chips; see, for example, Feo and Hochbaum (1986) and Ball and Magazine (1988).

Section 7 contains a brief discussion of stochastic programming extensions to the deterministic models of earlier sections. In section 8, we discuss the important role that mathematical programming models can play in coordinating manufacturing with other company activities. The chapter concludes with a section devoted to future directions for mathematical programming applications to production planning and scheduling.

Our reference citations are intended to be representative rather than encyclopedic. This is a necessity rather than a convenience since the number of articles and books on the subject published over the past 30 years is enormous. Preference in most instances is given to recently published articles. Interested readers can easily develop their own reference list by working backward from these recent papers.

2. SIMPLE MODELS

We characterize simple models for production planning and scheduling as those with the property that they can be solved in at most a few seconds, even on a microcomputer, for the majority of practical applications. Moreover, the problems addressed by simple models are primitive or elementary ones that reflect only myopic planning concerns. Two examples that we discuss below are: a dynamic programming model for production planning of a single item; and, a traveling salesman problem formulation of one machine scheduling problems.

The class of simple models includes those that can be optimized by polynomially bounded list processing algorithm. The class also includes some models, such as those for the traveling salesman problem, from the class of combinatorial models called NP-Complete for which polynomially bounded algorithms do not, in all likelihood, exist (see Papadimitrou and Stieglitz (1982)). However, for the vast majority of practical applications, optimization of traveling salesman problems, knapsack problems and several other NP-Complete problems, can be carried out very quickly, at least to a good approximation. Of course, other NP-complete models, such as the general zero-one programming model, can often be relatively difficult to optimize, and we will not consider them in the category of simple models.

From another perspective, the simple models we discuss here will serve mainly as elements in large-scale model syntheses.

This is their primary importance. Conversely, a practitioner would be hard pressed to utilize these simple models on a stand alone basis to support production decision-making. This is because, for example, production and inventory decisions about an individual product are difficult to isolate from those for other products, or, decisions about scheduling an individual machine are difficult to isolate from those for other machines that can process some or all of the same products.

2.1 SINGLE ITEM DYNAMIC LOT-SIZE MODELS

A classic production problem is to determine an optimal balance between set-up costs and inventory carrying costs. On the one hand, a production manager prefers long production runs to spread out set-up costs. On the other hand, he/she would like to make frequent, smaller runs that match demand, thereby reducing inventory carrying costs to a minimum. The basic dynamic lot-size model (Wagner and Whitin (1958)) formally optimizes these conflicting decisions for individual production items.

Parameters:

f = set-up cost ($)

c_t = variable production cost ($/unit)

h = inventory carrying cost ($/unit)

r_t = demand in period t (units)

M_t = upper bound on production in period t (units)

Decision Variables:

y_t = inventory at the end of period t

x_t = production during period t

$$\delta_t = \begin{cases} 1 \text{ if production occurs during period t} \\ 0 \text{ otherwise} \end{cases}$$

<u>Dynamic Lot-size Model</u>

$$v = \min \sum_{t=1}^{T} \{f\ \delta t + c_t\ x_t + h\ y_t\} \qquad (2.1)$$

Subject to

$$y_t = y_{t-1} + x_t - r_t \qquad (2.2)$$

for t=1,..,T

$$x_t - M_t\ \delta_t \leq 0 \qquad (2.3)$$

$$y_0 \text{ given} \qquad (2.4)$$

$$y_t \geq 0,\ x_t \geq 0,\ \delta_t = 0 \text{ or } 1 \qquad (2.5)$$

This is a very simple MIP model. As stated, demand in each period must be satisfied by some combination of starting inventory and production since ending inventory y_t is constrained to be non-negative; i.e., no backlogging is required. The model could easily be extended to allow backlogging by substituting

$$y_t = y_t^+ - y_t^-$$

with $y_t^+ \geq 0$, $y_t^- \geq 0$. The unit holding cost h is then associated with y_t^+ and a unit backlogging penalty cost, say p, with y_t^-.

Another simple model enhancement that may be worthwhile is to constrain final inventory y_T to lie in some set. Otherwise, the optimization model will tend to run inventories to zero at the end of the planning horizon, even though production and sale of the item will continue for the foreseeable future. Alternatively, the term $-rq_T$ could be added to the objective function where $r > 0$ is a reward per unit (negative unit cost) for terminal inventories.

A more extensive enhancement is to consider part or all of demand r_t to be tied to specific orders with specific due dates. We consider the following example. Suppose demand consists entirely of orders of size w_k for $k=1,\ldots,K$, where each order has a target completion (shipping) date t_k. Suppose further that every order must be complete before it is shipped. Finally, suppose an order is allowed to be completed one or two periods after the due date t_k, with penalties p_{k1} and p_{k2}, respectively.

We extend the formulation (2.1) - (2.5) to include the situation just described. Let $K_t = \{ k \mid t_k = t\}$. Let β_{tk}, $\beta_{t,k+1}$, $\beta_{t,k+2}$ denote zero-one variables that take on values of one only if order k is completed in periods t_k, $t_k + 1$, $t_k + 2$, respectively. With this new notation and variables, we change the equations (2.2) to

$$y_t = y_{t-1} + x_t - \sum_{k\epsilon K_t} w_k \beta_{tk} - \sum_{k\epsilon K_{t-1}} w_k \beta_{t,k+1} - \sum_{k\epsilon K_{t-2}} w_k \beta_{t,k+2} \qquad (2.6)$$

In addition, for each order k, we add the multiple choice constraint

$$\beta_{tk} + \beta_{t,k+1} + \beta_{t,k+2} = 1 \tag{2.7}$$

Lastly, we add to the objective function the penalty term

$$\sum_{k=1}^{K} p_{k1}\,\beta_{t,k+1} + p_{k2}\,\beta_{t,k+2} \tag{2.8}$$

The single item dynamic lot-size model (2.1) - (2.5) can be re-formulated and optimized as a dynamic programming model. Let

$G^t(y)$ = minimal cost of meeting demands in periods $t, t + 1, \ldots, T$ when inventory at the beginning of period t is y

We let Y^t denote the set of beginning inventory states that we need to consider for period t. For example, we might let

$$Y^t = \{y : 0 \le y \le \sum_{s=t}^{T} r_s + \bar{y}_T\}$$

where $\bar{y}_T$ is an upper bound on inventory at the end of the planning horizon.

The G^t functions satisfy the following recursive relationship for all $y \varepsilon Y^t$

$$G^t(y) = \text{minimum } \{f\delta_t + c_t x_t + h(y + x_t - d_t) + G^{t+1}(y + x_t - d_t)\} \tag{2.9}$$

Subject to $x_t \ge \max\{0, d_t - y\}$

$x_t - M_t\,\delta_t \le 0$

$x_t \ge 0,\ \delta_t = 0$ or 1

where $G^{T+1}(y)$ given for all y

The recursion (2.9) is solved by backward iteration starting with $t = T$ and continuing until $G^1(y_0)$ has been computed. Since the state space y in each period is continuous, some finite discretization is in general required to perform these computations. The result, as we shall shortly see, is a representation of the model and optimization algorithm as the problem of finding a shortest route in a network.

First, we discuss a special property of the Dynamic Lot Size Model. Using an induction argument, one can show that the functions $G^t(\cdot)$ are concave. This property can be used to prove the following result due to Wagner and Whitin (1958); see also Wagner (1969).

DYNAMIC LOT-SIZE THEOREM: If initial inventory $y_0 = 0$, an optimal solution to the Dynamic Lot-Size Model can be characterized by the condition

$$x_t \, y_{t-1} = 0 \text{ for } t=1,\ldots,T$$

This result implies further that an optimal production strategy must satisfy $x_t > 0$ only when $y_{t-1} = 0$ in which case x_t equals one of the quantities

$$r_t,\ r_t + r_{t+1},\ldots,\ \sum_{s=t}^{T} r_s$$

Unfortunately, the condition that starting inventory

inventory equals zero (or $\sum_{t=1}^{s} r_t$ for any s) is not realistic in the sense that if we were considering a real-life inventory system, we would in all likelihood find y_0 to be some random amount at the start of a planning horizon. Adjusting to the initial transient caused by positive inventory may require several periods.

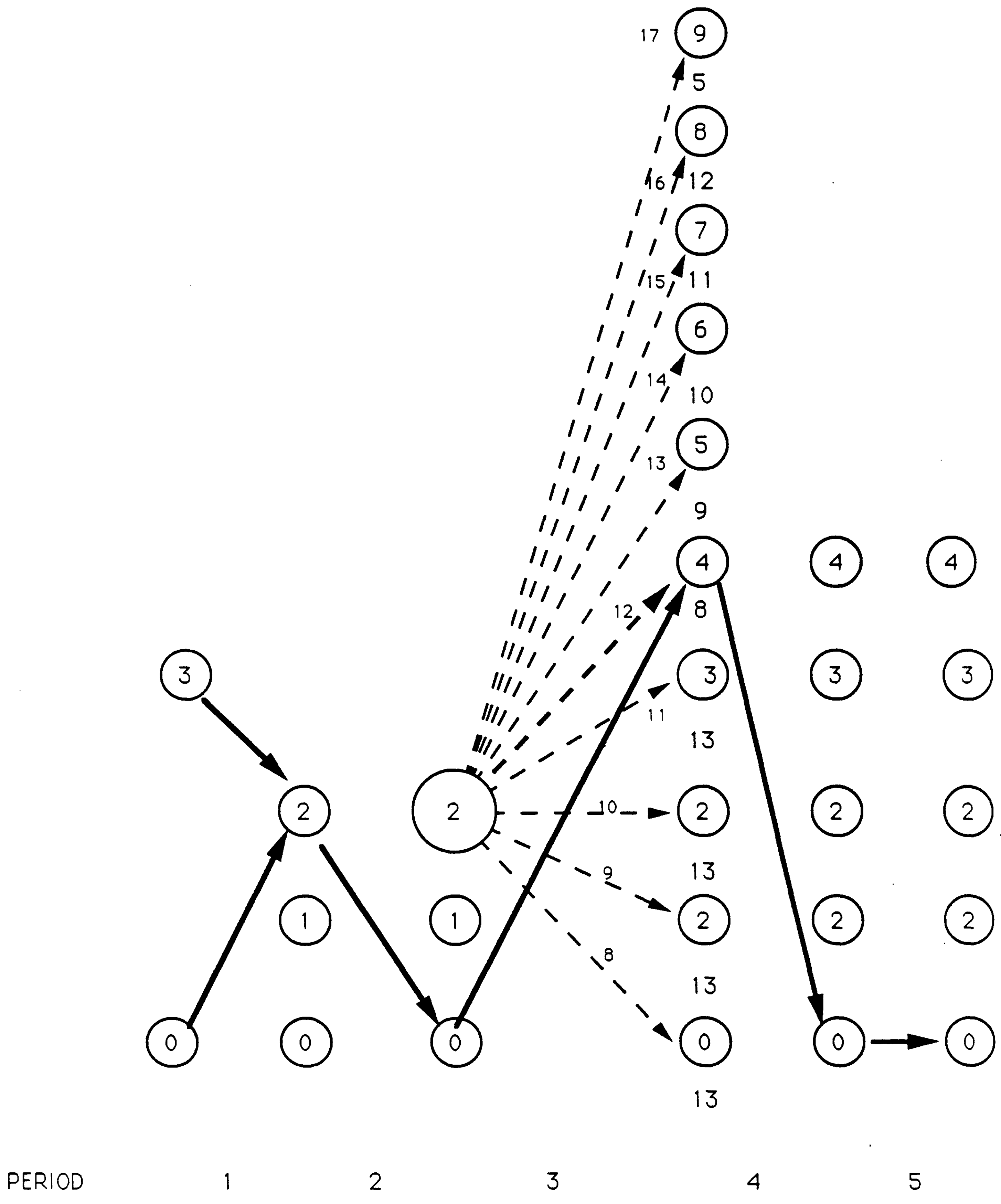
We return now to a consideration in more detail about how one computes optimal solutions with the recursion (2.6). Accordingly to the Dynamic Lot Size Theorem, when initial inventory is zero, we can solve the Dynamic Lot Size Model by a simple dynamic programming model that considers producing in lot amounts equal to the next k periods demand for appropriate values of k. Since the simplifying assumption may not hold, and also for the purposes of exposition, we ignore the consequences of the Theorem in illustrating the dynamic programming approach.

Consider a five period problem with demands $r_1=695$, $r_2=177$, $r_3=511$, $r_4=425$, $r_5=468$. Suppose that the set-up cost = \$800, and that the unit holding cost = \$1. We assume for simplicity that the variable production cost is constant over time and therefore can be ignored (the total variable cost of production is a fixed amount). Choosing the basic unit of inventory and demand to be 100 units and dollars to be \$100., we then have demands (rounding off to the nearest 100) $r_1=7$, $r_2=2$, $r_3=5$, $r_4=4$, $r_5=5$, a set-up cost of 8, and an inventory holding cost = 1. We consider two cases: starting inventory = 0 and starting inventory = 3.

Figure 2.1 depicts this problem as a shortest route problem in a network where the nodes correspond to starting inventory states in each period and the arcs correspond to feasible statetransitions. Not all nodes and arcs they need be considered are actually shown in the figure. The arc "lengths" are the immediate costs. Consider, for example, the choice of an optimal immediate decision at the start of period 3 when starting inventory equals 2 units. The numbers beneath the nodes at the end of period 3 are the $G_4(y)$ values previously computed by backward iteration. The decision choices are to set-up and make any number of units between 3 and 12, with associated transitions to ending inventory states ranging from 0 to 9. The length of the arc from node to node 3 is 11 corresponding to a set-up cost (8) plus an inventory holding cost (3) on three units of ending inventory. Comparing the 10 decision options, the optimal immediate decision to is to set-up and manufacture 7 units, thereby making the transition to 2 + 7 - 5 = 4 units of ending inventory.

The solid dark arcs represent shortest routes from the two possible initial inventory states. The implied optimal production strategy when starting inventory is 3 units, for example, is: $x_1 = 6$, $x_2 = 0$, $x_3 = 9$, $x_4 = 0$, $x_5 = 5$. Note that the optimal routes for both cases follow the pattern predicted by the Dynamic Lot Size Theorem; namely, produce only when inventory falls to zero.

The Dynamic Lot-Size Model has been studied and generalized by several authors. Zangwill (1966) extends the dynamic lot-size theorem to include the case of backlogging. Crowston, Wagner and



Shortest Route Network Representation
of Dynamic Lot-Size Model

Figure 2.1

Williams (1973) extend the model and the analysis to a multi-stage and multi-product assembly system in which each product may have many predecessors, but only one successor (multi-item assembly systems are discussed again in section 5.2). Over an infinite planning horizon, and under mathematical assumptions similar to those made for the Dynamic Lot Size Model and Theorem, they prove that the optimal lot size at each stage is an integer multiple of the lot size at the successor stage. Karmarkar, Kekre and Kekre (1987) generalize the model to allow long production runs for a given item that last more than one period. In such an event, only one set-up should be charged for production of that item.

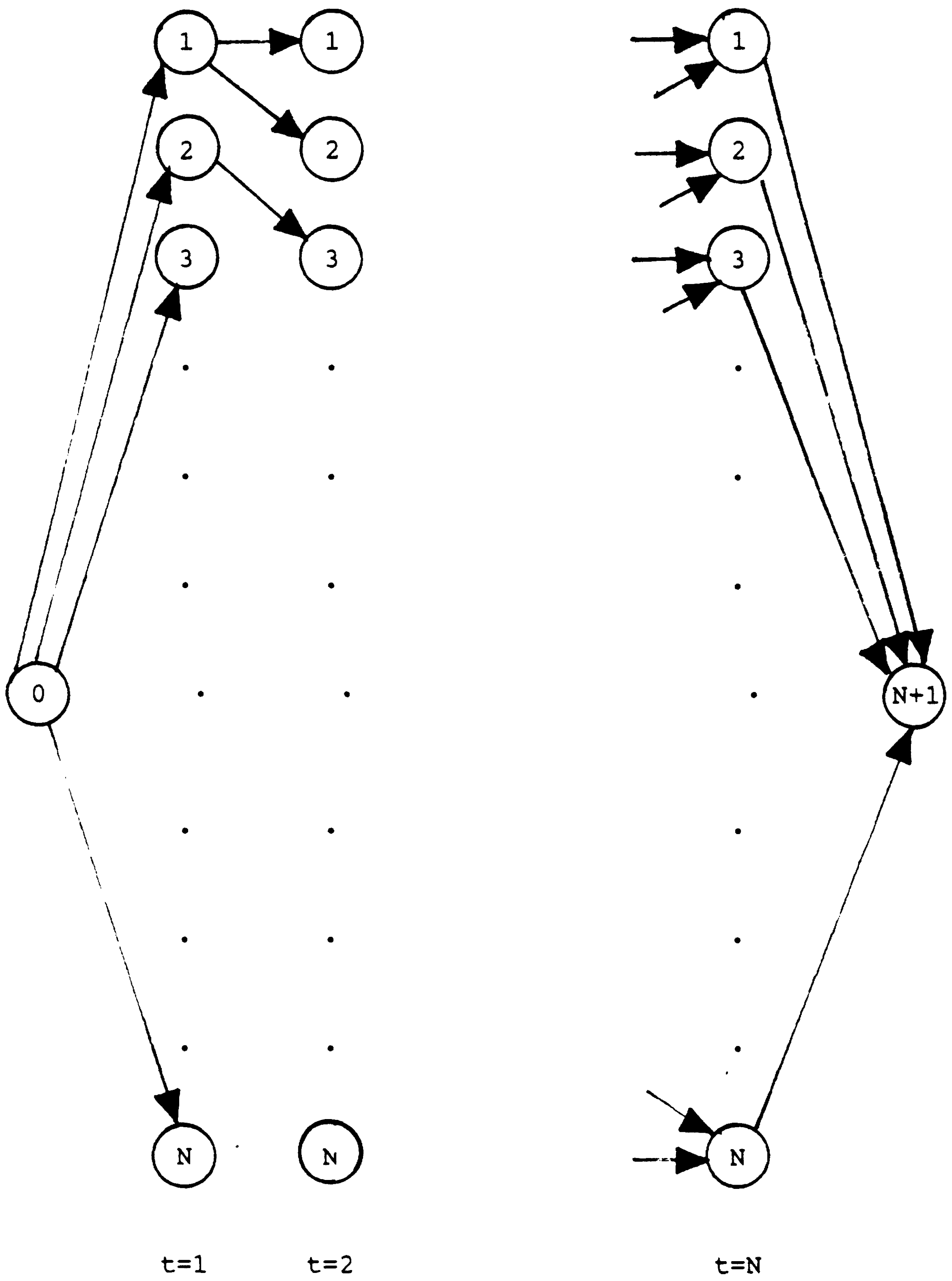
Love (1972) gives a dynamic programming approach and algorithm for the multi-stage serial manufacturing system in which each product may have exactly one predecessor and one successor. Crowston and Wagner (1973) give a similar but more general dynamic programming approach and algorithm for finite horizon, multi-stage assembly planning.

2.2 THE TIME DEPENDENT TRAVELING SALESMAN PROBLEM AND ONE MACHINE SCHEDULING

The Traveling Salesman Problem (TSP) is another classical and well studied combinatorial optimization problem. Simply stated, the TSP is concerned with finding a tour of a set of cities such that every city is visited exactly once and the total distance traveled is minimal. The TSP admits several distinct zero-one integer programming formulations (see Lawler et al (1985)). Its relevance here is to optimizing the schedule of several jobs processed sequentially on a single machine. In section 6, we discuss job shop scehduling problems that correspond roughly to multiple TSP models, one for each machine, with linking precedence constraints.

For our purposes, the formulation that we prefer is one consisting of a shortest route model with additional constraints requiring each node (city) to be visited exactly once. We present this formulation and then discuss how it can be adapted to one machine scheduling.

Consider a network with nodes 0, 1, 2,..., N, directed arcs (i, j) for all i and j = i, and associated arc lengths c_{ij}. Node 0 corresponds to the starting city. For technical purposes, we define a new node with the label N+1 which is located identically with node 0 but corresponds to terminating there. Thus, the arc costs $c_{i,N+1} = c_{i0}$ for all i. As shown in Figure 2.. we consider N replications of the nodes 1, 2,..., N, each replication indexed by t. The TSP can be expressed as the problem of sending one unit of flow from node 0 to node N+1 so that the distance traveled is minimized (this is a shortest route problem), but with the additional constraints that each node i must be visited exactly once.



Shortest Route Representation of TSP

Figure 2.2

Letting x_{ijt} denote zero-one decision variables, the model can be stated as

Traveling Salesman Problem

$$\min \sum_{j=1}^{N} c_{0j}\, x_{0j0} + \sum_{t=1}^{N-1} \sum_{i=1}^{N} \sum_{j=1}^{N} c_{ij}\, x_{ijt} + \sum_{i=1}^{N} c_{i,N+1}\, x_{i,N+1,N} \tag{2.10}$$

Subject to

$$\sum_{J=1}^{N} x_{0j0} = 1 \tag{2.11}$$

$$\left.\begin{array}{lll} -x_{0i0} + \sum_{j=1}^{N} x_{ij1} = 0 & & \text{for } i=1,\ldots,N \\ -\sum_{k=1}^{N} x_{ki,t-1} + \sum_{j=1}^{N} x_{ijt} = 0 & & \text{for } i=1,\ldots,N \\ & & \quad t=2,\ldots,N-1 \\ -\sum_{k=1}^{N} x_{ki,N-1} + x_{i,N+1,N} = 0 & & \text{for } i=1,\ldots,N \end{array}\right\} \tag{2.12}$$

$$\sum_{t=1}^{N} x_{i,N+1,N} = 1 \tag{2.13}$$

$$x0j0 + \sum_{t=1}^{N-1} \sum_{\substack{i=1 \\ i\neq j}}^{N} x_{ijt} = 1 \qquad \text{for } j=1,\ldots,N \tag{2.14}$$

$$x_{ijt} = 0 \text{ or } 1 \tag{2.15}$$

The constraints (2.11) - (2.13) describe the shortest route problem and the constraints (2.14) require each node 1,...,N to be visited exactly once. Note that the arc lengths c_{ij} do not depend on t. When they do depend on t, such as in the machine scheduling formulation about to be described, we say that we have a Time Dependent TSP.

Consider the problem of scheduling N+1 jobs on a single machine (Picard and Queyranne (1978)). For $i = 0, 1, 2, \ldots, N$, we associate with job i an integer processing time p_i, an integer due date d_i, an integer changeover time h_{ij}, and a tardiness cost $C_i(t)$ for completing the job by time t. We assume the job is currently (that is, at time t=0) set-up for job 0. Two examples of $C_i(t)$ are

(a)
$$C_i(t) = \begin{cases} 0 & \text{if } t \le d_i \\ \alpha_i & \text{if } t > d_i \end{cases}$$

(b) $C_i(t) = \alpha_i \max\{0, t - d_i\} + \beta_i \max\{0, d_i - t\}$

Note that the tardiness function (b) includes the possibility of penalizing ($\beta_i > 0$) or rewarding ($\beta_i < 0$) early completion of job i. Also associated with each job i and all other jobs j are changeover costs f_{ij}. The problem is to sequence the N jobs so as to minimize the sum of changeover and tardiness costs.

We modify the TSP formulation (2.10) - (2.15) as follows. Let node 0 correspond to the current configuration of the machine. Let T denote an upper bound on the time to complete all jobs, and replicate the N nodes T times. For future reference, let T* denote a lower bound on the completion of all jobs. Finally, let node N+1 denote the desired terminal configuration of the machine.

The model contains several types of arcs. Node 0 is connected to nodes j for all j = 0 at time $p_0 + h_{0j}$ by arcs of length $c_{0j0} = f_{0j} + C_0(p_0)$. Each node j at each time t ($t \geq p_0 + h_{0j}$) is connected to node k for all k = j at time $t + p_j + h_{jk}$ by an arc with length $c_{jkt} = f_{jk} + C_j(t + p_j)$. In addition each node i at time t is connected by an arc to node i at time t + 1 with arc length $c_{iit} = 0$; these arcs are needed in those cases when it is advantageous to delay processing certain jobs. Finally, each node i at each time t ($t \geq T^*$) is connected to node N + 1 by an arcs with length $c_{i,N+1,t} = f_{i,N+1} + C_i(t + p_i)$.

The algebraic model statement is

<u>One Machine Sequencing Model</u>

$$\min \sum_{j=1}^{N} c_{0j0}x_{0j0} + \sum_{i=1}^{N} \sum_{t=p_i}^{T} \sum_{j=1}^{N} c_{ijt}\, x_{ijt} + \sum_{i=1}^{N} \sum_{t=T^*}^{T} c_{i,N+1,t}\, x_{i,N+1,t} \qquad (2.16)$$

$$\text{Subject to} \quad \sum_{j=1}^{N} x_{0j0} = 1 \qquad (2.17)$$

$$\sum_{k=1}^{N} x_{k,i,t-pk} + \sum_{j=1}^{N} x_{ijt} = 0 \qquad \text{for } i = 1,\ldots,N \quad t = 1,\ldots,T^*-1 \qquad (2.18)$$

$$\sum_{k=1}^{N} x_{k,i,t-pk} + \sum_{j=1}^{N+1} x_{ijt} = 0 \qquad \text{for } i = 1,\ldots,N \quad t = T^*,\ldots,T-1 \qquad (2.19)$$

$$\sum_{i=1}^{N} \sum_{t=T^*}^{T} x_{i,N+1,t} = 1 \qquad (2.20$$

$$x0j0 + \sum_{\substack{k=1 \\ k \neq j}}^{N} \sum_{t=p_k}^{T} x_{k,j,t} = 1 \qquad \text{for } j = 1,\ldots,N \qquad (2.21)$$

$$x_{ijt} = 0 \text{ or } 1 \qquad (2.22)$$

The statement of the one machine scheduling model is very similar to that of the TSP given above. The objective function (2.16) consists of three terms: initial costs, intermediate costs, and the final cost associated with proceeding to the terminal configuration N+1. Constraints (2.17) - (2.20) are material balance equations. Flow out of node i,t to node N+1 is allowed only after t has reached T*. The constraints (2.21) impose the restriction that every job must be scheduled.

The model just described allows the jobs 0,1,...,N to be performed in any order. In many job-shop scheduling applications, the order is restricted by precedence constraints reflecting the physical necessity to perform a certain job on a particular component before a second job is performed. To illustrate, suppose job i can be performed only after job k has been completed. Mathematically, we require for any t that x_{ijt} = 1 ==> x_{kgs} = 1 for some g=k and for some $s \leq t - p_k$. This logical condition can be expressed as the constraint

$$\sum_{t=p_k}^{T} \sum_{\substack{j=1 \\ j \neq i}}^{N} t\, x_{ijt} - \sum_{s=1}^{T-p_i} \sum_{\substack{g=1 \\ g \neq k}}^{N} s\, x_{kgs} \geq p_k$$

Since we need such a constraint for each precedence relation, it is apparent why job-shop scheduling problems are often extremely difficult to model and optimize. These problems are discussed again in Section 6.

3. LAGRANGE MULTIPLIER AND DECOMPOSITION METHODS

In this section, we review Lagrange multiplier and associated price-directed decomposition methods; see Geoffrion (1974), Shapiro (1979a), (1979b), or Fisher (1981) for more details. These methods will be applied to several large scale production planning and scheduling models in the sections that follow. Resource-directed decomposition, also known as Benders decomposition, is another relevant and important method, especially for mixed integer programming models. Unfortunately, space limitations prevent us from reviewing this method; the reader is referred to Shapiro (1979a) or Nemhauser and Wolsey (1988).

Decomposition methods for large-scale mathematical programming models, and for complex models with special structures that can be algorithmically exploited, were thoroughly studied during the period 1960-1975. A few applications have been successfully implemented. But, despite their attractive features, practitioners have shown little interest in the methods. Nevertheless, for reasons we are about to explain, the methods are worth examining here in detail.

A primary reason for the lack of applications is that decomposition methods can be inefficient on serial computers when compared with a monolithic (non-decomposition) approach. Of course, for very large models, a monolithic approach may not be possible. The imminent availability of commercial parallel computers should greatly renew interest in decomposition methods because the methods can exploit parallel architectures by distributing sub-models to separate processors. The methods allow solutions from these sub-models to be re-assembled in a rigorous manner to find a globally optimal solution. The combination of decomposition methods and parallel computers is particularly alluring for time critical decision support applications such as production scheduling. An important added feature of decomposition methods for these applications is that they provide objective function bounds on how far the best known solution is from optimality.

A second reason for the lack of applications is that decomposition methods need to be tailored to a particular application. This means first that greater effort is required to implement the software than that required by a monolithic method. Moreover, since decomposition methods perform best when provided with good starting information, the user must involve himself/herself more heavily in initialization procedures. Similarly, automatic re-start procedures should be implemented so that those sub-models that have changed since the most recent optimization can be given the greatest attention. Tailoring is also required to fully exploit the sub-model approximations and associated bounds provided by a decomposition approach. In short, the successful implementation and use of a decomposition method may require more work than a monolithic method, but it offers the potential for more flexible and efficient computation, particularly on a parallel computer.

In addition to this the potential afforded by parallel computers, decomposition methods are excellent pedagogical tools for presenting and examining production planning and scheduling models, particularly in describing how these models can be constructed as syntheses of smaller sub-models. Moreover, the methods mechanize concepts of economic equilibria by providing a rigorous means for computing them, thereby affording us with insights into the supply and demand forces that abound in many production environments.

We begin our mathematical development by considering the mathematical program

$$\begin{aligned} v = \min\ & cx \\ \text{Subject to}\quad & Ax \geq b \qquad\qquad (P) \\ & x \in X \end{aligned}$$

which we refer to as the primal problem. The constraints in (P) have been partitioned into the easy to solve implicit constraints $x \in X$, and the more difficult to solve, or complicating, constraints $Ax \geq b$. The matrix A is an $m \times n$. For the vast majority of cases of interest to us here, the set X is finite and can be enumerated as $\{x^t \mid t=1,..,T\}$. For example, X might consist of the finite set of zero-one solutions of an integer program, or the finite set of extreme points of a linear programming model.

An alternate form of (P) that we will sometimes employ is

$$v = \min c_1x_1 + c_2x_2$$

Subject to $A_{01}x_1 + A_{02}x_2 \geq b_0$ (P')

$$x_1 \in X_1, \quad x_2 \in X_2$$

where

$$X_1 = \{x_1 | A_{11}x_1 \geq b_1, \ x_1 \geq 0\} \quad X_2 = \{x_2 | A_{22}x_2 \geq b_2, \ x_2 \geq 0\}$$

The model (P') is no more than a structured linear program. Note that it would decompose into two separate models if it were not for the linking constraints $A_{01}x_1 + A_{02}x_2 \geq b_0$.

Returning to model (P), we attempt to eliminate the complicating constraints by pricing them out and placing them in the objective function. The result is the Lagrangean function defined for any m-vector $u \geq 0$

$$L(u) = ub + \min (c - uA)x$$

Subject to $x \in X$

Optimization of the Lagrangean produces the solution $x(u) \in X$. A central question is: When is $x(u)$ optimal in (P)?

The answer to this question is related to the following conditions: We say $x \in X$, $u \geq 0$ satisfy the global optimality conditions for (P) if

(i) $L(u) = ub + (c - uA)x$

(ii) $u(Ax-b) = 0$

(iii) $Ax \geq b$

The global optimality conditions are sufficient conditions for optimality as demonstrated by the following theorem that we state without proof.

THEOREM 1: If $x \varepsilon X$, $u \geq 0$ satisfy the global optimality conditions, then x is optimal in (P).

The global optimality conditions are not necessary in the following sense. For a given x optimal in (P), there may be no $u \geq 0$ such that the global optimality conditions hold. This is frequently the case when X is a discrete or otherwise non-convex set. It is easy to demonstrate, however, that $L(u) \leq v$ regardless of the structure of X.

A second important question is: How should we choose u? The answer is to search for the best lower bound; namely to solve

$$d = \max_{u \geq 0} L(u) \qquad (D)$$

We call this the <u>dual problem</u>. L is a concave function of u and therefore (D) is a well behaved mathematical program. Clearly, $d \leq v$; if $d < v$, we say there is a <u>duality gap</u>.

The following theorem tells us that we need consider only optimal solutions to (D) in attempting to solve (P) via the global optimality conditions.

THEOREM 2: If $x \varepsilon X$, $u \geq 0$, satisfy the global optimality conditions, then u is optimal in (D). Moreover, $d = L(u) = v$; that is, there is no duality gap.

To summarize, a goal of the Lagrangean analysis is to compute some $u \geq 0$ that is optimal in (D), and then seek an $x \varepsilon X$ such that the global optimality conditions hold at that optimal u. If X is not convex, which is the case with integer programming and combinatorial optimization models, this objective may not be met for any optimal u and any $x \varepsilon X$. These are the primal models for which there is a duality gap. Geoffrion (1974) discusses properties of integer programming models and their Lagrangean analysis that ensure no duality gap.

It is important to recognize, however, that the Lagrangean constructions derived from (P) can be useful even when there is a duality gap. First, the bounds from the Lagrangean analysis can be employed in a branch and bound scheme for optimizing (P); see Shapiro (1979a), (1979b). Second, the Lagrangean analysis may produce a feasible solution $x \varepsilon X$ to (P) (conditions (i) and (iii) hold), but the complementary slackness condition (ii) may fail to hold. In such an event, the quantity $u(Ax - b) > 0$ is an upper bound on the duality gap. If it is sufficiently small, the search for an optimal solution to (P) can be abandoned.

Imbedded in a branch and bound scheme, the Lagrangean analysis just described is complementary to heuristic methods for mixed integer programming and other combinatorial optimization models. Optimizing the Lagrangean will only rarely produce a feasible or provably optimal solution to (P). By contrast, problem specific heuristics applied to each sub-model in a branch and bound search are intended to yield good feasible solutions to (P). A heuristic may even employ information from the Lagrangean; for example, the heuristic may be based in part on the relative importance of the variables x_j as measured by the reduced cost values $c_j - ua_j$ where u is a "good" dual solution. If v^* is the cost of the best known solution found by the heuristic, the quantity $v^* - L(u^*)$ is a bound on the objective function error if we decide to terminate with this solution, where u^* is the computed dual solution that produces a maximal dual lower bound. Heuristics and Lagrangean analyses have been employed in this manner by Christofides et al (1987) for a job-shop scheduling model,and by Aftentakis and Gavish (1986) for a discrete parts manufacturing model.

Moreover, if branch and bound continues, the <u>surrogate constraint</u>

$$(c - u^*A)x \leq v^* - u^*b$$

added to (P) will serve to limit the search for a better solution (e.g., see Karwan and Rardin (1979).

We consider briefly two algorithmic methods for optimizing the dual problem (D). The first is an ascent method called <u>subgradient optimization</u>. Suppose L has been evaluated at $u \geq 0$. Let $x \epsilon X$ satisfy

$$\begin{aligned} L(u) &= ub + (c-uA)\,x \\ &= cx + u(b - Ax) \end{aligned}$$

Define $\tau = b - Ax$; this m-vector is called a subgradient of L at u. It points into the half space containing all optimal solutions. That is, if u^* is optimal in (D), $(u^* - u)\,\tau \geq 0$. This property is sufficient to ensure convergence of the following algorithm despite the fact that τ may not actually point in a direction of increase of L at u.

The algorithm is to choose a sequence $\{u^k\}$ of dual solutions by the formula

$$u_i^{k+1} = \max \{0,\ u_i^k + \frac{\alpha_k\ (d-L(U^k)}{||\tau^k||^2} \cdot \tau_i^k\} \qquad \text{for } i=1,\ldots,m.$$

where $0 < \varepsilon_1 < \alpha_k < 2-\varepsilon_2 < 2$ and $||\cdot||$ denotes Euclidean norm. It can be shown that u^k converges under weak conditions to an optimal dual solution u^*. The difficulty, however, is that we do not know the value of d. Thus, we must guess at d and therefore subgradient optimization is essentially a heuristic method.

The other algorithm we discuss for optimizing the dual problem (D) is generalized linear programming. This approach to mathematical programming pre-dates, and in a sense anticipates, the development of general mathematical programming dual methods. Since $X = \{x^t \mid t=1,\ldots,T\}$, we may write

$$L(u) = \underset{t=1,\ldots,T}{\text{minimum}} \{ub + (c-uA)x^t\}$$

This representation of L permits us to express (D) as the linear program

$$\begin{aligned} d = {} & \max w \\ \text{s.t.} \quad & w \le ub + (c-uA)x^t \qquad \text{for } t=1,\ldots,T \\ & u \ge 0 \end{aligned}$$

because, for any $u \ge 0$, the value achieved by w, say w(u), equals L(u). Taking the dual of this LP, we obtain

$$d = \min \sum_{t=1}^{T} (cx^t)\ \Omega_t$$

$$\text{s.t.} \sum_{t=1}^{T} (Ax^t)\ \Omega_t \ge b$$

$$\sum_{t=1}^{T} \Omega_t = 1$$

$$\Omega_t \geq 0$$

The difficulty with this representation of the dual problem is that the number T of solutions in X will generally be too large to allow explicit enumeration. Generalized linear programming proceeds by generating solutions implicitly. Thus, suppose we have somehow obtained the solutions $x^t \epsilon X$, t=1,...,K where K is a reasonable, practical number. The method begins by using the simplex method to solve the LP

<u>GLP Master Model</u>

$$d^K = \min \sum_{t=1}^{K} (cx^t)\, \Omega_t$$

$$\text{s.t.} \quad \sum_{t=1}^{K} (Ax^t)\, \Omega_t \geq b$$

$$\sum_{t=1}^{K} \Omega_t = 1$$

$$\Omega_t \geq 0$$

Let Ω_t^k, t=1,...,K, denote optimal values, let u^K denote optimal LP dual variables for the constraint rows, and let θ^K denote an optimal dual value for the convexity row. Note that $d^K \geq d$ since columns have been omitted in the Master Model.

The next step is to solve the Lagrangean sub-model $L(u^K)$ to obtain the solution $x^{K+1} \varepsilon X$ that satisfies

$$\begin{aligned} L(u^K) &= u^K b + (c-u^K A)\, x^{K+1} \\ &= u^K b + \min_{x \varepsilon X} (c-u^K A)\, x \end{aligned}$$

Note that we have reverted to the original definition of L involving minimization over the set X in its given form.

There are two possible outcomes as the result of the Lagrangean optimization.

Case One:

$$L(u^K) - u^K b = (c-u^K A) x^{K+1} < \Theta^K$$

In this case, the solution x^{K+1} is represented by a new column in the master model with variable $\Omega^K{}_{+1}$. The new column will cause the master model solution to change since its reduced cost in the previously optimal LP is negative; namely,

$$cx^{K+1} - uAx^{K+1} - \Theta^K < 0$$

Case Two:

$$L(u^K) - u^K b = (c-u^K A) x^{K+1} = \Theta^K$$

Note that $L(u^K) - u^K b$ cannot be greater than Θ^K since the basic columns in the master model price out zero. In this case, we have

$$L(u^K) = u^K b + \Theta^K = d^K \geq d \geq L(u^K)$$

where the second equality follows from LP duality. Thus, $L(u^K) = d$ and u^K is optimal in the dual.

Convergence of the generalized programming method to case two is finite since we have assumed there are only a finite number of possible columns to be added. As discussed above, when an optimal u^* has been obtained, we seek an $x \varepsilon X$ such that the global optimality conditions hold at u^*.

We return for the moment to the alternate primal problem (P'). For $u \geq 0$,

$$L(u) = ub + F_1(u) + F_2(u)$$

where $$F_i(u) = \min\ (c_i - uA_{0i})\, x_i \quad \text{s.t.} \quad x_i \varepsilon X \qquad \text{for } i = 1,2$$

Thus, the optimization of (P') separates. Moreover, since the submodels i are LP's, LP duality ensures that the global optimality conditions will hold for some u. For (P') the master model becomes

$$d^K = \min \sum_{t=1}^{K} (c_1 x_1^t)\,\Omega_t = \sum_{t=1}^{K} (c_2 x_2^t)\,\beta_t$$

$$\text{s.t.} \sum_{t=1}^{K} (A_{01} x_1^t)\,\Omega_t + \sum_{t=1}^{K} (A_{02} x_2^t)\,\beta_t \geq b$$

$$\sum_{t=1}^{K} \Omega_t = 1$$

$$\sum_{t=1}^{K} \beta_t = 1$$

$$\Omega_t \geq 0, \quad \beta_t \geq 0$$

Letting Ω_t^K, β denote optimal values and u^K, θ_1^K, θ_2^K denote optimal LP dual variables for the master model, we optimize the two LP sub-models $F_1(u^K)$. If

$$F_1(u^K) = \theta_1^K$$

and

$$F_2(u^K) = \theta_2^K,$$

the method terminates. In this case, since (P') is an LP,

$$x_1^* = \sum_{t=1}^{K} (x_1^t) \Omega_t^K \text{ and } x_2^* = \sum_{t=1}^{K} (x_2^t) \beta_t^K$$

are optimal solutions in (P'). Otherwise, we add a column to one or both of the two sets in the Master Model.

4. PROCESS MANUFACTURING MODELS

The first significant applications of mathematical programming were to oil refinery, food processing and steel manufacturing planning (see Dantzig (1963) for a discussion of these early industrial applications). Through the years similar types of models have been developed and applied to process manufacturing of other products such as chemicals, paper, soap, and industrial gases. Historically, these modeling applications have been mainly for tactical planning; for example, determining an optimal monthly production plan for a chemical plant. However, due to advances in computer and process control technologies, and to growing international competition that is spurring companies to seek greater efficiencies, many process manufacturing companies have introduced or are now considering scheduling systems based in part on mathematical programming models.

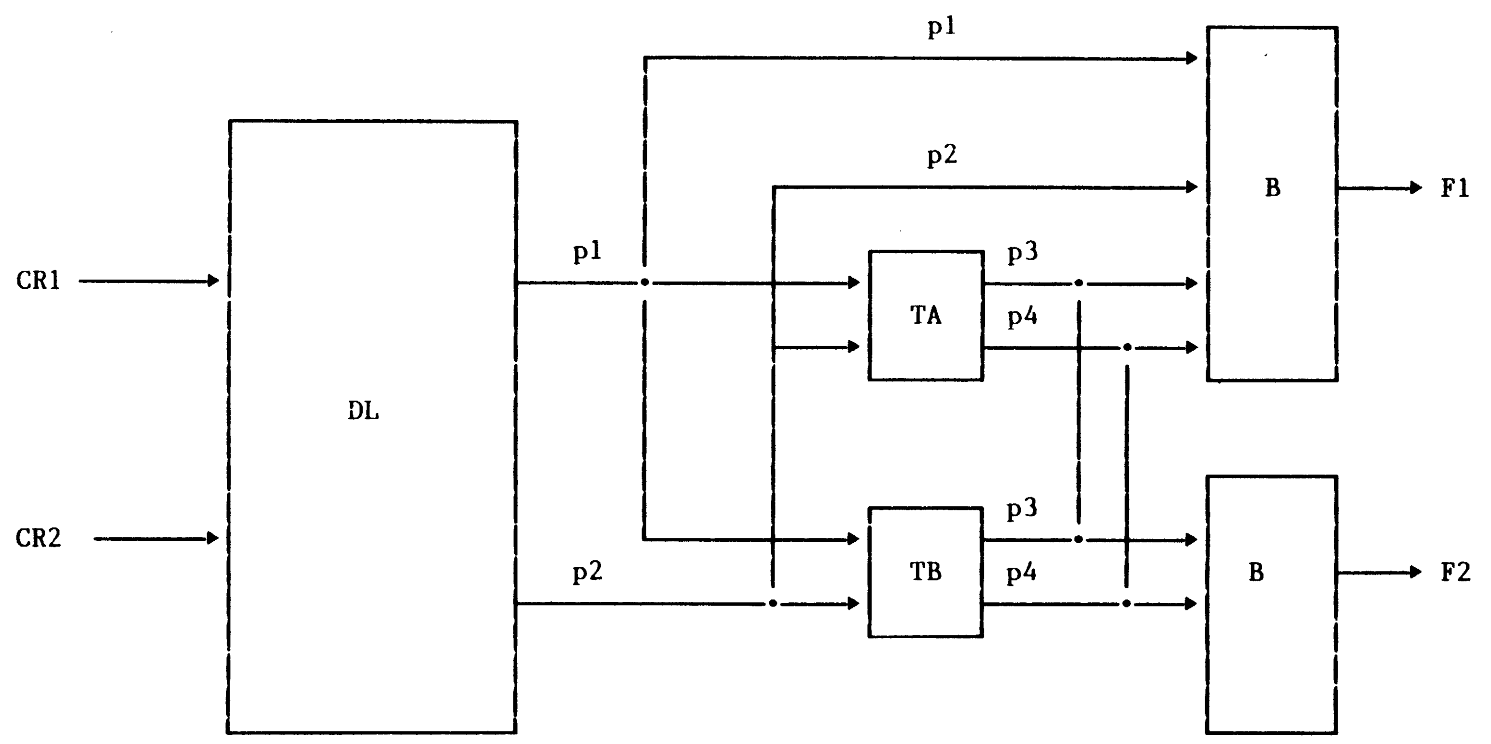
Process manufacturing planning and scheduling problems are characterized by the following features: (1) expensive machines and plants that must be operated near capacity if investments are to be profitably amortized; (2) for each stage of production, product transformation activities that are smooth and continuous (although not necessarily linear) as long as the equipment associated with a stage remains in the same major configuration; (3) time consuming and sometimes costly change-overs that occur whenever equipment undergoes an alteration in its major configuration.

Process manufacturing planning models ignore changeovers, and treat products grouped into aggregate families. Process manufacturing scheduling models explicitly consider changeovers and consider products in greater detail, including the shipment of specific orders for specific products to specific customers. In the paragraphs that follow, we discuss first process manufacturing planning models, and then process manufacturing scheduling models.

4.1 OIL REFINERY EXAMPLE

We begin by considering a numerical example of an oil refinery planning model. The processing stages of the refinery are depicted in Figure 4.1. The first stage at the left performs a distillation of two crudes CR1 and CR2. The outputs of the distillation, P1 and P2, are transformed by different recipes to produce intermediate products P3 and P4. The final stage is blending the four intermediate products P1, P2, P3, P4 into the final products F1 and F2.

Table 4.1 is a tabular display of a linear programming model to optimize production over a single period which we consider to be 72 shifts (24 days) long. The first row (REV) is the net revenue to be maximized over that period. The costs of acquiring crudes and transforming them to finished products are subtracted from the gross revenues accruing from the sale of finished products. Note that no cost is associated with blending; this



Flow Chart of an Oil Refinery

Figure 4.1

	BYCR1	BYCR2	DLCR1	DLCR2	TAP1	TAP2	TBP1	TBP2	BF1P1	BF1P2	BF1P3	BF1P4	BF2P3	BF2P4	SLF1	SLF2	RHS
REV	-20	-22	-1.20	-1.25	-.50	-.52	-.61	-.68							31	33	
MBCR1	-1		1														=0
MBCR2		-1		1													=0
MBP1			-.48	-.49	1		1		1								=0
MBP2			-.46	-.48		1		1		1							=0
MBP3					-.55	-.59	-.65	-.60			1		1				=0
MBP4					-.37	-.34	-.31	-.38				1		1			=0
CYVA					1	1											≤72
CYVB							1	1									≤72
MBF1									-1	-1	-1	-1			1		=0
QAF1									-15	-17	-24	-30			20		≤0
MBF2													-1	-1		1	=0
QAF2													-21	-30		25	≤0
LB															105	45	
UB	170	160													135	80	

Statement of the Refinery Model

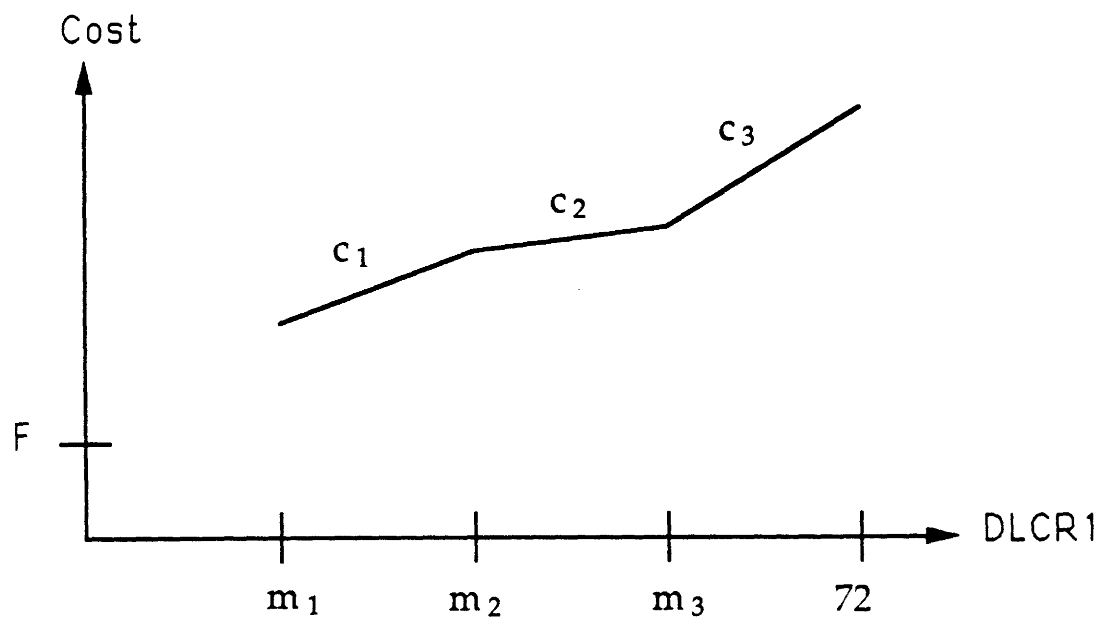
Table 4.1

cost is netted out of the unit sales price of the finished products.

The first six constraints of the model are material balance equations on the crudes CR1 and CR2 into the distiller, and the intermediate products that result from the distillation and the two process operations TA and TB. The following two inequalities reflect capacity available for the operations TA and TB over the 72 shift period. The next four rows are material balance and quality constraints (eg, octane rating) on the final products F1 and F2. For example, the QAF1 inequality states that the quality of the blended product F1 must be at least 20. Finally, there are upper bound constraints on the quantities of crudes that can be bought, and upper and lower bounds on the quantities of finished products that can be sold.

The linear programming model just discussed assumes that all transformation activities are linear and additive. Although this assumption has been accepted through the years by practitioners responsible for implementing refinery planning models, important nonlinearities do occur. Advanced modeling techniques involving mixed integer programming and/or nonlinear programming can be applied to achieve greater accuracy in describing operations.

Mixed integer programming constructs can easily capture nonlinearities if they can be described by separable functions. For example, Figure 4.2 depicts the non-linear cost of the process transformation activity TAP1 as a function of the DLCR1 input to that process. The quantity F is the fixed cost of



Nonlinear Operational Cost Curve
Approximated by Mixed Integer
Programming

Figure 4.2

setting-up the process activity, m_1 is the conditional minimum operating level for the process, c_1 is the initial unit cost of operations, c_2 is a reduced unit cost of operations that prevails once the level m_2 has been reached, and c_3 is the increased unit cost of operating in the range m_3 to 72 that is very close to capacity.

The nonlinear characteristics of chemical processing activities may be too complex to easily allow the type of mixed integer programming approximation just presented. In particular, if cross-product terms involving decision variables are important to accurately describe a chemical process, the formalisms of nonlinear programming are probably required. These points are discussed again in Section 4.3 below.

4.2 DECOMPOSITION EXAMPLE

We illustrate the price-directed decomposition method discussed in section 3 by applying it to the process planning problem depicted in Figure 4-1 and modeled in Table 4-1. Specifically, we decompose the model by splitting it into two parts; one part consists of the distillation and transformation activities and constraints, and the other consist of the blending activities. We note that this decomposition is based on the block diagonal form of the matrix in Table 4-1

$$\begin{matrix} D & T \\ 0 & B \end{matrix}$$

where B consists of the lowest four rows (excluding lower and upper bounds) and the right-most eight columns of the matrix. The master model is associated with D and T, whereas the sub-model is associated with B.

The master model is initialized with the four feasible blending solutions shown in Table 4.2; note that these solutions satisfy the blending constraints. The four solutions are

	SOLN 1	SOLN 2	SOLN 3	SOLN 4
B1P1	60	0	0	32
B1P2	0	65	45	32
B1P3	0	65	45	32
B1P4	60	0	45	32
B2P4	40	30	30	35
B2P4	40	30	50	45

Table 4.2

	SOLN 1	SOLN 2	SOLN 3	SOLN 4
BP1	60	0	0	32
BP2	0	65	45	32
BP3	40	95	75	67
BP4	100	30	95	77
GROSS REVENUE	6360	6010	6925	6608

Table 4.3

directly translated to the associated four product inputs and gross revenue outputs shown in Table 4.3. We use these data in constructing the master model

```
MAX  6360 L1 + 6010 L2 + 6825 L3 + 6608 L4 - 20 CR1 - 22 CR2
- 1.2 DLCR1 - 1.25 DLCR2 - 0.5 TAP1 - 0.52 TAP2 - 0.61 TBP1
- 0.68 TBP2
 SUBJECT TO
        2) - CR1 + DLCR1 =    0
        3) - CR2 + DLCR2 =    0
        4)   60 L1 + 32 L4 - 0.48 DLCR1 - 0.49 DLCR2 + TAP1
                + TBP1 = 0
        5)   65 L2 + 45 L3 + 32 L4 - 0.46 DLCR1 - 0.48 DLCR2
              + TAP2 + TBP2 = 0
        6)   40 L1 + 95 L2 + 75 L3 + 67 L4 - 0.55 TAP1
                - 0.59 TAP2 - 0.65 TBP1 - 0.6 TBP2 = 0
        7)   100 L1 + 30 L2 + 95 L3 + 77 L4 - 0.37 TAP1
                - 0.34 TAP2 - 0.31 TBP1 - 0.38 TBP2 = 0
        8)   TAP1 + TAP2 <=   72
        9)   TBP1 + TBP2 <=   72
       10)   CR1 <=   180
       11)   CR2 <=   160
       12)   L1 + L2 + L3 + L4 =    1
```

The last constraint in this model requires that the weights L1, L2, L3, L4 on the four blending strategies sum to one. This convexification provides the master model with a useful but incomplete description of the blending capabilities of the refinery.

The decomposition proceeds by optimizing the master model. The solution is:

```
        OBJECTIVE FUNCTION VALUE

               1512.31900

  VARIABLE        VALUE          REDUCED COST
        L1          .000000        117.867400
        L2          .544085           .000000
        L3          .000000         52.819460
        L4          .455915           .000000
       CR1       180.000000           .000000
       CR2        37.518890           .000000
     DLCR1       180.000000           .000000
     DLCR2        37.518890           .000000
      TAP1        69.049250           .000000
      TAP2          .000000           .567981
      TBP1        21.145720           .000000
      TBP2        50.854280           .000000
```

```
ROW   SLACK OR SURPLUS      DUAL PRICES
 2)           .000000        21.327400
 3)           .000000        22.000000
 4)           .000000        23.630510
 5)           .000000        24.314690
 6)           .000000        24.738380
 7)           .000000        28.444320
 8)          2.950755          .000000
 9)           .000000          .657178
10)           .000000         1.327402
11)        122.481100          .000000
12)           .000000      1226.069000
```

The values of the weights imply a blending strategy B1P1 = 14.592, B1P2 = 49.942, B1P3 = 49.952, B1P4 = 14.592, B2P3 = 32.280, B2P4 = 36.840. This strategy, along with the solution listed above, represent a feasible solution to the original LP model with net revenue equalling 1512.319.

Optimizing the master model also produces shadow prices on the four intermediate products BP1, BP2, BP3, and BP4 produced by the distillation and processing activities: $\pi_1 = 23.631$, $\pi_2 = 24.315$, $\pi_3 = 24.738$, $\pi_4 = 28.444$, where π_i = shadow price on product BPi. These represent the master model's best estimate of the unit cost to provide each of these products to the blending activities. The decomposition continues by solving the sub-model in which the blending activities are charged these unit prices for the products that consume. The sub-model is

```
MAX  31 F1 + 33 F2 - 23.631 B1P1 - 24.315 B1P2 - 24.738 B1P3
 - 28.444 B1P4 - 24.738 B2P3 - 28.444 B2P4

 SUBJECT TO
        2)   F1 - B1P1 - B1P2 - B1P3 - B1P4 = 0
        3) - 5 B1P1 - 3 B1P2 + 4 B1P3 + 10 B1P4 >= 0
        4)   F2 - B2P3 - B2P4 = 0
        5) - 4 B2P3 + 5 B2P4 >= 0
        6)   F1 >= 105
        7)   F1 <= 135
        8)   F2 >= 45
        9)   F2 <= 80
```

The optimal solution to the sub-model is

```
        OBJECTIVE FUNCTION VALUE

               1440.98100

  VARIABLE          VALUE           REDUCED COST
        F1       135.000000             .000000
        F2        80.000000             .000000
      B1P1        60.000000             .000000
      B1P2          .000000             .438000
      B1P3        75.000000             .000000
      B1P4          .000000            2.968000
      B2P3        44.444440             .000000
      B2P4        35.555560             .000000
```

This solution suggests the new strategy

```
     BP1                      60
     BP2                       0
     BP3         =           119.44
     BP4                      35.56
```

to add to the master model with gross revenue = 6825 and variable L5.

The new master model optimal solution is

```
        OBJECTIVE FUNCTION VALUE

              1529.45400

  VARIABLE          VALUE           REDUCED COST
        L1          .000000           17.710450
        L2          .439749             .000000
        L3          .000000          192.848000
        L4          .480502             .000000
        L5          .079749             .000000
       CR1       180.000000             .000000
       CR2        40.124490             .000000
     DLCR1       180.000000             .000000
     DLCR2        40.124490             .000000
      TAP1        72.000000             .000000
      TAP2          .000000             .332096
      TBP1        13.900010             .000000
      TBP2        58.099990             .000000
```

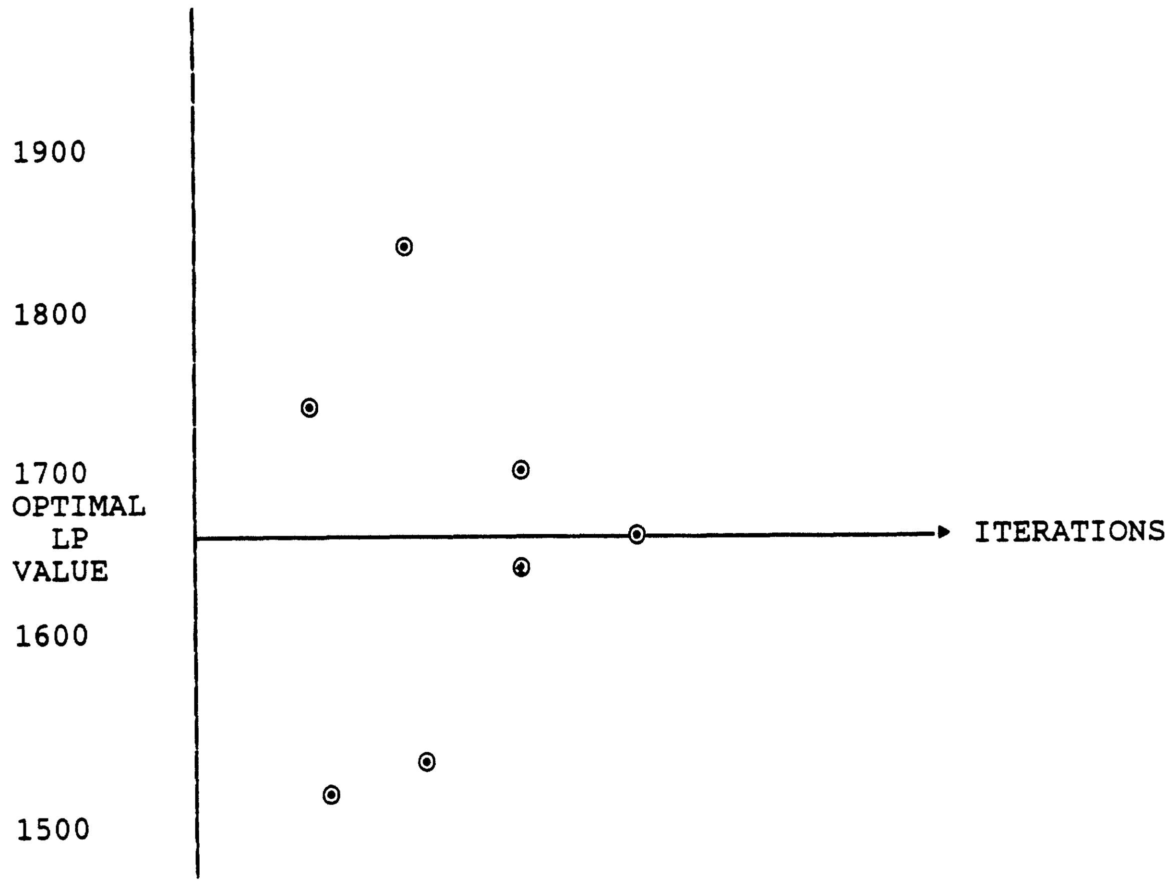
```
ROW   SLACK OR SURPLUS     DUAL PRICES
 2)          .000000         21.327740
 3)          .000000         22.000000
 4)          .000000         23.663310
 5)          .000000         24.281200
 6)          .000000         32.340550
 7)          .000000         32.927430
 8)          .000000          5.807140
 9)          .000000          6.955549
10)          .000000          1.327744
11)       119.875500           .000000
12)          .000000        371.546600
```

Note that the new master model activity associated with the variable L5 is used to advantage in maximizing net revenues. Note also that the master model has revised its estimates of the shadow price charges on rows 4, 5, 6, and 7 for the intermediate products BP1, BP2, BP3, BP4.

The communication between the master model and the sub-model continued for four iterations until an optimal solution to the original model was computed. Figure 4.3 depicts the net revenue of the feasible solution found by the master model at each iteration. It also depicts the upper bound determined by the sub-model at each iteration. At the fourth iteration, the lower and upper bounds converged on the optimal LP value.

4.3 SUCCESSIVE LINEAR PROGRAMMING

Nonlinearities arise in refining, petrochemical and other process manufacturing models due to a variety of factors. A major factor is the need to model properties or qualities of product flows as well as the flows themselves. For example, when intermediate products are combined in a tank or pool, nonlinear



Bounds Produced by Decomposition Method

Figure 4.3

relationships are required to capture pool qualities. Mathematically, the simplest form of the pooling problem arises when streams of intermediate products are blended on the basis of quality-volume relationships of the form

$$V_p = \sum_i V_i$$

$$q_p = \left(\sum_i q_i V_i\right) / V_p$$

where V_i and q_i represent the volume and quality respectively of incoming stream i. The volume of the pool is represented by V_p. The expression for q_p, the quality of the pool, is a nonlinear function of the volume V_i.

Note that if p were a final product constrained, say, to be above some level q, then the relationship $q_p \geq q$ could be linearized simply by multiplying both sides of the inequality thereby creating

$$\sum_i (q_i - q) V_i \geq 0$$

This is the type of constraint we had on the blended final products in the refinery model displayed in Table 4.1. The difficulty occurs, and thus the need for nonlinear programming techniques, when the intermediate pooled products are themselves pooled in downstream operations.

Nonlinearities may also arise in blending final products if the qualities of the component streams affect the qualities of the blended product in a nonlinear manner. Baker and Lasdon (1985) cite the example of gasoline blending where the octane contribution of components depends radically on the presence of lead in the blend. Nonlinear process yields may similarly occur as functions of operating conditions of the process unit. Finally, costs may prove to be nonlinear functions of operating variables such as throughput or severity. As we pointed out in section 4.1, however, many of these nonlinearities can be approximated as piecewise linear functions. Zero-one variables are required to capture any non-convexities in these curves.

Linear programming approximation techniques for capturing these types of nonlinearities in process manufacturing have proven effective for over 20 years. Baker and Lasdon (1985) provide a survey of the more popular methods. They discuss two specific approaches. One is a successive linear programming modeling technique based on first order Taylor series expansions of the nonlinear functions. The expansions are systematically updated based on user supplied tolerances on the range of the expansions and the optimal values of variables in successive linear programming approximations.

In some instances, however, the nonlinearities are not available in functional form to be expanded in Taylor series. Rather, the nonlinear relationships are implicitly captured by process simulation models. An approach suggested for this case is to use the process simulators to identify trial production strategies for various units in a plant, and to construct a plant model that allows convex combinations of these strategies in the spirit of price-directed decomposition. Another approach is to employ the simulators to compute function values and approximate derivatives which become input data to successive linear programming models. The reader is referred to Biegler (1985), Biegler and Hughes (1985), and Lasdon, Waren and Sarkar (1988) for more details.

4.4 SCHEDULING MODELS

The planning models discussed above were not concerned with sequencing operations in a process manufacturing environment. In this section, we discuss a detailed model that addresses the timing of configuration changes and the management of inventories. The model suggested is appropriate, for example, for optimizing the production schedule over a given planning horizon of one or several linked paper machines in a mill, or for optimizing production in an industrial gases plant.

Indices:

i	:	time periods $i = 1, 2, \ldots, N$
r	:	configuration states $r = 0, 1, 2, \ldots, R$
j	:	process manufacturing slates
J_r	:	slates available under configuration r
k	:	products $k = 1, 2, \ldots, K$
Q	:	$(r1, r2) : r1 \neq r2,\ r1, r2, = 0, 1, 2, \ldots, R$

T	:	set of time indices for production targets

Parameters:

c_j	=	rate at which direct cost is incurred using slate j (\$/hr)
f_{rv}	=	cost of changing the configuration from r to v (\$)
d_{ik}	=	cumulative target production for product k at the end of period i for $i \varepsilon T$ (#)
p_{ik}^+	=	penalty for each unit of product k exceeding the target at the end of period i (\$/#)
p_{ik}^-	=	penalty for each unit of product k falling below the target at the end of period i (\$/#)
a_{jk}	=	rate at which product k is produced by slate j (#/hr)
m_{iw}	=	quantity of raw material w available in period i (#)
q_{jw}	=	rate at which raw material w is consumed by slate j (#/hr)
t_i	=	time when period i ends
t_{rv}	=	changeover time from configuration r to configuration v (hrs)

Variables:

x_{ij} = time that slate j is used in period i

$$y_{irv} = \begin{cases} 1 & \text{if configuration } v \text{ is run in period } i \text{ and configuration } r \neq v \text{ is run in period } i-1 \\ 0 & \text{otherwise} \end{cases}$$

$$z_{ir} = \begin{cases} 1 & \text{if configuration } r \text{ is run in period } i \\ 0 & \text{otherwise} \end{cases}$$

s_{ik}^+	=	surplus cumulative production of product k above target at the end of period $i \varepsilon T$
s_{ik}^-	=	unmet cumulative production of product k below target at the end of period $i \varepsilon T$

Process Manufacturing Scheduling Model

$$\min \sum_{i=1}^{N} \sum_{r=0}^{R} \sum_{j \in J_r} c_j x_j + \sum_{i=1}^{N} \sum_{(r1,r2) \in Q} f_{r1,r2} \; y_{i,r1,r2}$$

$$+ \sum_{i \in T} \sum_{k=1}^{K} (p^{+}_{ik} s^{+}_{ik} + p^{-}_{ik} s^{-}_{ik}) \qquad (4.1.1)$$

Subject to:

$$\sum_{r=1}^{R} \sum_{j \in J_r} q_{jw} x_{ij} \le m_{iw} \qquad i=1,\ldots,N \qquad (4.1.2)$$

$$\sum_{i=1}^{g} \sum_{r=1}^{R} \sum_{j \in J_r} a_{jk} x_{ij} + s^{-}_{gk} - s^{+}_{gk} = \sum_{i=1}^{g} d_{ik} \qquad (4.1.3)$$

$$k=1,\ldots,K \quad g \in T$$

$$\sum_{r=0}^{R} \sum_{j \in J_r} x_{ij} + \sum_{(r1,r2) \in Q} t_{ri,r2} \, y_{i,r1,r2} = t_i - t_{i-1} \qquad (4.1.4)$$

$$i=1,\ldots,N$$

$$\sum_{j \in J_r} x_{ij} - (t_i - t_{i-1}) z_{ir} \le 0 \qquad r=0,\ldots,R \quad i=1,\ldots,N \qquad (4.1.5)$$

$$\sum_{r=0}^{R} z_{ir} = 1 \qquad i=1,\ldots,N \qquad (4.1.6)$$

$$y_{i,r1,r2} \ge z_{i-1,r1} + z_{i,r2} - 1 \qquad (4.1.7)$$

for all $(r1,r2) \in Q$ and $i=1,\ldots,N$

$$x_{ij} \ge 0, \quad y_{i,r1,r2} = 0 \text{ or } 1, \quad z_{ir} = 0 \text{ or } 1, \quad s^{+}_{ik} \ge 0, \quad s^{-}_{ik} \ge 0 \qquad (4.1.8)$$

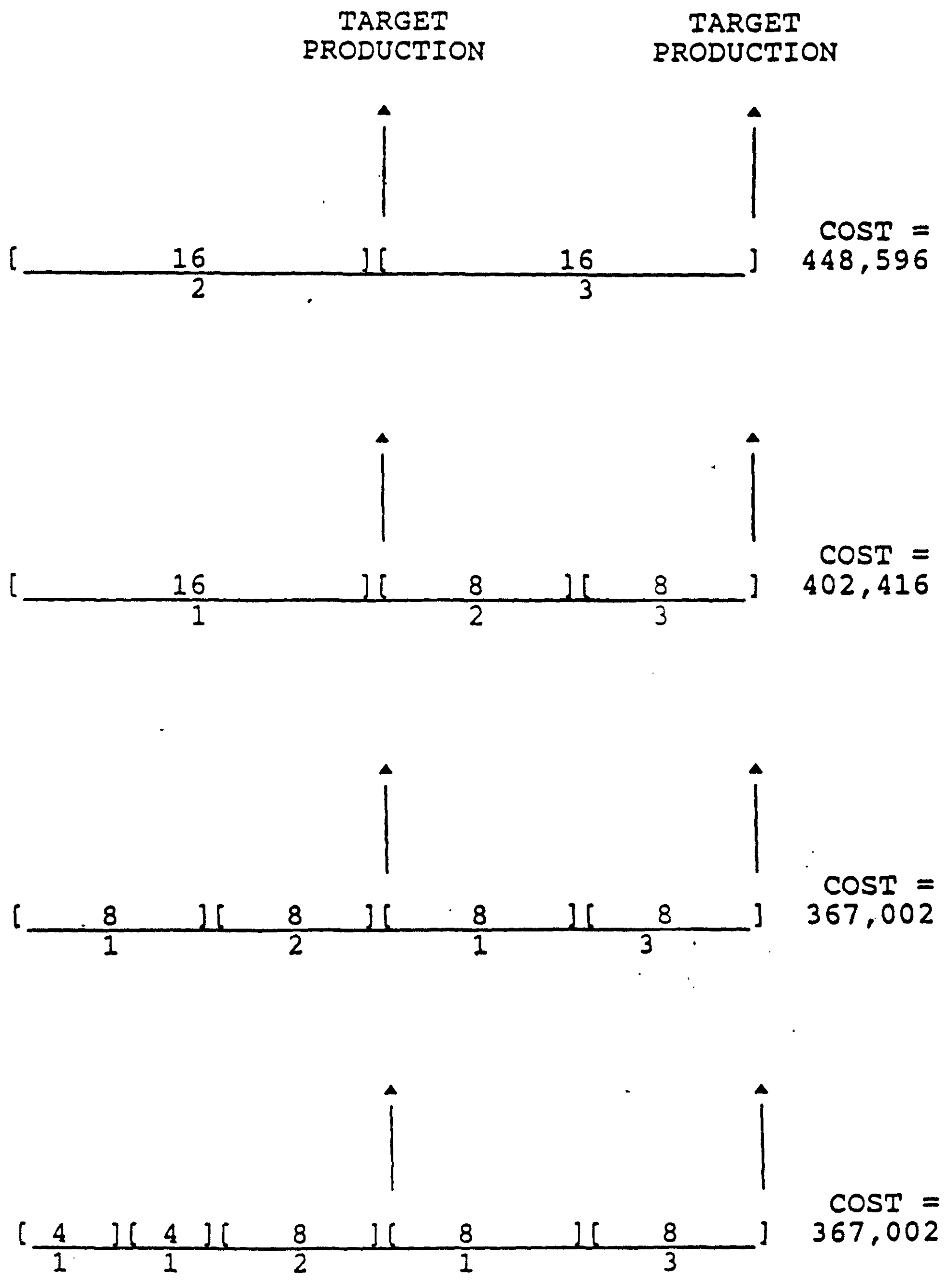
A brief explanation of the mixed integer programming model is the following. The objective function (4.1.1) is comprised of direct manufacturing costs, changeover costs, and penalty costs for being above or below cumulative production targets at certain specified points in time. The relations (4.1.2) state that production in each time period is constrained by the amount of raw materials available for that period and the rate at which they are consumed. The equations (4.1.3) relate cumulative production to the production targets. Note that r=0 corresponds to the shut-down configuration for which there is no production; hence this term has been omitted from (4.1.3).

The equations (4.1.4) constrain production and change-over time in each time period to equal the time available. The constraints (4.1.5) and (4.1.6) effectively select one major configuration for each period, and restrict the slates that may be used to that configuration. The constraints (4.1.7) provide the correct relation between the changeover variables and the configuration selection variables; namely, a changeover from configuration r1 to configuration r2 is allowed ($z_{i-1,r1} = 1$ and $z_{i,r2} = 1$) only if the changeover variable $y_{i,r1,r2} = 1$ thereby forcing the changeover costs and times to be incurred.

The number of constraints and variables in this model is largely dependent on NR^2. Thus, if there are 4 major configurations and if one wishes to determine hourly plans for a week, the model will have in excess of 168*16=2688 constraints and variables. This is an overly large model, even for a supercomputer. Moreover, most of the variables and constraints are not active in the sense that very few major configuration changes will actually take place.

An iterative modeling approach under investigation (Shapiro (1988)) is intended to greatly reduce difficulties due to size by beginning with a coarse time grid and refining time periods where optimization indicates the model is undecided about the major configuration it desires for a particular period. Re-optimization of a refined model can be quite rapid because the optimal solution for a parent model becomes a feasible solution in its refined descendant. In many instances, the major goal of the model analysis would be to determine the timing of the next (that is, the first) major change in configuration.

Figure 4.3 depicts an implementation using realistic data for industrial gas production (see Brown, Shapiro and Singhal (1987)). The planning horizon consists of 32 periods with targets on production at the end of periods 16 and 32. Five major configuration states, including shut-down, are allowed; the system is initially in configuration state 1. Four runs were made with iterative refinement of the time periods. The numbers below the axis indicate the configuration in each time period. The final run indicated that the current configuration should not be changed until 8 periods into the planning horizon, and then the change should be to configuration 2. Although we have shown a linear sequence of approximations, the refinement should be elaborated in a tree structure in the manner of branch-and-bound.



Iterative Process Manufacturing Scheduling Models

Figure 4.3

5. DISCRETE PARTS MANUFACTURING MODELS

In this section, we consider syntheses and generalizations of the simple dynamic lot size model considered in Section 2. The problems addressed by the generalized models remain ones in which items are intermittently produced with associated set-up costs, and stored in inventory with associated holding costs until they are needed. The generalized models, however, take into account production capacities to be shared by the individual items. The models also address plans for coordinating the timing and sizing of production runs of items produced at each of several manufacturing stages. The models discussed in this section are applicable to production planning and scheduling problems in the automobile, aircraft, computer and electronics industries, to name only a few. An extensive review of discrete parts manufacturing models can be found in Bahl, Ritzman and Gupta (1987).

The model we will consider in section 5.1 extends the scope of decision making from the timing and sizing of production runs for a single item to that of many items sharing production capacity. A further generalization considered in section 5.2 leads to models for determining the timing and sizing of runs for items in a multi-stage production environment. Finally, in section 5.3, we consider a model possessing a hierarchical structure in which families of items can be grouped together into types for aggregate planning purposes. As we shall demonstrate, decomposition methods are useful for exploiting the structure of individual sub-models that make up the various large-scale models.

5.1 MULTI-ITEM SINGLE-STAGE MODELS

Manne (1958) was the first to propose a model for this class of problems. The model and the optimization approach was further developed by Dzielinski and Gomory (1965), and specialized and applied to tire manufacture by Lasdon and Terjung (1971). For expositional convenience, the version we present here assumes that there is only one type of capacity (e.g., machine hours, labor hours) to be shared in producing items in each period.

Indices:

i: items $(i = 1, 2, \ldots., I)$

j: trial production strategies

J_i: index set of trial production strategies for item i

t: time periods $(t = 1, 2, \ldots., T)$

Parameters:

c_i = set-up cost for item i ($)

a_i = set-up resource utilization for item i (hrs)

b_i = resource utilization rate for item i (hrs/unit)

h_i = unit holding cost for item i ($/unit)

y_{i0} = initial inventory of item i (units)

r_{it} = demand for item i in period t (units)

q_t = shared resource availability in period t (hrs)

M_{it} = upper bound on production of item i in period t

Variables:

x_{it} = quantity of item i produced in period t

y_{it} = quantity of item i held in inventory at the end of period t

$$\delta_{it} = \begin{cases} 1 & \text{if } x_{it} > 0 \\ 0 & \text{otherwise} \end{cases}$$

Implicit in the definition of this problem is the fact that all periods are of the same length (eg, one week). The model will assume that demand must be met in each period from inventory or production; no backlogging is allowed. Variable production costs have been ignored since they can be assumed to be the same in every period and therefore represent an unavoidable cost. Finally, note that we have included both a set-up cost and a set-up resource u‵.lization for each item. The set-up cost might equal the cost of unusable production at the beginning of a run. The set-up resource utilization might be lost time in adjusting a machine for a new item to be produced.

<u>Multi-item, Single Stage Discrete Parts Manufacturing Model</u>

$$v = \min \sum_{i=1}^{I} \{c_i \sum_{t=1}^{T} \delta_{it} + h_i \sum_{t=1}^{T} y_{it}\} \qquad (5.1.1)$$

Subject to

$$\sum_{i=1}^{I} a_i \delta_{it} + b_i x_{it} \le q_t \qquad \text{for } t=1,2,..,T \qquad (5.1.2)$$

For i=1,2,...,I

$$y_{it} = y_{i,t-1} + x_{it} - r_{it} \qquad (5.1.3)$$

for t=1,2,...,T

$$x_{it} - M_{it} \delta_{it} \le 0 \qquad (5.1.4)$$

$$x_{it} \ge 0, \quad y_{it} \ge 0, \quad \delta_i t = 0 \text{ or } 1 \qquad (5.1.5)$$

This is an MIP model of potentially very large size. It has (2I+1)T constraints and 3IT variables; thus, if I=500 and T=10, we have a model with 10,010 constraints and 15000 variables. Of course, the constraints (5.1.3) and (5.1.4) are quite simple, and for each I constitute an algebraic representation of the single item dynamic lot-size model discussed in section 2. For future reference, we define the set

$N_i = \{(x_{it}, y_{it}, \delta_i t)$ that satisfy (5.1.3), (5.1.4), (5.1.5)$\}$

A price-directed decomposition approach for this model is to dualize on the capacity constraints (5.1.2), thereby allowing the individual items to be optimized separately. Following the development in section 3, for any $\pi = (\pi_1, \pi_2, ..., \pi_T) \ge 0$, we construct the Lagrangean

$$L(\pi) = - \sum_{t=1}^{T} q_t \pi_t + \min \sum_{i=1}^{I} \sum_{t=1}^{T} \{(c_i + a_i \pi_t) \delta_{it}$$

$$+ (b_i \pi_{it}) x_{it} + h_i y_{it}\}$$

$$\text{Subject to} \quad (x_{it}, y_{it}, \delta_i t) \in N_i \quad \text{for all } i \qquad (5.1.6)$$

This function can be re-written as

$$L(\pi) = -\sum_{t=1}^{T} q_t \pi_t + \min \sum_{i=1}^{I} F_i(\pi)$$

where

$$F_i(\pi) = \sum_{t=1}^{T} \{(c_i + a_i \pi_t)\, \delta_{it} + (b_i \pi_{it})\, x_{it} + h_i y_{it}\} \qquad (5.1.7)$$

$$\text{Subject to} \quad (x_{it}, y_{it}, \delta_i t) \in N_i$$

In words, each $F_i(\pi)$ is a dynamic lot size model defined over the set N_i of exactly the form discussed in section 2.1. To reflect the price π_t being charged for shared resource in period t, the set-up costs associated with the δ_{it} have been expanded to $c_i + a_i \pi_t$. In addition, a time-dependent variable production cost term $b_i \pi_t$ has been added. Thus, if $\pi_3 > 0$ for the example depicted in 2.2, arcs connecting beginning inventory in period 3 to ending inventory in period 3 associated with setting up and producing the item would be "longer" (more costly), and the dynamic programming recursion would seek a shorter path.

Based on our development in section 3, We can state the three main properties of the Lagrangean construction.

Property 1: (Optimality Test) The solution $(x_{it}(\pi), y_{it}(\pi), \delta_i t(\pi))$ that is optimal in the Lagrangean $L(\pi)$ is optimal in the Multi-Item Model (5.1.1) - (5.1.5) if

$$\sum_{i=1}^{I} a_i \delta_{it}(\pi) + b_i x_{it}(\pi) \le q_t \qquad \text{for } t=1,2,..,T$$

with equality on constraint t if $\pi_t > 0$.

Property 2: For any $\pi \ge 0$, $L(\pi) \le v$.

Property 3: If $\pi \geq 0$ produces an optimal solution to the Model according to the optimality conditions of Property 1, then π is optimal in the dual problem

$$d = \max L(\pi)$$
$$\text{Subject to } \pi \geq 0 \qquad (5.1.8)$$

Moreover, if the optimality conditions hold, we have $v = d$.

There is no guarantee, however, that the optimality conditions can be made to hold for any dual vector $\pi \geq 0$. This is because the Multi-Item Model is a mixed integer program, a class of non-convex models for which duality gaps ($v - d > 0$) are common. To ensure convergence to an optimal solution to the Model, we would need to imbed the dual methods in a branch-and-bound procedure (see Shapiro (1979a, 1979b)). As we shall see below, for this particular model, it is possible to characterize the difficulty due to the model's non-convexity in applying Lagrangean methods.

The price-directed generalized linear programming approach discussed in section 3 can be specialized for this application. In particular, suppose for each i that we have the trial production strategies $(x_{itr}, y_{itr}, \delta_{itr}) \in N_i$ for $r = 1, 2, \ldots, R_i$. For $t = 1, 2, \ldots, T$, let

$$\beta_{itr} = a_i \delta_{itr} + b_i x_{itr}$$

and let

$$C_{ir} = c_i \sum_{t=1}^{T} \delta_{itr} + h_i \sum_{t=1}^{T} y_{itr}$$

Given this data, we construct the master model

$$\min \sum_{i=1}^{I} \sum_{r=1}^{R_i} C_{ir}\, \Omega_{ir} \tag{5.1.9}$$

Subject to

$$\sum_{i=1}^{I} \sum_{r=1}^{R_i} \beta_{itr}\, \Omega_{ir} \le q_t \qquad \text{for } t=1,2,..,T \tag{5.1.10}$$

$$\sum_{r=1}^{R_i} \Omega_{ir} = 1 \qquad \text{for } i=1,2,...,I \tag{5.1.11}$$

$$\Omega_{ir} \ge 0 \tag{5.1.12}$$

In words, this linear program is intended to select a schedule r for each item from among the trial schedules generated so far so as to minimize total cost without exceeding available capacity. Of course, there is no guarantee that a pure (unique) strategy will be selected for each item since more than one Ω_{ir} may take on fractional values between zero and one. This is a serious deficiency of the approach. However, with the reader's indulgence, we delay addressing it until after we have discussed how the optimal LP dual variables for the master model can be employed in generating effective new production strategies.

Let π_t denote the shadow prices on the resource constraints (5.1.10) and let Φ_i denote the shadow prices on the convexity rows (5.1.11). For each i, we use the π_t in solving $F_i(\pi)$ given in (5.1.8). This is a single-item dynamic programming calculation of the type discussed in section 2.2. If $F_i(\pi) < \Phi_i$, we add a column to the master model corresponding to the production strategy with this value. If $F_i(\pi) = \Phi_i$, we do not add a column to the master model. If no new columns can be added to the master model because the latter condition obtains for all i, then the generalized linear programming method terminates. We refer to such a master model as an optimal master model. All earlier master models (that is, those admitting new columns) are called non-optimal. Termination or convergence to an optimal master model must occur in a finite number of steps because there are only a finite number of feasible solutions to each of the $F_i(.)$ models.

We return now to the central issue in applying generalized linear programming to a structured mixed integer programming model such as the Multi-Item Model: Namely, how to interpret solutions from an optimal or non-optimal master model. In general, we cannot expect that applying generalized linear programming as just described will lead to an optimal solution to the Multi-Item Model. It is instructive to investigate exactly why this is the case. Let Ω_{ir} denote optimal values to any optimal or non-optimal master model. We say that the master model produces a <u>pure strategy</u> for item i if $\Omega ip = 1$ for some p and $\Omega_{ir} = 0$ for all $r = p$. Conversely, we say that the master model produces a <u>mixed strategy</u> for item i if $\Omega_{ir} > 0$ for two or more strategies r; this implies $\Omega ir < 1$ for these strategies. Obviously, when the master model suggests a mixed strategy for item i, we are in a quandary about which strategy to employ.

The difficulty is that each time we solve the master model, we can expect mixed strategies to occur for a certain number of items in our Multi-item Model. If this number is small, say less than 5% of the items, the difficulty may not be not be serious. For items with mixed strategies, we may choose the strategy that was assigned the highest weight Ω_{ir}. The resulting infeasibility or lack of optimality will probably be negligible relative to the accuracy of the demand data.

We can calculate a bound on the maximal number of items that will have mixed strategies in the master model based on properties of the simplex method. In particular, assuming $I > T$, a simple counting argument establishes that the number of mixed strategies cannot exceed T. Thus, if $I = 500$ and $T = 10$, the number of items with mixed strategies cannot exceed 10 or 2%. This is an extreme case, however, because we assumed only one resource to be shared in each period. For example, if $I=500$, $T = 20$, and the number of shared resources is each period is 10 (different types of machines, labor categories, raw materials, etc.), the potentially maximal number of items with shared resources equals 200, or 40% of the total number of items. Although the actual percentage of mixed strategies would probably be much lower, the percentage could easily be difficult to handle in the heuristic manner just outlined.

A subtle but important point regarding the validity of the Dynamic Lot-Size Theorem for a single item discussed in section 2.1 in the context of the Multi-Item Model (5.1.1) - (5.1.5) and the model decomposition is the following. Due to the capacity constraints (5.1.2), it may not be optimal or even feasible to limit production of any given item to those periods when inventories have fallen to zero. However, by the Lagrangean decomposition (5.1.7), the iteraction between items due to constraints (5.1.2) have been eliminated, and the Dynamic Lot-Size Theorem is valid for computing the $F_i(\pi)$ for all i. In this sense, the dual decomposition is deficient because it fails to provide structure for generating necessary shortest route paths in the dynamic programming network, and therefore columns for the master model (5.1.9) - (5.1.12), that violate the Theorem. This deficiency is an unavoidable result of the discrete, non-convex nature of the scheduling sub-models.

Lasdon and Terjung (1971) applied a version of the Multi-Item Model to the production of automobile tires. Recently, Eppen and Martin (1987) have developed variable redefinition procedures for this class of models that produce tighter linear programming and Lagrangean relaxations. They report on successful experiments with models consisting of up to 200 items and 10 time periods.

5.2 MULTIPLE ITEM, MULTIPLE STAGE MODELS

The models we shall discuss in this subsection are mixed integer programming generalizations of the single stage models discussed in the previous subsection. They are applicable to a wide range of discrete parts manufacturing problems although, thus far, they have found little application to real world problems. The lack of application is due in part to the size and complexity of the models which make them difficult to construct and optimize. This technical difficulty should gradually disappear as computer technology continues to improve.

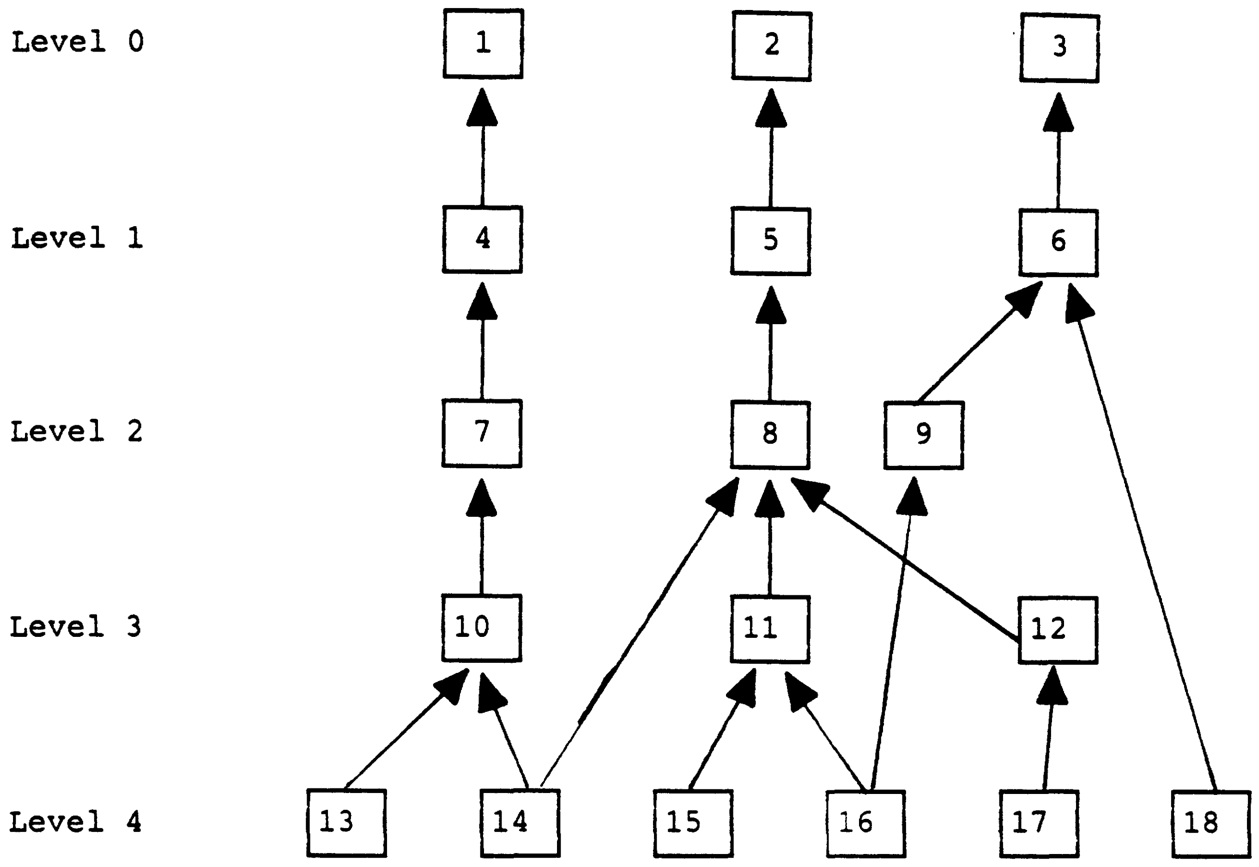
In addition, the practical use of these production models should benefit from recent research into stronger mixed integer programming model formulations and more efficient mixed integer programming algorithms. Crowder, Johnson and Padberg (1983) report on successful computational experience with cutting plane and other procedures for achieving tighter formulations of pure integer programs. Van Roy and Wolsey (1986, 1987) develop valid inequalities and reformulations for mixed integer programs. Martin (1987) develops a theory of variable redefinition for mixed integer programs that creates or exposes special structure which can be exploited by tailored solution algorithms. These general integer and mixed integer programming procedures have direct application to the types of models discussed here; for example, see Barany, Van Roy and Wolsey (1984), or Eppen and Martin (1987).

However, the barrier to use of these models is more than a mathematical or technical one. Materials Requirements Planning systems have found widespread use in industry during the past ten years. The models we discuss must be conceptually and practically integrated with MRP if they are to have a significant future.

At the moment, the outlook for new modeling applications is promising because many manufacturing managers have come to realize the limitations of their MRP systems. These systems are, in fact, no more than informational programs that provide detailed bill of materials from pre-determined master schedules. Moreover, although some MRP systems will calculate capacity loadings that result from a detailed schedule, the systems make no attempt determine an effective master schedule by allocating capacity resources in an optimal or efficient manner. The models discussed here are intended to fill this planning vacuum. But, the vast majority of manufacturing managers need to be educated about their practical relevance.

At the core of any MRP system, and of mathematical programming models for discrete parts manufacturing problems, is the product structure. A typical product structure is depicted in Figure 5.1. Each box refers to an item to be produced and the number in the box is the index of the item. A chain links each item to all other items that require it. For example, item 17 is linked to items 12, 8, 5, 2, and 3. Each item is associated with exactly one level in the product structure; this level equals the length of the longest chain beginning at the item. The levels are indexed 0 to L - 1 where items at level 0 are called finished goods and items at Level 1 - 1 are called raw materials.

With this background, we can state a mathematical programming model for optimizing over general product structures. The model statement follows Billington, McLain and Thomas (1983); see also Chapman (1985).



Example of a Product Structure

Figure 5.1

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	$\equiv$	inventory of holding cost ($ per unit of item i)
cs_i	$\equiv$	setup cost ($ per setup of item i)
co_{kt}	$\equiv$	overtime cost ($ per unit of capacity in period t at facility k
L_i	$\equiv$	minimum lead time for item i
f_i	$\equiv$	yield of item i (fraction)
a_{ij}	$\equiv$	number of units of item i required for the production of one unit of item j
r_{it}	$\equiv$	demand for item i in period t
b_{ik}	$\equiv$	capacity utilization rate of item i at facility k (capacity units per unit)
s_{ik}	$\equiv$	setup utilization of facility capacity k by item i (capacity units)
CAP_{kt}	$\equiv$	capacity of facility k at time t (units of capacity)
q_{it}	$\equiv$	upper bound on the production of item i that can be initialized in period t

Variables:

y_{it}	$\equiv$	inventory of item i at the end of period t
δ_{it}	$\equiv$	1 if item i made in period t, 0 otherwise
O_{kt}	$\equiv$	overtime capacity at facility k in period t
x_{it}	$\equiv$	production of item i initiated in period t

<u>Multi-Stage, Multi=Item Discrete Parts Manufacturing Model</u>

$$v = \text{minimize} \sum_{i=1}^{N} \sum_{t=1}^{T} (h_i y_{it} + cs_i \delta_{it}) + \sum_{k=1}^{K} \sum_{t=1}^{T} (co_{kt} O_{kt}) \qquad (5.2.1)$$

$$\text{s.t. } y_{i,t-1} + f_i x_{i,t-Li} - y_{it} - \sum_{j=1}^{N} a_{ij} x_{jt} = r_{it} \qquad \begin{matrix} i=1,\dots,N \\ t=1,\dots,T \end{matrix} \qquad (5.2.2)$$

$$\sum_{i=1}^{N} (b_{ik} x_{it} + s_{ik} \delta_{it}) - O_{kt} \le CAP_{kt} \qquad \begin{matrix} k=1,\dots,K \\ t=1,\dots,T \end{matrix} \qquad (5.2.3)$$

$$x_{it} - q_{it} \delta_{it} \le 0 \qquad \begin{matrix} i=1,\dots,N \\ t=1,\dots,T \end{matrix} \qquad (5.2.4)$$

$$\delta_{it} = 0 \text{ or } 1,\ x_{it} \ge 0,\ y_{it} \ge 0,\ O_{kt} \ge 0 \qquad (5.2.5)$$

The objective function (5.2.1) represents avoidable inventory, set-up, and overtime costs that we seek to minimize. The constraints (5.2.2) are generalized inventory balance constraints. Ending inventory of each item i in each period t equals starting inventory plus net production (recall f_i is the yield factor) of the item in period t - L_i (recall L_i is the lead time) minus internal and external demand for the item in that period. Here the lead time L_i should equal the minimal lead time that is required to produce or otherwise acquire the item. Capacity restrictions and set-up constraints are given in (5.2.3) and (5.2.4).

The Multi-Item and Multi-Stage Model can easily attain an enormous size; if $N = 1000$, $T = 13$, and $K = 50$, the model would have 26650 constraints, 52650 continuous variables and 26000 zero-one variables. We will discuss two distinct decomposition approaches for breaking up the Model into manageable sub-models. One is a nested decomposition scheme proposed by Chapman (1985) that we will discuss in the paragraphs that follow. The other is a hierarchical planning approach that will be discussed in the next sub-section. Although both approaches require further basic research and computational experimentation, they offer considerable promise as effective means for dealing with the size and complexity of monolithic mixed integer programming models to support MRP.

The nested decomposition approach is based on two observations about the Model. First, if the set-up variables δ_{it} are fixed at zero-one values, the residual sub-model is a linear program. This is the starting point for Benders' decomposition method. The second observation, which we discuss in detail below, is that the generalized inventory balance equations (5.2.2) possess dynamic Leontief structures. It is easier to optimize over such structures than over ordinary linear programming equations. Thus, if we were to apply Lagrangean relaxation on the capacity constraints (5.2.3) in the linear programming sub-model, the special structures could be exploited. The resulting decomposition scheme is depicted in Figure 5.2.

A Leontief substitution model is a linear program

$$\begin{aligned} \min \quad & cx \\ \text{s.t.} \quad & Ax = b \\ & x \geq 0 \end{aligned}$$

where $b \geq 0$ and A is an mxn matrix ($n \geq m$) of rank m with the property: Any mxm invertible sub-matrix B is of the form $B = I - Q$ where the non-negative matrix Q has the property that Q^n goes to zero as n goes to infinity. This property implies that any basis B has an inverse B^{-1} consisting of non-negative elements. This in turn implies that an optimal basis B for any $b \geq 0$ is optimal for all $b \geq 0$ since $B^{-1}b \geq 0$.

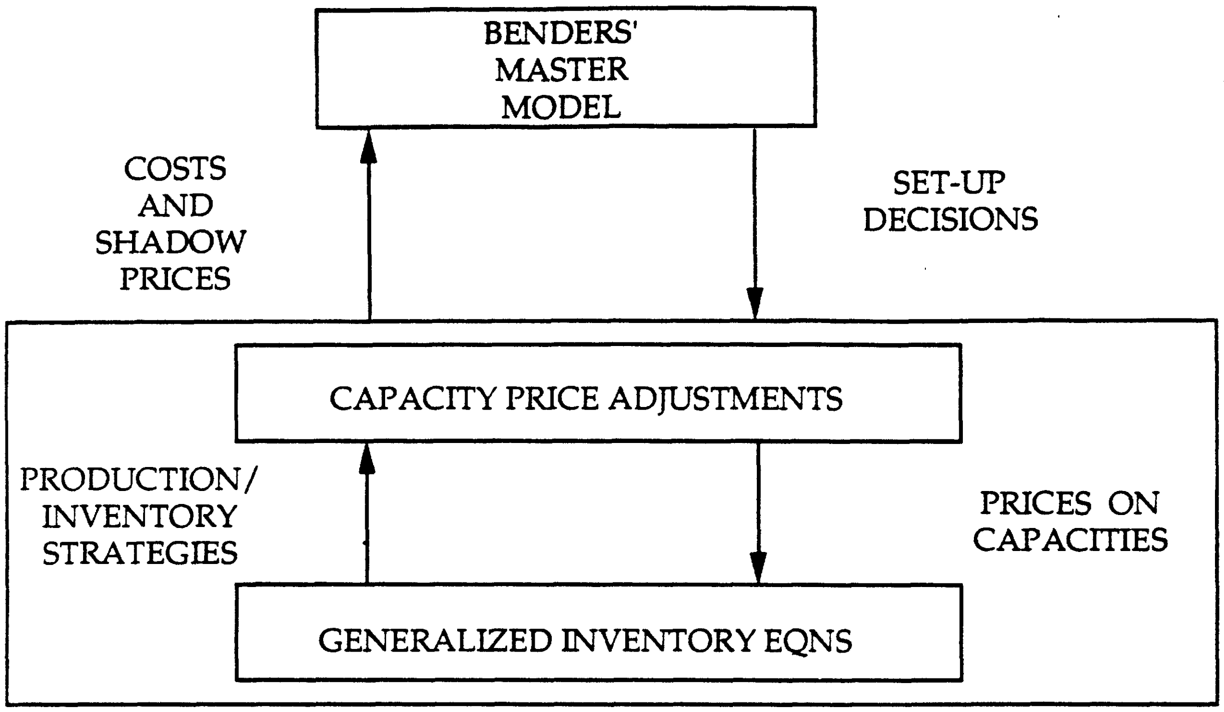
A dynamic generalization is

$$\min \sum_{t=1}^{T} (c_t x_t)$$

$$\text{subject to} \quad A_t x_t = b_t + R_{t-1}\, x_{t-1} \qquad \text{for } t=1,\ldots,T$$

$$x_t \geq 0 \quad (x_0 \text{ given})$$

where A_t is a Leontief substitution matrix and R_t is non-negative. It can be shown that this multi-period linear


BENDERS' MASTER MODEL
COSTS AND SHADOW PRICES
SET-UP DECISIONS
CAPACITY PRICE ADJUSTMENTS
PRODUCTION/ INVENTORY STRATEGIES
PRICES ON CAPACITIES
GENERALIZED INVENTORY EQNS

Nested Decomposition Scheme for the Multi-Stage Model

Figure 5.2

programming model can be optimized by computing an optimal basis for each A_t and then computing $x_t = (x_{Bt}, x_{Nt})$ for each t by

$$x_{Bt} = B^{-1} (b_t + R_{t-1} x_{t-1})$$

$$x_{Nt} = 0$$

Classic references to dynamic Leontief substitution models and their application to production models are Dantzig (1955) and Vernott (1969).

Now the generalized production/inventory balance equations (5.2.2) can be viewed as a dynamic Leontief system with substitution, except possibly for transient conditions due to initial inventories and initial production decisions with positive lead times. Looking at (5.2.2), we can set that net demands for item i

$$r_{i1} - y_{i,0} - x_{i,1-Li}$$

and

$$r_{it} - x_{i,t-Li} \qquad \text{for } t=2,\ldots,L_i$$

may be negative, thereby destroying the direct application of the Leontief substitution properties. Indeed, we would expect and hope that net demand for some items are negative in early periods, thereby relieving us of the need to produce items in those periods.

Chapman (1985) gives a procedure for identifying the transients at the start of the planning horizon. Moreover, the residual Leontief substitution systems in this case have a simple structure allowing a solution to be computed without matrix inversion. The algorithm for accomplishing this scans each column of the constraint set exactly once to determine a primal optimal basis and corresponding shadow prices. Calamaro (1985) reports on computational experience with this algorithm and the nested decomposition scheme of Figure 5.2.

The Multi-Item and Multi-Stage Model allows general product structures and the imposition of capacity constraints. A number of authors have developed algorithmic approaches to restricted, easier versions of the Model. Assembly systems have product structures for which each item in the product structure has at most one successor. For uncapacitated assembly systems, Afentakis et al (1984) developed a Lagrangean relaxation approach that allows production of each item in the product structure to be scheduled separately by a dynamic programming algorithm. The efficiency of their model construction and the decomposition relies upon variables, constraints and costs associated with echelon stock which is defined to be the number of units of an item held explicitly in inventory and implicitly in inventory as parts of its successors. Afentakis and Gavish (1986) have extended this approach by showing how a general product structure can be transformed to an equivalent assembly structure.

Aftentakis et al (1984) and Aftentakis and Gavish (1986) employed subgradient optimization of the Lagrangean relaxations which were imbedded in tailored branch and bound approaches for globally optimizing the multi-stage models. They also used forward heuristics for generating good solutions. As a result, they were able to report efficient computational result for general multi-stage problems with up to 40 items in which the upper bounds (from the heuristics) and the lower bounds (from the Lagrangean relaxation) are within a few small percentage points.

5.3 HIERARCHICAL PLANNING

We saw in the previous two sub-sections how large scale, monolithic mixed integer programming models for discrete parts manufacturing could be usefully decomposed. In the process, we were able to identify and exploit special structures contained within these monolithic models. Even with the decompositions, however, the models remained monolithic in terms of the treatment of individual items. Specifically, all items in the discrete parts manufacturing models presented above were treated as separate and equal entities to be produced in batches and stored in inventory. No recognition of the possibility for aggregating items into larger units for planning or scheduling purposes was made.

Our objective in this section is to discuss how mathematical programming models and decomposition methods can be extended to capture and exploit natural aggregations that occur in many manufacturing environments. For ease of exposition in developing and analyzing the hierarchical model, we will consider a single stage manufacturing system. Our discussion follows closely the development in Graves (1982).

The qualitative basis for considering item aggregation is the concept of <u>hierarchical planning</u> (Hax and Meal (1975)).

Using this concept for the single stage system, items may be aggregated into families, and families into types. A type is a collection of items with closely similar seasonal demand patterns and production rates. A family is a set of items within a type such that the items in the family share a common setup. In the mixed integer programming model that we develop below, we consider that the planning function of the model is to determine time dependent resource requirements to satisfy aggregate demand for types over a tactical planning horizon. We consider that the scheduling function of the model is to determine how to allocate these resources over a shorter, scheduling horizon.

The hierarchical approach has three advantages over the discrete parts models considered in the previous sub-sections. First, the aggregation serves to further reduce the dimensionality of the models, both as monolithic and decomposed mixed integer programs. Second, the hierarchical model requires less detailed demand data since demands need only be forecasted for the planning horizon (see Axsater (1981) for further discussion of aggregation methods in production planning). The third advantage is that the sub-models of the decomposed hierarchical model will, or at least should, correspond to the organizational and decision-making echelons of the manufacturing firm.

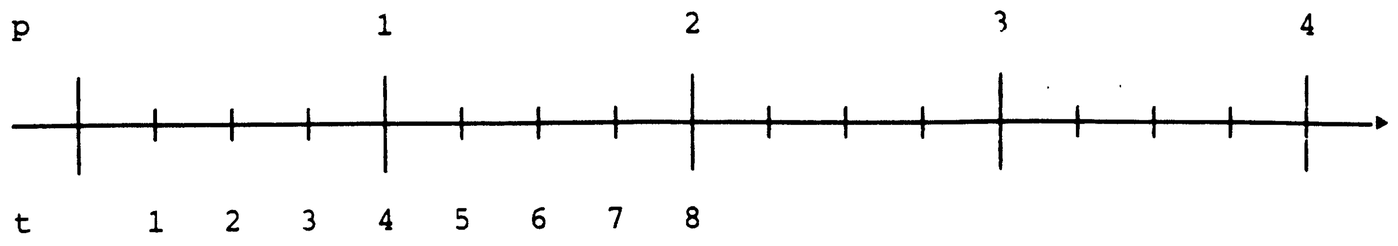
With this background, we can present a mathematical statement of the model. We employ the notation used in the Multi-Item Multi-Stage Model presented in the previous section.

The main differences are in the definitions of the entities to be produced and the treatment of time. In particular, these are changes in the definitions and meanings of the index sets.

Indices:

i	=	$1,\ldots,I$	index of types
j	=	$1,\ldots,N$	index of families
J_i	=	set of families belonging to type i	
P	=	length of planning horizon	
p	=	index of periods in planning horizon	
T	=	length of scheduling horizon (measured in planning horizon period units)	
t	=	index of periods in scheduling horizon	
k	=	number of scheduling periods in a planning period	

Figure 5.3 illustrates how we differentiate the length and period definitions for the planning horizon and the scheduling horizon. For P = 4@4-week months, T = 2@4-week months, and k = 4 weeks in a month, the scheduling periods are indexed by $t = 1,\ldots,kT=8$, and $p = 1, 2, 3, 4$.



Scheduling and Planning Horizons
Figure 5.3

Hierarchical Planning Model

$$v = \text{minimize} \quad \sum_{p=1}^{P} (co_p O_p + \sum_{i=1}^{I} h_i y_{ip}) + \sum_{j=1}^{N} \sum_{t=1}^{kT} (cs_j \delta_{jt}) \qquad (5.3.1)$$

$$\text{s.t.} \quad y_{i,p-1} + x_{ip} - y_{ip} = r_{ip} \qquad i=1,\ldots,I \quad p=1,\ldots,P \qquad (5.3.2)$$

$$\sum_{i=1}^{I} b_i x_{ip} - O_p \le CAP_p \qquad p=1,\ldots,P \qquad (5.3.3)$$

$$\sum_{j\varepsilon J_i} \sum_{t=k(p-1)+1}^{kp} y_{jt} - y_{ip} = 0 \qquad i=1,\ldots,I \quad p=1,\ldots,T \qquad (5.3.4)$$

$$y_{j,t-1} + x_{jt} - y_{jt} = r_{jt} \qquad j=1,\ldots,I \quad t=1,\ldots,kT \qquad (5.3.5)$$

$$x_{jt} - q_{jt}\delta_{jt} \le 0 \qquad j=1,\ldots,N \quad t=1,\ldots,kT \qquad (5.3.6)$$

$$x_{ip} \ge 0, \quad y_{ip} \ge 0, \quad O_p \ge 0 \qquad (5.3.7)$$

$$\delta_{jt} = 0 \text{ or } 1, \; x_{jt} \ge 0, \; y_{jt} \ge 0 \qquad (5.3.8)$$

The objective function (5.3.1) in this formulation consist of two terms. The first corresponds to avoidable capacity and inventory holding costs associated with the aggregate planning problem over the longer planning horizon consisting of P periods. The second corresponds to avoidable set-up costs associated with the detailed scheduling problem over the shorter scheduling horizon consisting of T periods. As we discussed, the time periods in the time summation for the scheduling problem are a refinement of the time periods used in the planning problem.

The constraints (5.3.2) and (5.3.3) describe inventory balance and capacity decisions for the planning problem. The constraints (5.3.4) link aggregate inventories of types to more detailed inventories of families in each type category. Again, the time summation reflects the finer time period definition used for the scheduling problem. Constraints (5.3.5) and (5.3.6) describe inventory balance and set-up decisions for the scheduling problem. Demands for types and demands for families are implicitly linked by the relation

$$\sum_{j \varepsilon J_i} \sum_{t=(k-1)p+1}^{kp} r_{jt} = r_{ip} \qquad \text{for } p = 1,2,\ldots,T$$

In words, this relation says that the sum of demands for families j in type i used in (5.3.5) during the scheduling periods falling in the pth planning period equals the aggregate demand used in (5.3.2).

The Hierarchical Planning Model can be effectively decomposed by pricing out the linking constraints (5.3.4). Letting π_{ip} denote the dual variable on each of these constraints, and π the IT-vector with components π_{ip}, the resulting Lagrangean separates as follows

$$L(\pi) = A(\pi) + \sum_{i=1}^{I} \sum_{j \varepsilon J_i} F_j(\pi)$$

where $A(\pi)$ is a linear programming model for optimizing the planning problem

$$A(\pi) = \min \sum_{p=1}^{P} \{co_p O_p + \sum_{i=1}^{I} (h_i - \pi_{ip}) y_{ip}\} \qquad (5.3.9)$$

Subject to (5.3.2), (5.3.3), (5.3.7)

and the $F_j(\pi)$ for each family item j is a single item dynamic lot-size model of the type discussed in section 2.2

$$F_j(\pi) = \min \sum_{p=1}^{T} \sum_{t=(k-1)+1}^{kp} (\pi_{ip} y_{jt} + cs_j \delta_{jt}) \qquad (5.3.10)$$

Subject to (5.3.5), (5.3.6), (5.3.8)

Note that, in the dynamic-lot size sub-model (3.10), the decomposition has induced holding prices π_{ip} lot-size on inventories of family items in the aggregate period p. Unlike previous instances of this model, however, these holding prices might be negative as well as positive. A negative holding price on type i items in planning period p would indicate that, from a planning viewpoint, scheduling production to create inventories of families in the stated type in the stated planning period has global benefits.

The same algorithmic procedures discussed in section 3, and in section 5.1 above, for determining the dual variables π_{ip} are available for this decomposition. Since the dynamic lot-size sub-models involve integer variables, the dual decomposition method will suffer from duality gaps between the Hierarchical Planning Model and its dual model. Graves (1982) discusses methods for overcoming duality gaps and choosing dual variables, and reports on computational experience with test models.

6. JOB-SHOP SCHEDULING

Job-shop scheduling problems involve jobs consisting of a variety of machine operations to be processing on a number of machines in a random or arbitrary pattern. For example, these might be finishing operations on paper produced at a paper mill or castings produced at a foundry, or, maintenance and repair operations on jet engines. As we shall see, the complexity of job-shop scheduling leads to large and difficult combinatorial optimization models. Thus, the practical use of models and associated analytical and heuristic methods should be to identify demonstrably good schedules, rather than to persist in trying to find an optimal solution.

6.1 BASIC MODEL

We consider a combinatorial optimization model proposed by Lagewig, Lenstra and Rinooy Kan (1977) and a number of other authors for a large class of job-shop scheduling problems. This model is comprised of a number of jobs, each consisting of a number of operations to be processed on pre-assigned machines. The objective is to minimize the total length of time required to finish all jobs. In so doing, the model must simultaneously sequence the processing of operations assigned to each machine and ensure that the precedence relations among operations of each job are obeyed. Extensions of the basic model are considered in the following section.

Indices and Index Sets:

i	=	jobs for $i = 1, \ldots, I$
n_i	=	number of operations in job i
j	=	operations for $j = 1, \ldots, N = \sum_{i=1}^{I} n_i$
k	=	machine for $k = 1, \ldots, K$
k_j	=	machine on which operation j is to be performed
J_k	=	$\{ j \mid k_j = k \}$ = operations assigned to machine k
Rk	=	$\mid J_k \mid$ = number of jobs assigned to machine k
r	=	machine sequence order, $r = 1, \ldots, R_k$

The assumption for this class of job-shop scheduling problems is that the operations in a given job are to be processed sequentially; that is, operation $j-1$ must be completed before operation j may begin. Thus, there is a total ordering of the operations of each job. For notational purposes, we assume that the operations of job i are indexed by $j = N_{i-1} + 1, \ldots, N_i$, where

$$N_i = \sum_{g=1}^{i} n_g.$$

Parameters:

p_j	=	processing time for operation j
T	=	upper bound on total processing time

Variables:

t_j = start time for operation j

$$x_{jg} = \begin{cases} 1 \text{ if operation } j \text{ performed before operation } g \\ 0 \text{ if operation } g \text{ performed before operation } j \end{cases} \quad \text{for } j,g\varepsilon J_k$$

F = total processing time for all jobs

<u>Job-Shop Scheduling Model</u>

$$v = \min F \qquad (6.1.1)$$

Subject to

For $i = 1, \ldots, I$

$$t_j \geq t_{j-1} + p_{j-1} \qquad \text{for } j = N_{i-1} + 2, \ldots, N_i \qquad (6.1.2)$$

$$F \geq t_{N_i} + p_{N_i} \qquad (6.1.3)$$

For $k = 1, \ldots, K$

$$t_g \geq t_j + p_j x_{jg} - T(1 - x_{jg})$$

$$\text{for all } j,g\varepsilon J_k \qquad (6.1.4)$$

$$t_j \geq t_g + p_g(1 - x_{jg}) - Tx_{jg}$$

$$x_{jg} = 0 \text{ or } 1 \text{ for all } j,g\varepsilon K_j \qquad (6.1.5)$$

$$F \geq 0, \quad t_j \geq 0 \quad \text{for } j = 1, \ldots, N \qquad (6.1.6)$$

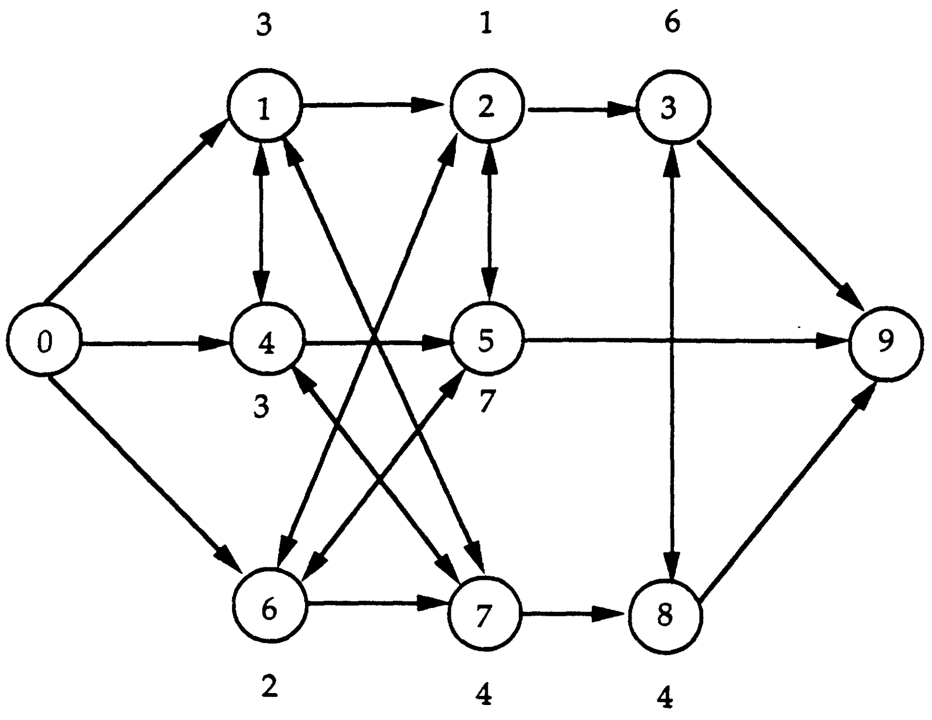
The objective function (6.1.1) is to minimize the time to complete all jobs. The variable time F is, by (6.1.3), equal to the maximum of the completion times of the I jobs. The constraints (6.1.2) state that the start times of operation j for j=2 through $j=n_i$ for each job i must occur after operation j-1 has been completed. The start time for operation 1 is not constrained by this total precedence ordering.

The start times t_j of operations are also constrained by the sequence of operations to be performed on each machine k. Specifically, for each pair of operations j,g assigned to machine k, the constraints (6.1.4) state that either operation j will precede operation g, or that operation g will precede operation j. For example, if $x_{jg} = 1$, then the first of the two constraints is binding ($t_g \geq t_j + p_j$), whereas if $x_{jg} = 0$, then the second of the two constraints is binding ($t_j \geq t_g + p_g$).

To sum up, the start time t_j of operation j is constrained both by the total precedence ordering on the operations of its job, and by the sequencing order of operations from a variety of jobs on the machine k_j on which j is processed. The former constraints (6.1.2) and (6.1.3) are the simple constraints of a network optimization type found in CPM models (see Schrage (1986)). The latter constraints (6.1.4) are logical constraints of a type referred in the literature as disjunctive (Roy and Sussmann (1964), Balas (1979)).

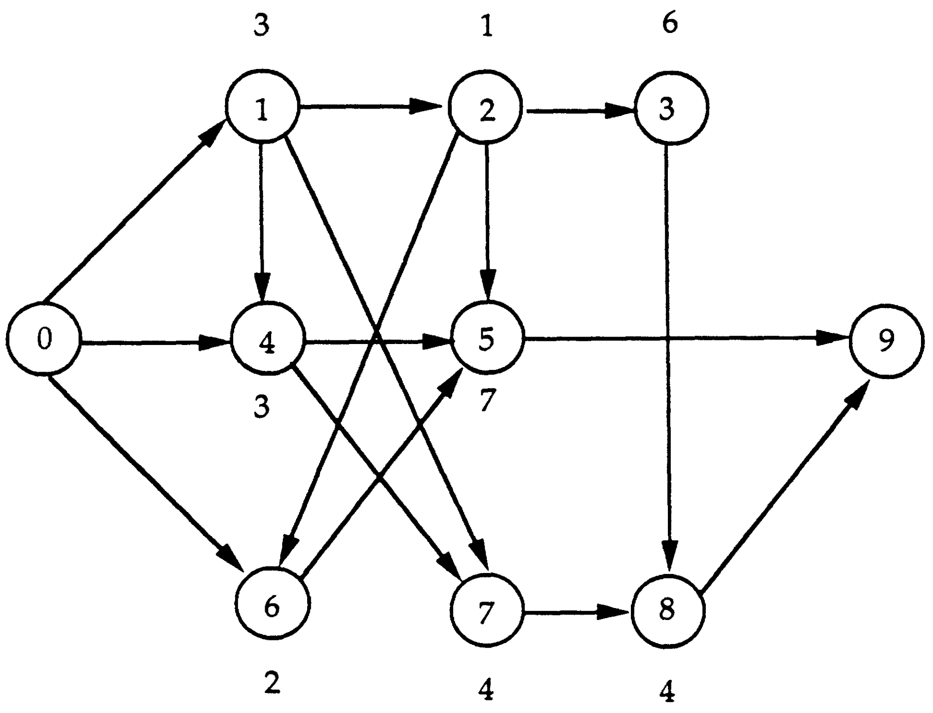
The result is a mixed integer programming model of great size and complexity. If $I = 20$, $n_i = 10$ for all i, $K = 8$, and $R_k = 25$ for all k, the Model would have 2580 constraints, 201 continuous variables, and 2400 zero-one variables. To try to circumvent extreme computational difficulties, several researchers have proposed solutions techniques based on combinations of branch-and-bound schemes, heuristics and lower bounds based on easy to optimize relaxations of the Job Scheduling Model (see Lagewig, Lenstra and Rinooy Kan (1977), Adams, Balas and Zawack (1988)). A central construct is the disjunctive graph which consists of a node for each operation j with an associated weight p_j, plus a dummy node 0 for the start of all operations, and a dummy node N+1 for the completion of all operations. For each consecutive pair of operations j - 1 and j of the same job, there is a directed conjunctive arc. For each pair of operations assigned to the same machine, there is a (undirected) disjunctive edge. The major task in optimizing a given job-shop scheduling problem is to pick a direction for each edge thereby determining an order for the pair of operations. If all the edges have been ordered, and an acyclic network results, the time associated with the implied schedule is computed by finding the longest path, based on the node weights, in the directed network. If the resulting network is cyclic, the implied schedule is infeasible.

Figure 6.1 taken from Lagewig, Lenstra and Rinooy Kan (1977) depicts a 3 job, 8 operation, disjunctive graph. The jobs are: 1) 1,2,3; 2) 4,5; 3) 6,7,8. The machine assignments are: machine 1) 1,4,7; machine 2) 2,5,6; machine 3) 3,8. The numbers next to the nodes are the processing times of the jobs. Figure 6.2 is an acyclic network representing a feasible (and optimal) schedule. The longest path in 6.2 is 0, 1, 4, 7, 8, 9 with time equal to 14.



Job-Shop Scheduling as a Disjunctive Graph

Figure 6.1



Optimal Acyclic Graph for Job-Shop Scheduling

Figure 6.2

In a typical branch-and-bound scheme, a subproblem would be characterized by a subset of disjunctive arcs that have been resolved with respect to order (direction); the directed network including these arcs and all the conjunctive arcs must be acyclic. The method proceeds to try to fathom this subproblem (that is, find its best completion, or establish that all completions will have higher objective function value than an optimal solution) by optimizing easy to solve relaxations of the residual job-shop scheduling problem, thereby determining lower bounds. If lower bounding fails to fathom a subproblem, two or more new subproblems are created from it by branching on one or more disjunctive arcs. Knowledge about scheduling conflicts and other problem specific information is used in making the branching decisions. These analytic and experimental approaches are reviewed and extended in Lagewig, Lenstra and Rinooy Kan (1977).

For completeness, we give the mixed integer programming formulation of the job-shop scheduling problem of Figure 6.1. This formulation is

```
MIN      F
SUBJECT TO
       2)    F - T3 >=   6
       3)    F - T5 >=   7
       4)    F - T8 >=   4
       5)    T2 - T1 >=   3
       6)    T3 - T2 >=   1
       7)    T5 - T4 >=   3
       8)    T7 - T6 >=   2
       9)    T8 - T7 >=   4
      10) - 103 X14 - T1 + T4 >= - 100
      11)   103 X14 + T1 - T4 >=   3
      12) - 103 X17 - T1 + T7 >= - 100
      13)   104 X17 + T1 - T7 >=   4
```

```
14) - 103 X47 - T4 + T7 >= - 100
15)   104 X47 + T4 - T7 >=   4
16) - 101 X25 + T5 - T2 >= - 100
17)   107 X25 - T5 + T2 >=   7
18) - 101 X26 - T2 + T6 >= - 100
19)   102 X26 + T2 - T6 >=   2
20) - 107 X56 - T5 + T6 >= - 100
21)   102 X56 + T5 - T6 >=   2
22) - 106 X38 - T3 + T8 >= - 100
23)   104 X38 + T3 - T8 >=   4
```

An optimal solution to this model is

```
      OBJECTIVE FUNCTION VALUE

             14.0000000

  VARIABLE        VALUE
       X14         1.000000
       X17         1.000000
       X47         1.000000
       X25         1.000000
       X26         1.000000
       X56          .000000
       X38         1.000000
         F        14.000000
        T3         4.000000
        T5         6.000000
        T8        10.000000
        T2         3.000000
        T1          .000000
        T4         3.000000
        T7         6.000000
        T6         4.000000
```

6.2 EXTENSIONS TO THE BASIC MODEL

We discuss briefly a number of important extensions to the basic model presented in the previous sub-section. The extensions consider

- o partial ordering of precedences
- o assignment of operations to machines
- o changeover times
- o alternative objective functions

o resource allocation

Partial Ordering of Precedences:

The precedences governing the processing of operations $j = N_{i-1},....,N_i$ of job i may constitute a partial rather than a total ordering as we assumed in developing the basic model. The partial ordering can be described by the relation P where jPg indicates that operation g may begin only after operation j has been completed. The Job Shop Model of the previous sub-section may be easily modified to accommodate the partial order. We simply replace the constraints (6.1.2) by the constraints

$$t_g - t_j \geq p_j \quad \text{for all } jPg \text{ in the partial order}$$

Assignment of Operations to Machines:

We assumed in the basic model that each operation is assigned a priori to a machine. The implication is that only one machine can perform the operation, or possibly that the choice of machine for each operation is obvious. In the more general situation, there would be a set, which we denote by K_j, of machines that can perform the operation. In this case the assignment of operations to machines must be added to the basic model as decision options.

The extension requires that we add constraints and variables. First, we define new variables

$$y_{jk} = \begin{cases} 1 & \text{if operation } j \text{ is assigned to machine } k \\ 0 & \text{otherwise} \end{cases}$$

Then for each operation j for which there is a choice of machine, we add the constraint

$$\sum_{k \varepsilon J_k} y_{jk} = 1$$

In addition, the constraints (6.1.4) for each pair of operations j and g that <u>might</u> (are allowed to) be assigned to the same machine k are required to hold only if $y_{jk} = 1$ <u>and</u> $y_{gk} = 1$. This condition can be enforced by changing (6.1.4) to

$$t_g \geq t_j + p_j x_{jgk} - T(1 - x_{jgk}) - T(2 - y_{jk} - y_{gk})$$

and

$$t_j \geq t_g + p_g(1 - x_{jgk}) - Tx_{jgk} - T(2 - y_{jk} - y_{gk})$$

Here, the zero-one variable x_{jgk} has a physical interpretation only if operations j and g are actually assigned to machine k. If the number of combinations of possible operations and machines is large, this extension causes the already difficult MIP formulation of basic model to become even more difficult.

Changeover Times:

In general we can expect the time to schedule an individual machine will be affected by changeover times p_{jg} that depend on the specific consecutive pairs (j,g) encountered in a sequence of operations on the machine. Since, however, a zero-one variable is required to capture each changeover option, a direct mixed integer programming extension of the basic model that included changeovers would be excessively large.

A less monolithic mixed integer programming approach appears necessary for this class of problems. We saw in section 2.2 how to model such changeover times on a single machine as a time dependent traveling salesman problem. Ignoring precedence constraints for the moment, the resulting sequencing/scheduling problem with changeovers, including the assignment of operations to machines, can be viewed as a collection of traveling salesman problems, one for each machine, linked together by assignment constraints. The traveling salesman problems can, in principle, be used to generate sequencing schedules for each machine. Feasible sequences must then be assembled to satisfy the precedence constraints by using dual decomposition methods. This is an area of current research.

Alternative Objective Functions:

The objective function in the basic model was to minimize the time to complete all jobs. A more general objective would be to associate a tardiness cost function $C_i(t_{N_i} + p_{N_i})$ (see section 2.2) with completing each job by $t_{N_i} + p_{N_i}$, and to minimize their sum over all jobs. We point out, however, that this relatively straightforward extension of the basic Job Shop Scheduling mixed integer programming model is a serious complication for the disjunctive graph optimization approach. This is because the job-shop scheduling problem no longer reduces to a longest path problem once all disjunctive edges have been oriented. With the more general objective function, we are in effect seeking to compute simultaneously the longest path for each job.

More general objectives would involve a combination of job completion times and avoidable costs to be minimized. For example, in the case discussed above in which each operation j may be assigned to a machine in a set K_j, differentiated costs for processing operation j on different machines could be captured by adding an appropriate term to the objective function. This term might take the form

$$\sum_{j=1}^{N_I} \sum_{k \in K_j} c_{jk} y_{jk}$$

where c_{jk} is the cost of processing operation j on machine k.

Resource Allocation:

A somewhat simpler version of the job-shop scheduling problem is to treat the machine scheduling part of the problem as resource utilization constraints, rather than detailed sequencing constraints as we treated them in the model (6.1.1) through (6.1.6). This model, called the project scheduling with resource constraints model, has been studied by Christofides, Alvarez-Valdes and Tamarit (1987) and by Talbot and Patterson (1978). Christofides et al (1987) apply Lagrangean relaxation methods and, at the same time, develop stronger mixed integer programming formulations of the project scheduling with resource constraints model. Fisher (1973a, 1973b) proposed a similar Lagrangean relaxation for this class of models.

7. TREATMENT OF UNCERTAINTY

The mathematical programming models considered thus far have been strictly deterministic. The models treated all parameters of the future - demand, production rates, yields, lead times, and so on - as if they were known with certainty. This was clearly a simplifying, perhaps even a heroic, assumption, although only certain key parameters will be highly uncertain in a given application.

The fact that the models we have discussed do not explicitly treat critical uncertainties does not seriously detract from their usefulness in many if not most cases. Sensitivity analyses and multiple optimization runs under various parameter settings can usually provide the insights needed to plan effectively for an uncertain future. Moreover, developing explicit descriptions of the uncertainties may be difficult, time consuming and costly. Since production managers have only recently begun to seriously consider using deterministic mathematical programming models to support their planning and scheduling activities, it would be imprudent to suggest that their attention should be directed immediately at the sophisticated and more difficult to understand stochastic models that we discuss in this section.

Nevertheless, some production planning and scheduling applications would undoubtedly benefit greatly from explicit modeling of key uncertainties. This is the situation, for example, in the manufacture of style goods (eg, dinnerware, clothing). Typically, a company with a product line of 100 distinct style goods items might expect only 5 to 10 items to ultimately experience heavy sales, but it is difficult to forecast in advance which items these will be. Another example where uncertainty is critical is in the manufacture of semiconductor wafers where the yields of certain production stages have high variance, especially in the early periods of production.

Ignoring for the moment issues of model size and computational complexity, stochastic programming with recourse models (see Wets (1983), or Wagner (1969)) are an extension of the deterministic models discussed above that permit production planning and scheduling uncertainties to be explicitly modeled and evaluated. These models consider simultaneously multiple scenarios of an uncertain future. Optimal contingency plans for each scenario are computed along with here-and-now strategies that optimally hedge against these plans. Such hedging strategies cannot in general be identified by any deterministic model.

Bitran, Haas and Matsuo (1987) present a stochastic mixed integer programming model for production planning of style goods with high setup costs and forecast revisions. The model is a stochastic generalization of the multi-item discrete parts production model considered in section 5.2. The key element of uncertainty in their model is demand for the style goods which is concentrated in the last period of the year; forecasts of demands are revised each period.

The objective of their model is to determine a production plan that maximizes expected profit. The objective function includes selling price, salvage value, and the costs of material, inventory holding costs, setup costs, and the loss of goodwill by not being able to satisfy demand. Using the model's hierarchical structure (see section 5.4), Bitran et al formulate a deterministic approximation to the stochastic model that gives good results under certain assumptions about problem structure.

Hiller (1986) generalized the multi-stage discrete parts manufacturing model of section 5.3 to a stochastic programming model with recourse for the case when demand for finsihed goods is uncertain. This model rationalizes the computation of safety stocks at all stages in a multi-stage environment taking into account capacity constraints that limit the buildup of such stocks. This stochastic programming model also provides new insights into the risks faced by a production manager (de-valued inventory or lost sales). The model's structure allows such risks to be constrained; parametric analyses of expected costs versus various measures of risk can also be computed. Beale, Forrest and Taylor (1980) report on computational experiments with a similar class of models.

We return briefly to a discussion of approaches for dealing with the size and complexity of stochastic programming models for production planning and scheduling which are generalizations of large scale deterministic models that are complex in their own right. Again, decomposition methods are available for breaking up large, monolitihic models. In particular, Benders' decomposition method applied to a stochastic programming with recourse model allows the model to be decomposed into a master model involving here-and-now decisions, plus a sub-model for each scenario of the uncertain future (Birge (1985)). Bienstock and Shapiro (1988) report on a successful implementation of such a decomposition for a capacity expansion planning model; they also discuss how to combine Benders' decomposition applied to mixed integer programming with its application to stochastic programming.

8. COORDINATING PRODUCTION WITH OTHER COMPANY ACTIVITIES

Mathematical programming models provide management with analytic tools that allow company activities to be much more broadly coordinated than they were in the days before the information revolution. The implication to manufacturing firms goes beyond opportunities for more effective production planning and scheduling. Using decision support systems based on mathematical programming models, management can also make more effective decisions about purchasing, distribution, marketing and strategic plans from the perspective of the manufacturing engine that drives the firm. In this section, we review briefly several applications that indicate how models have promoted improved coordination within manufacturing firms.

Purchasing is an important, but generally neglected, functional area that admits useful analysis by models. Bender, Brown, Isaac and Shapiro (1985) report on the application of mixed integer programming models to support vendor contract selection within IBM. The models consider a family of parts; as many as 100 parts have been evaluated by a single model. Contract selection is difficult because a buyer must consider simultaneously multiple vendors and contracts. In particular, each vendor will offer one or more contracts for one or more parts with volume price breaks, fixed costs, and even resource constraints (machine time, labor). The purchasing decisions are driven by parts requirements that are obtained automatically from an MRP system. The objective is to minimize acquisition, in-bound transportation and inventory costs over multiple period planning horizons of a few months to three years.

The initial application of the modeling system was to the acquisition of parts for mainframe computers. Subsequently, the system has been used to develop purchasing plans for parts from a range of products. The system has also been employed as a negotiating tool with vendors whose quoted prices exclude them from an optimal solution.

Coordination of production plans with distribution plans for firms with multiple production sites is a second important area where models have proven useful. Klingman, Phillips, Steiger, Young (1987) report on an ambitious and successful project at Citgo Petroleum Company in which large scale linear programming refinery models were linked to a network optimization model for evaluating decisions regarding short-term distribution, pricing, inventory and exchange agreements. These models were imbedded in an integrated information and corporate planning system whose use has contributed significantly to marketing and refining profits.

Brown, Shapiro and Singhal (1987) report on the successful implementation and testing of a tactical production/distribution planning model for an industrial gases company with mutliple plants. Given monthly demand forecasts for products by customer locations, the model determines simultaneously which customers to allocate to each plant, how much of each product should be made by each plant, and the production slates and lengths of time slates are run at each plant, so as to minimize total production and distribution costs. The production sub-model in this application was derived from chemical engineering models that describe optimal plant configurations under a variety of output rates for each product. This sub-model was subsequently transformed into an individual plant production scheduling model that runs on a mini-computer in the plant.

Shapiro (1984) discusses a model for optimizing manufacturing and distribution plans for a nationwide consumer products company. This model takes into account manufacturing costs and capacities, warehouse locations, and distribution costs in computing a strategy that minimizes the total cost of products delivered to markets with fixed demands. Using a quantitative marketing models developed by Little (1975), Covert (1987) extended a version of the consumer products model to incorporate marketing and sales plans. In the extension, demand is considered endogeonous (variable) and created by various marketing and sales strategies regarding prices, promotion, advertsing, and so on. The objective of the model becomes maximization of net revenues which equals gross revenues from variable sales minus marketing and sales costs to create optimal demand, and minus manufacturing and distribution costs to meet these demands. Shycon (1988) reports on the application of this approach to tactical and strategic planning in a forest products company.

Brown, Geoffrion and Bradley (1981) implemented and tested a large scale mixed integer programming for coordinating yearly production and sales plans for manufacturers with limited shared tooling. The objective function in this model is maximization of net revenues equalling gross revenues minus production and inventory holding costs. Product mix decisions are allowed within a range of forecasted demand. Their model is appropriate for companies that make cast, extruded, molded, pressed or stamped products.

Lagrangean relaxation methods (see section 3) were used to efficiently extract demonstrably good solutions to this model. The sub-models in the dual decomposition consisted of: (1) network optimization models to determine monthly production plans for tools and tool/machine combinations; and (2) dynamic lot-size models (see section 2.2) to determine production plans for each item. A planning system based on the model was implemented and used regularly at a large manufacturing company.

Models for analyzing strategic planning issues in a manufacturing firm require aggregate production planning and scheduling sub-models. The types of options to be evaluated by such a model include plant and infrastructure capacity expansion, mergers, acquisitions, long term raw materials contracts, new technologies, new markets, and so on. Dembo and Zipkin (1983) discuss methods for aggregating refinery models such as the one discussed in section 4.1 so that they may be used in strategic analyses. Hiller and Shapiro (1986) show how to incorporate manufacturing learning effects into capacity expansion planning.

Nonlinear mixed integer programming models for capacity expansion planning of electric utilities have received considerable attention (e.g., Noonan and Giglio (1977), Bloom (1983)). Bienstock and Shapiro (1988) devised and implemented a stochastic programming with recourse model that explicitly treats uncertainties reagrding demand, fuel costs, and environmental restrictions.

9. FUTURE DIRECTIONS

In previous sections, we discussed a broad range of production planning and scheduling applications of mathematical programming. Although many of the models we presented were large scale, methodologies for determining demonstrably good solutions were presented. Suprisingly large linear and mixed integer programming models can be optimized by monolithic branch-and-bound approaches, at least to a small error tolerance, by commercial codes on today's mainframe computers. Decomposition methods for large scale models are conceptually appealing and, for selected applications, have already proven themselves effective in extracting good solutions reasonably quickly.

This is not to suggest, however, that major improvements in optimization algorithms and methods are not desirable and necessary, especially for time critical scheduling applications. Although mathematical programming modeling constructs provide a rich and powerful basis for abstracting and analyzing production planning and scheduling problems, the scientific community has not yet demonstrated that the models can be routinely applied to real-world problem solving in a flexible, relaible and understandable manner. The time appears exceedingly ripe for pursuing basic and applied research developments that will permit the promise of mathematical programming models to be better realized.

We envision four main areas of important research in mathematical programming over the next five to ten years:

- o mixed integer programming model formulations and solution methods
- o parallel computing
- o modeling languages
- o integration with knowledge-based systems

These areas are discussed briefly in the paragraphs that follow.

MIP Model Formulations and Solution Methods:

The central role played by mixed integer programming models in production planning and scheduling is evident from the applications reviewed in previous sections. In section 5, we discussed the promising new research on stronger mixed integer programming formulations that have already led to computational breakthroughs on some models. We have also seen a growing number of applications of Lagrangean relaxation (dual decomposition) methods. When combined with heuristics for determining feasible solutions, these methods have proven extremely successful in rapidly determining demonstrably good solutions to complex models. Moreover, new heuristic methods based on simulated annealing (e.g., Hajek (1988)) have recently produced promising results for certain classes of combinatorial optimization models. Once the research community reaches an understanding of how to integrate these three complementary approaches to mixed integer programming - model formulation, Lagrangean relaxation, and heuristics - the expectation is that greatly enhanced computation of demonstrably good solutions will be achieved.

Parallel Computing:

Mathematical programming model optimization should also be greatly enhanced by computations carried out on coarse grained parallel computers. These are computers consisting of a dozen to several hundred interdependent processors. Each processor has significant computing power and core memory in its own right, and is efficiently linked to other processors and/or large shared memories. The advantages of implementing decomposition or mixed integer programming branch-and-bound methods on such a computer are obvious (see Brown, Shapiro and Waterman (1988) or Magee and Shapiro (1989)); we simply await projects that will allow these advantages to be fully demonstrated.

Modeling Languages:

In many respects, the process of moving forward from a production manager's problems and his/her data base to an appropriate and correct family of mathematical programming models is more arduous than the process of optimizing the model. For this reason, the science and art of model generation has recently attracted a great deal of attention (Brown, Northup and Shapiro (1986), Geoffrion (1987), Fourer, Gay and Kernighan (1987), Brooke, Kendrick and Meeraus (1988)). Space does not permit a discussion of the central issues and proposed approaches. We simply mention that these researchers and others are studying symbolic, object-oriented, non-procedural computer implementation approaches to model generation that should lead to a much better understanding of how to create flexible and effective decision support systems based on mathematical programming models.

Integration with Knowledge-Based Systems:

A great deal has been spoken and written about the potential integration of knowledge-based (that is, expert systems) with mathematical programming models. Very little has been achieved from a formal viewpoint, although a number of ad hoc implementations have been reported in the literature. Formalisms of model generation discussed in the previous paragraph may soon provide the missing links for insightful integrations of the two technologies.

An expert system could prove very useful in determining the most effective mathematical programming model to generate for a given production planning or scheduling problem. An expert system would also be very useful in explaining or interpreting an optimal solution to such a model. Finally, for time-critical applications requiring a large scale model, computation could be significantly speeded up by having an expert system direct the brnach-and-bound search or other solution algorithms. These three potential areas of integration require the automation of expertise about <u>two</u> domains: the real-world production planning and scheduling domain, and the associated domain of relevant models.

10. REFERENCES

Adams, J., E. Balas and D. Zawack (1988), "The shifting bottleneck procedure for job-shop scheduling," Man. Sci., 34, pp 391-401.

Afentakis, P. and B. Gavish (1986), "Optimal lot-sizing algorithms for complex product structures," Opns. Res., 34, pp 237-249.

Afentakis, P., B. Gavish and U. Karmarkar (1984),"Computationally efficient optimal solutions to the lot-sizing problem in multistage assembly systems," Mgmt. Sci., 30, pp 222-239.

S. Axsater (1981), "Aggregation of product data for hierarchical production planning," Opns. Res., 29, pp 744-756.

Bahl, H. C., L. P. Ritzman and J. N. D. Gupta (1987), "Determining lot sizes and resource requirements: A review," Opns. Res., 35, pp 329-345

Baker, T. E. and L. S. Lasdon (1985), "Successive linear programming at Exxon," Man. Sci., 31, pp 264-274.

Ball, M. O. and M. J. Magazine (1988), "Sequencing of insertions in printed circuit board assemblies," Opns. Res., 36, pp 192-201.

Balas, E. (1979), "Disjunctive programming," Annals of Discrete Mathematics 5, pp 3-51.

Barany, I., T. J. Van Roy and L. A. Wolsey (1984), "Strong formulations for multi-item capacitated lot-sizing," Mgmt. Sci., 30, 1255-1261.

Beale, E. M. L., J. J. H. Forrest and C. J. Taylor (1980), "Multi-time-period stochastic programming," chapter 23 in Stochastic Programming, edited by M. A. H. Dempster, Academic Press.

Beigler, L. T. (1985), "Improved infeasible path optimization for sequential modeular simulators-II: the optimization algorithm", Computers and Chemical Engineering, pp 257-267.

Beigler, L. T. and R. R. Hughes (1985), "Feasible path simulationwith sequential modeular simulators", Computers and Chemical Engineering, pp 379-394.

Bender, P. S., R. W. Brown, M. H. Isaac and J. F. Shapiro (1985), "Improving purchasing productivity at IBM with a normative decision support system," Interfaces, 15, pp. 106-115.

Bienstock, D. and J. F. Shapiro (1988), "Optimizing resource acquisition decisions by stochastic programming," Man. Sci., 34, pp 215-229.

Billington, P. J., J. O. McClain and L. J. Thomas (1983), "Mathematical approaches to capacity-constrained MRP systems: review, formulation and problem reduction," Mgmt. Sci., 29, 1126-1141.

Birge, J. R. (1985), "Decomposition and partitioning methods for multistage stochastic linear programs," Opns. Res., 33, pp 989-1007.

Bitran, G. R., E. A. Haas and A. C. Hax (1981), "Hierarchical production planning: A single stage system," Opns. Res.,29, pp 717-743.

Bitran, G. R., E. A. Haas and H. Matsuo (1987), "Production planning of style goods with high setup costs and forecast revisions," Opns. Res., 34, 226-236.

Bloom, J. A. (1983), "Solving an electricity generating capacity expansion planning problem by generalized Benders'decompositon," Opns. Res., 31, pp 84-100.

Brooke, A. D. Kendrick and A. Meeraus (1988), GAMS: A User's Guide, The Scientific Press.

Brown, G. G., A. M. Geoffrion, and G. H. Bradley (1981), "Production and sales planning with limited shared tooling at the key operation," Man. Sci., 27, 247-258.

Brown, R. W., W. D. Northup and J. F. Shapiro (1986), "A modeling and optimization system for business planning," in Computer Methods to Assist Decision Making, edited by G. Mitra, North-Holland.

Brown, R. W., J. F. Shapiro and P. J. Waterman (1988), "Parallel computing for production scheduling," Manufacturing Systems, 6, pp 56-64.

Brown, R. W., J. F. Shapiro and V. M. Singhal (1987), "Production allocation modeling system: Optimizing for competitive advantage in a mature manufacturing industry," submitted for publication.

Buchanan, J. E. (1988), personal communication, Stone & Webster Overseas Consultants, Inc., Houston.

Buzacott, J. A. and D. D. Yao (1986), "Flexible manufacturing systems: A review of analytical models," Mgmt. Sci., 32, 890-905.

Calamaro, J. P. (1985), "Implementation of a multi-stage production planning system," S. M. Thesis, Operations Research Center, Massachusetts Institute of Technology.

Chapman, P. T. (1985), "Decision models for multi-stage production planning," Technical Report No. 186, Operations Research Center, MIT.

Christofides, N., R. Alvarez-Valdes and J. M. Tamarit (1987), "Project scheduling with resource constraints: A branch and bound approach," European J. of Opns. Res., 29, pp 262-273.

Covert, K. B. (1987), "An optimiziation model for marketing and production planning," M. S. Thesis, Sloan School of Management, Massachusetts Institute of Technology.

Crowder, H., E. L. Johnson and M. Padberg (1983), "Solving large-scale zero-one linear programming problems," Opns. Res., 31, pp 803-834.

Crowston, W. P. and M. H. Wagner (1973), "Dynamic lot-size models for multistage assembly systems," Mgmt. Sci. 20, 14-21.

Crowston, W. P., M. H. Wagner and J. F. Williams (1973), "Economic lot-size determination in multistage assembly systems," Mgmt. Sci., 19, 517-527.

Dantzig, G. B. (1955), "Optimal solution of a dynamic Leontief model with substitution," Econometrica, 23, 295-302.

Dantzig, G. B. (1963), Linear Programming and Extensions, Princeton University Press.

Dembo, R. S. and P. Zipkin (1983), "Construction and evaluation of compact refinery models," pp 525-540 in Energy Models and Studies, edited by B. Lev, North-Holland.

Dzielinski, B. and R. Gomory (1965), "Optimal programming of lot sizes, inventories, and labor allocations," Mgmt. Sci., 11, 874-890.

Eppen, G. D. and R. K. Martin (1987), "Solving multi-item capacitated lot-sizing problems using variable reduction," Opns. Res., 35, 832-848.

Feo, T. A. and D. S. Hochbaum (1986), "Lagrangian relaxation for testing infeasibility in VSLI routing," Opns. Res., 34, pp 819-831.

Fisher, M. L. (1973a), "Optimal solution of scheduling problems using lagrange multipliers: Part I," Opns. Res., 21, pp 1114-1127.

Fisher, M. L. (1973b), "Optimal solution of scheduling problems using lagrange multipliers: Part II," pp. 294-318 in Symposium on the Theory of Scheduling and Its Applications, edited by S. Elmagrahby, Springer-Verlag.

Fisher, M. L. (1981), "The Lagrangean relaxation method for solving integer programming problems," Man. Sci., 27, pp 1-18.

Fourer, R., D. Gay and B. Kernighan, (1987), "AMPL: A mathematical programming language," Computing Science Technical Report No. 133, AT&T Bell Laboratories, Murray Hill, NJ.

Graves, S. C. (1982), "Using Lagrangean techniques to solve hierarchical production planning problems," Mgmt. Sci., 28, 260-275.

Geoffrion, A. M. (1974), "Lagrangean relaxations for integer programming," pp 82-114 in Math. Prog. Study 2: Approaches to Integer Programming, edited by M. L. Balinski, North-Holland.

Geoffrion, A. M. (1987), "An introduction to structured modeling," Man. Sci., 33, pp 547-588.

Hajek, B. (1988), "Cooling schedules for optimal annealing," Math. of Opns. Res., 13, pp 311-329.

Hax, A. C. and H.C. Meal (1975), "Hierarchical integration of production planning and scheduling," pp 53-69 in North Holland/TIMS Studies in the Management Sciences, Vol 1. Logistics, North Holland and American Elsevier.

Hiller, R. S. (1986), "Stochastic programming approximation methods with applications to multistage production planning," Ph.D. dissertation, Operations Research Center, MIT.

Hiller R. S. and J. F. Shapiro (1986), "Optimal capacity expansion when there are learning effects," Man. Sci., 32, pp 1153-1163.

Karmarkar, U. S. Kekre and S. Kekre (1987), "The dynamic lot-sizing problem with startup and reservation costs," Opns. Res., 35, pp 389-398.

Karwan, M. H. and R. L. Rardin (1979), "Some relationships between Lagrangian and surrogate duality in integer linear programming," Math. Prog., 17, pp 320-334.

Klingman D., N. Phillips, D. Steiger and W. Young (1987), "The successful deployment of management science throughout Citgo Petroleum Corporation," Interfaces, 17, pp 4-25.

Lagewig, B., J. K. Lenstra and A. H. G. Rinooy Kan (1977), "Job shop scheduling by implicit enumeration, Man. Sci. 24, pp 441-450.

Lasdon, L. S. and R. C. Terjung (1971), "An efficient algorithm for multi-item scheduling," Opns. Res., 19, 946-969.

Lasdon, L. S., A. D. Waren and S. Sarkar (1988) "Interfacing optimizers with planning languages and process simulators", pp 238-262 in Mathematical Models for Decision Support, edited by G. Mitra, Springer Verlag.

Lawler, E. L. (1978), "Sequencing jobs to minimize total weighted completion time subject to precedence constraints, Ann. Disc. Math., 34, pp 75-90.

Lawler, E. L., J. K. Lenstra, A. H. G. Rinooy Kan, and D. B. Shmoys (1985), editors, The Traveling Salesmand Problem: A Guided Tour of Combinatorial Optimization, John Wiley & Sons.

Little, J. D. C. (1975), "Brandaid: A marketing mix model, Part I, Structure," Opns. Res., 23, pp 628-655.

Love, S. F. (1972), " A facilities in series inventory model with nested schedules," Man. Sci., 18, pp 327-338.

Magee, T. M. and J. F. Shapiro (1989), "An implementation of branch-and-bound methods for mixed integer programming on the Butterfly parallel computer," in progress.

Manne, A. S. (1958), "Programming of economic lot sizes," Mgmt. Sci., 4, 115-135.

Martin, R. K. (1987), "Generating alternative mixed-integer programming models using variable redefinition," Opns. Res., 35, 820-831.

Nemhauser, G. L. and L. A. Wolsey (1988), Integer and Combinatorial Optimization, John Wiley & Sons.

Noonan F. and R. J. Giglio (1977), "Planning power generation: A nonlinear mixed integer model employing Benders decomposition," Man. Sci. 23, pp 946-956.

Papadimitrou, C. H. and K. Stieglitz (1982), Combinatorial Optimization: Algorithms and Complexity, Prentice-Hall.

Picard J. C. and M. Queyranne (1978), " The time-dependent traveling salesman problem and application to the tardiness problem in one-machine scheduling," Opns. Res., 26, pp 86-110.

Roy, B. and B. Sussmann (1964), "Les problemes d'ordonnancement avec contraintes disjonctives," Note DS No. 9 bis, SEMA.

Schrage, L. (1986), Linear, Integer and Quadratic Programming with Lindo, Third Edition, The Scientific Press.

Schrage, L. and K. R. Baker (1978), "Dynamic programming solution for sequencing problems with precedence constraints," Opns. Res., 26, pp 444-449.

Shapiro, J. F. (1979a), Mathematical Programming: Structures and Algorithms, John Wiley & Sons.

Shapiro, J. F. (1979b), "A survey of Langrangean techniques for discrete optimization," Ann. Discrete Math., Vol. 5, pp 113-138.

Shapiro, J. F. (1984), "Practical experience with optimization models, pp 67-78 in The Future of Optimization Models for Strategic Planning, edited by T. H. Naylor and C. Thomas, North-Holland.

Shapiro, J. F. (1989) "Iterative modeling and optimization of process manufacturing scheduling problems," in progress.

Shycon, H. N. (1988), personal communication, Ernst & Whinney, Washington D.C.

Stecke, K. E. (1983), "Formulation and solution of nonlinear integer production planning problems for flexible manufacturing systems," Mgmt. Sci., 29, 273-288.

Talbot, F. B. and J. H. Patterson (1978), "An efficient integer programming algorithm with network cuts for solving resource constrained scheduling problems," Man. Sci., 24, pp 1163-1174.

Van Roy T. J. and L. A. Wolsey (1986), "Valid inequalities for mixed 0-1 programs," Discrete Applied Mathematics, 14, pp 199-213.

Van Roy T. J. and L. A. Wolsey (1987), "Solving mixed 0-1 programs by automatic reformulation," Opns. Res., 35, pp 45-57.

Veinott, A. F. (1969), "Minimum concave cost solution of Leontief substitution models of multi-facility inventory systems," Opns. Res., 17, pp 262-291.

Wagner, H. M. and T. M. Whitin (1958), "Dynamic version of the economic lot size model," Mgmt. Sci., 5, 89-96.

Wagner, H. M. (1969), Principles of Operations Research, first edition, Prentice-Hall.

Wets, R. J. B. (1983), "Stochastic programming solution techniques and approximation schemes," pp 506-603 in Mathematical Programming State-of-the-Art, edited by A. Bachem, M. Grotschel, and B. Korte, Springer-Verlag.

Williams, H. P. (1985), Model Building in Mathematical Programming, John Wiley & Sons, second edition.

Zangwill, W. I. (1966), "A deterministic multi-period production scheduling model with backlogging," Man. Sci., 13, pp 105-119.

Zangwill, W. I. (1968), "Minimum concave cost flows in certain networks," Mgmt. Sci., 14, 429-450.

Zangwill, W. I. (1969), "A backlogging model and multi-echelon model of a dynamic economic lot-size production system -- a network approach," Mgmt. Sci., 15, 506-527.