A FORWARD DYNAMIC PROGRAMMING APPROACH FOR GENERAL UNCAPACITATED MULTI-STAGE LOT-SIZING PROBLEMS

Jeffrey M. Alden
Candace Arai Yano

Department of Industrial & Operations Engineering

University of Michigan

Technical Report 86-11
April 1986

A FORWARD DYNAMIC PROGRAMMING APPROACH FOR GENERAL UNCAPACITATED MULTI-STAGE LOT-SIZING PROBLEMS

Jeffrey M. Alden

and

Candace Arai Yano

Department of Industrial and Operations Engineering

University of Michigan

April 1986

A FORWARD DYNAMIC PROGRAMMING APPROACH FOR GENERAL UNCAPACITATED MULTI-STAGE LOT-SIZING PROBLEMS

Abstract

We propose and test a forward dynamic programming procedure for lot-sizing in general multi-stage production and distribution systems without capacity constraints. The procedure permits time-varying demands and costs as well as demand for component parts. A simple yet extremely efficient pruning procedure which relies upon a novel state definition is developed. Under relatively mild conditions, it is shown that the pruning technique limits the size of the state space (in the worst case) to a linear function of the number of periods in the study horizon. We also discuss how the procedure can be used to detect forecast horizons. Computational results are reported.

A FORWARD DYNAMIC PROGRAMMING APPROACH FOR GENERAL UNCAPACITATED MULTI-STAGE LOT-SIZING PROBLEMS

Introduction

Lot-sizing problems in multi-stage production and distribution systems have received considerable attention (see Billington, et al. [1983] for a recent review). Most of the effort has focused on uncapacitated arborescent systems (i.e., pure assembly or pure distribution systems), but recent work (e.g., Maxwell and Muckstadt, [1985]) is beginning to deal with more general systems with capacity constraints.

In this paper we propose and test a forward dynamic programming approach for lot-sizing in general multi-stage production-distribution systems without capacity constraints. (Extensions of the procedure to capacity constrained systems will be considered in a sequel). We view the approach as a generalization of earlier work since it permits time-varying demand and costs for all components. It can therefore handle demand for components as repair parts.

Using a forward dynamic programming procedure gives us an important capability: the procedure can identify forecast horizons when they exist. (A forecast horizon is a point in time beyond which problem data do not affect any decision in the current or previous periods.) As such, it has the potential for identifying (initial) decisions which are optimal in the long run even if data are available for only a finite horizon.

There are, however, some limitations of the approach. As a dynamic programming procedure, it suffers to a limited degree from the "curse of dimensionality." Although our state pruning technique (described later) limits the state space to a very small fraction of the worst case scenario, the procedure would be best utilized for product structures that have been collapsed or reduced in some fashion. This problem reduction can be done using the approach of Billington, McClain and Thomas [1983] or of Maxwell and Muckstadt [1985].

In the next section, we describe the problem assumptions and the basic dynamic programming procedure. We then describe a state pruning technique which reduces the

size of the state space and makes the procedure computationally tractable. We also present a proof that the number of states in a period is uniformly bounded from above when there is no speculative motive for holding inventory. In section 3, we explain how forecast horizons can be detected using the proposed procedure. Section 4 presents computational results using the procedure. We conclude with a discussion in Section 5.

2.0 The General Multi-item Lot-Sizing Problem

We examine lot-sizing problems in general multi-stage production-distribution systems without capacity constraints. We use the term production-distibution system to reflect the fact that each item in the product structure can have multiple predecessors and multiple successors. We assume that demands and costs are deterministic, but they are permitted to vary with time. External demands for both finished products and components (e.g., for repair parts) are permitted. We assume that production at each stage and transportation between stages occur instantaneously. (Constant production times can be incorporated by an appropriate shift of time indices). We also assume that all demand must be satisfied on time (i.e., no backordering).

Costs to be included are production and order setup costs, linear production or procurement costs, and linear inventory holding costs charged on end-of-period inventory. We also assume that initial inventory of all items is zero. If there is no speculative motive for holding inventory, a problem with positive initial inventory can be transformed into an equivalent problem with zero initial inventory by using net production requirements instead of gross demand (see Zabel [1964]). Except for dealing with initial inventory, the procedure does not require the assumption of no speculative motive. Nonetheless, this assumption in conjunction with a mild assumption regarding demand is sufficient to guarantee a *linearly* bounded state space growth with the study horizon and hence we will assume no speculative motive for holding inventory.

Consistent with this assumption, we consider uncapacitated problems with solutions satisfying the Wagner-Whitin [1958] property for each production run or setup. Hence, whenever production occurs, the quantity produced will satisfy demand (for that item) over

a set of consecutive periods. This means that an optimal solution exists where demand for an item is satisfied from the most recent prior production run of the item and where positive inventory of an item will not be carried into a period in which the item is produced. This implies that the optimal lot-sizes of each item can be calculated from the optimal sequence of setups of the items (given the problem structure and demand data) and so we can concentrate on simply finding the optimal sequence of setups for each item. Limiting consideration to solutions of this type for uncapacitated problems is consistent with results in Erickson et al.[1980], which extend the results of Wagner and Whitin [1958] and Zangwill [1969] to more general networks.

Before proceeding with a description of the dynamic programming procedure, we present notation used throughout the remainder of the paper. Note that we use some standard network terminology, where the production facility of item i is denoted as node i, and successors and predecessors are defined in the usual way for directed graphs. We only require that the network representation of the product structure is finite and contains no directed cycles.

Notation:

$K_{it} \equiv$ setup cost for item i in period t,

$h'_{it} \equiv$ inventory holding cost for item i in period t,

$c_{it} \equiv$ variable production cost per unit for item i in period t,

$N \equiv$ the number of nodes (items) in the network (item $i \in \{1, 2, ..., N\}$),

$P_i \equiv$ immediate predecessors of node i,

$r_{ij} \equiv$ amount of item j required in each unit of item i where $j \in P_i$,

$T \equiv$ number of periods in the study horizon, and

$d_{it} \equiv$ independent demand for item i in period t.

We distinguish two types of demand for an item in a period: dependent demand which are requirements for material used in producing immediate successors, and independent demand which reflect requirements generated outside of the production system. Independent demand for each item is given as data, whereas dependent demand for an item will

also depend on the sequence of setups made for successor items.

For the sake of notational simplicity, throughout the remainder of the paper we consider an equivalent problem (see Blackburn and Kunreuther [1974]) using a revised inventory holding cost, h_{it}, where $h_{it} = h'_{it} + c_{it} - c_{i,t+1}$. This permits us to eliminate reference to the variable production cost by letting h_{it} refer to the relative cost of producing and holding a unit in period t rather than producing it in period $t+1$. The assumption of no speculative motive is equivalent to requiring that $h_{it} \geq 0$ for all i and t.

2.1 The Decision Process and Two Dynamic Programming State Definitions

We first introduce a decision process which gives our interpretation of how decisions are made in the problem and then we define two equivalent Dynamic Programming (DP) states. The first state definition, based on setup times, is a natural result of the decision process and is the easier of the two to understand. We use the first definition of a state to facilitate our discussion. The second state definition, based on net inventory holding rates, arises from the information required to determine the costs associated with making decisions in the future. The second state definition is much easier to use in proving some of our results and in the coding of the DP algorithm.

We view the problem as making a sequence of binary decisions over time to meet both independent demand and dependent demand for an item in each period. Our alternatives are to

1) setup and produce the item, or

2) increase the quantity of the most recent production run of the item and carry the inventory forward.

To represent the decision process, we define a policy for period t to be the vector $Y(t)$, where the i^{th} element, $Y_i(t)$, is given by

$$Y_i(t) = \begin{cases} 1, & \text{if item } i \text{ has a setup in period } t; \\ 0, & \text{otherwise.} \end{cases}$$

Note that a policy $Y(t)$ may fix the setup of an item in period t, but does not fix the

lot-size of the item. The lot-size will not be determined until later setups of the item and successor items are made.

A state can be represented by a set of ordered sets $A(t)$ with the i^{th} ordered set, denoted by $A_i(t)$, giving the production runs (identifying the periods) for each item which would increase in quantity with an increase in the the demand for item i in period t. Thus, $A_i(t)$ includes the most recent setup for item i in or prior to period t, as well as the setups of all predecessors of item i which would supply, either directly or indirectly, the material required in the production run of item i. The set $A(t)$, in conjunction with problem data, thus stores all the information required to calculate the cost of any future decision which might affect production quantities in period t or earlier.

There are two reasons for considering the possibility of demand for item i increasing in period t. First, as time progresses, the production lot in or beyond period t for a successor may increase, thus creating a larger dependent demand in period t for the item. Second, we may want to increase the inventory of the item in period t to supply independent demand for the item in a period beyond t. These possibilities must be anticipated to correctly account for the effects of current decisions on the costs of future decisions.

It is important to note that $A_i(t)$ may have more than N elements. For example, this occurs when two different items, say i and j, with a common immediate predecessor (say k), may draw from two different production runs of item k. This situation occurs in the example which follows.

We illustrate the decision process and the DP state with a simple product structure containing both distribution and assembly characteristics (see Figure 1). Assume that production of each item requires one item from each predecessor (i.e, $r_{ij} = 1$ for $j \in P_i$) except that two units of item 4 are required for each unit of item 2 ($r_{24} = 2$). Suppose that the decisions made over the first five periods result in the setups depicted in Figure 2.

Let $a_i(t)$ represent the time of the production run of item i which would directly supply input material for the production run of a successor in period t. Then the state $A(t)$ can

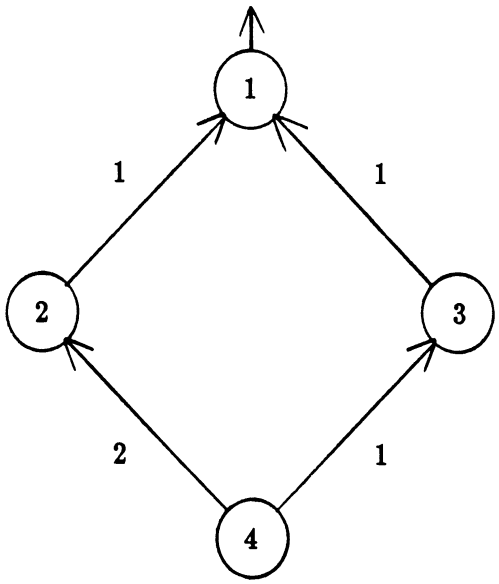

Figure 1. Network Representation of an Example Product Structure.

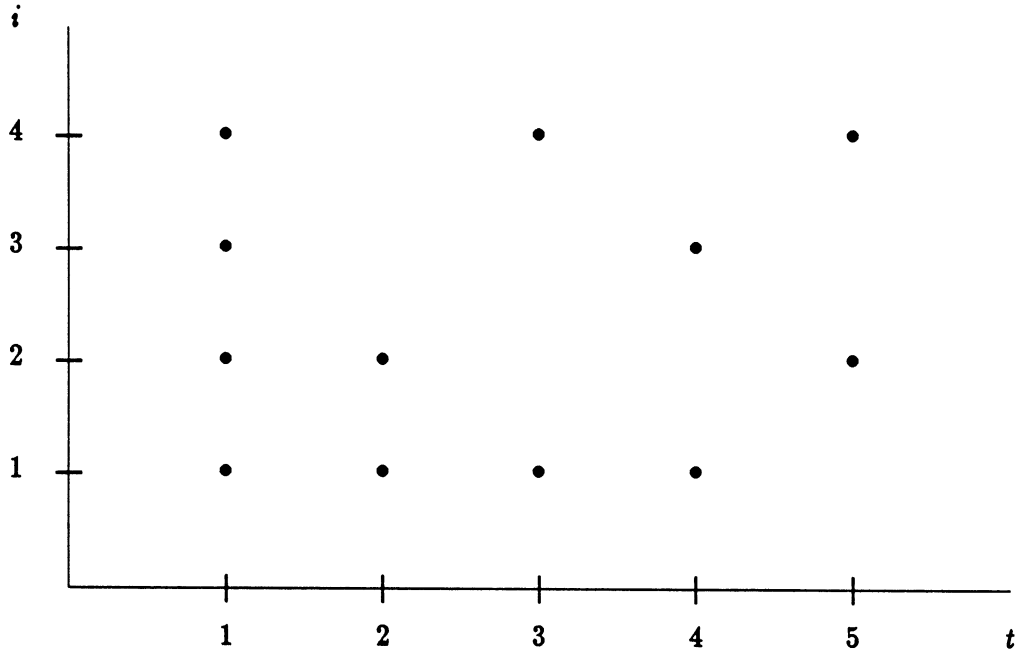

Figure 2. Graphical Representation of $Y_i(t)$. (• denotes a setup.)

be generated from the $a_i(t)$ through the recursive definition

$$A_i(t) = \begin{cases} \{a_i(t)\} \cup \{\cup_{j \in P_i} A_j(a_i(t))\}, & \text{if } P_i \neq \emptyset; \\ \{a_i(t)\}, & \text{if } P_i = \emptyset. \end{cases}$$

This recursive definition is simply a mathematical statement of the requirement that a state

must contain the latest setup of item i, which is $a_i(t)$, plus the setups of item i's predecessors that supply inputs for the setup, which is simply the union of the sets $A_j(a_i(t))$ for $j \in P_i$.

We use the example of Figure 2 to illustrate this definition. Since item 4 has no predecessors, $A_4(t)$ simply stores the most recent setup of item 4. Hence,

$$A_4(t) = \{a_4(1)\} = \{1\} \text{ for } t = 1,2;$$
$$A_4(t) = \{a_4(3)\} = \{3\} \text{ for } t = 3,4; \text{ and}$$
$$A_4(5) = \{a_4(5)\} = \{5\}.$$

Now consider item 3. In order to produce a unit of item 3 we must have produced two units of item 4 in the same or prior period. Thus $A_3(t) = \{a_3(t)\} \bigcup A_4(a_3(t))$, and

$$A_3(t) = \{1,1\} \text{ for } t = 1,2,3; \text{ and}$$
$$A_3(t) = \{4,3\} \text{ for } t = 4,5.$$

$A_2(t)$ can be determined in a similar fashion.

We now turn to item 1. Since item 1 uses items 2 and 3, which in turn require item 4, the set $A_i(t)$ must include at least 4 elements. However, two <u>different</u> production runs of item 4 may supply input to item 1 via items 2 and 3. For instance, in period 5 of the example there is a setup of item 1 which requires input from the production runs of items 2 and 3 in periods 5 and 4 respectively. The production runs of item 4 that supply these particular production runs of items 2 and 3 occur in two different periods: in period 5 to supply item 2 and in period 3 to supply item 3. So we have, after expanding the A_j's for the predecessors of item 1:

$$A_1(t) = \{a_1(t),\ a_2(a_1(t)),\ a_4(a_2(a_1(t))),\ a_3(a_1(t)),\ a_4(a_3(a_1(t)))\},$$

and in our example, $A_1(5) = \{4,2,1,4,3\}$.

To write the dynamic programming functional relationship we define

$S_t \equiv$ the set of states of period t, and

$f_t[A(t)] \equiv$ the minimum cost to reach state $A(t) \in S_t$ from the initial state $A(1)$.

Then

$$f_t[A(t)] = \min_{A(t-1)\in S_{t-1}} \{C[A(t-1), A(t)] + f_{t-1}[A(t-1)]\},$$

where $C[A(t-1), A(t)]$ is the cost of reaching state $A(t)$ from state $A(t-1)$, which is infinite if it cannot be done.

The cost of the minimum cost strategy for a T period problem is

$$f_T = \min_{A(T)\in S_T} \{f_T[A(T)]\}.$$

If a policy $Y(t)$ from state $A(t-1)$ leads to the state $A(t)$ of the next period, then we write $A(t-1) \overset{Y(t)}{\rightarrow} A(t)$ to denote this relationship. We also say that $A(t-1)$ reaches $A(t)$ when a policy $Y(t)$ exists such that $A(t-1) \overset{Y(t)}{\rightarrow} A(t)$.

The cost of any decision in period t is the cost of setting up and producing d_{it} (including holding costs to supply input to the production) if $Y_i(t) = 1$ or the cost of producing d_{it} in the period indicated by $A_i(t-1)$ and holding it until period t if $Y_i(t) = 0$. Computation of the cost of reaching any state is simplified by two facts. First, in any valid state it is necessary to have the time of a setup for item i to occur at or before the time of a setup for item j if i is a predecessor of j since otherwise there would be component backorders. Second, since variable production and inventory holding costs are linear functions, for any production run of an item, there is always a least cost source (production run) for each predecessor. When $h_{it} \geq 0$ for all i and t, this least cost source is always the most recent production run of the item. Thus, for any given state $A(t)$ each demand for a finished product or component has a unique set of source production batches, and thereby also a unique cost per unit associated with producing and holding inventory which we call the net (revised) holding cost. This leads to our a second definition of a DP state based on net holding costs.

Let $H_i[A(t)]$ represent the marginal inventory holding cost which would be incurred if one more unit of item i were produced using the setups in $A(t)$ to satisfy demand in period t. Note that $H_i[A(1)] = 0$ for the initial state $A(1)$ since we assume $Y_i(1) = 1$ for each i.

If $Y_i(t) = 1$, then the demand would be satisfied by production in the same period, so there is no inventory holding cost. On the other hand, if $Y_i(t) = 0$, then updated net

inventory cost is the inventory holding cost already incurred up to period $t-1$ from using the setups in $A(t-1)$, where $A(t-1) \overset{Y(t)}{\rightarrow} A(t)$, plus the cost of holding the unit for one more period, i.e., period $t-1$. Hence, for an item i with no predecessors we have

$$H_i[A(t)] = \begin{cases} H_i[A(t-1)] + h_{i,t-1}, & \text{if } Y_i(t) = 0; \\ 0, & \text{if } Y_i(t) = 1. \end{cases} \tag{1}$$

For items with predecessors, we can express $H_i[A(t)]$ in terms of the net holding costs for its immediate predecessors (provided these costs have already been computed). That is,

$$H_i[A(t)] = \begin{cases} H_i[A(t-1)] + h_{i,t-1}, & \text{if } Y_i(t) = 0; \\ \sum_{j \in P_i} r_{ij} H_j[A(t)], & \text{if } Y_i(t) = 1. \end{cases} \tag{2}$$

If item i is not produced in period t, then the net holding cost is the cost already incurred up to period $t-1$ plus the cost of holding item i for one additional period, namely period $t-1$. If item i is produced in period t, then only the predecessor-related inventory costs need to be considered. All predecessors are included in the summation since each $H_j[A(t)]$ accounts for its own predecessors and the computations begin with those items having no predecessors.

We can easily calculate the cost $C[A(t-1), A(t)]$ in terms of the net holding costs where $A(t-1) \overset{Y(t)}{\rightarrow} A(t)$ as

$$C[A(t-1), A(t)] = \sum_i d_{it} H_i[A(t)] + Y_i(t) K_{it}.$$

From this relationship we see that the net holding cost vector $H[A(t-1)]$ is sufficient to determine the cost of any policy $Y(t)$ selected while in state $A(t-1)$ and hence defines a DP state; our second DP state definition. Thus, using the net holding rate vector as the DP state and, for the moment, writing $H(t)$ instead of $H[A(t)]$, we can recast the DP formulation in the following way.

First, to simplify the presentation, we redefine

$S_t \equiv$ the set of states of period t which we will index as $(1, 2, \cdots, s_t)$,

$H^k(t) \equiv$ the k^{th} state of period t, where $H^k(t) \in S_t$,

$f_t[H^k(t)] \equiv$ the minimum cost to reach state $H^k(t) \in S_t$ from the initial state $H(1)$,

and use $H^j(t-1) \overset{Y(t)}{\rightarrow} H^k(t)$ to indicate that the policy $Y(t)$ out of state $H^j(t-1)$ produces the state $H^k(t)$.

Then the DP recursion becomes

$$f_t[H^k(t)] = \min_{j, Y(t)} \left\{ \sum_i \left[d_{it} H_i^k(t) + Y_i(t) K_{it} \right] + f_{t-1}[H^j(t-1)] \right\}, \tag{3}$$

where $H^j(t-1) \in S_{t-1}$ and $H^j(t-1) \overset{Y(t)}{\rightarrow} H^k(t)$. The net holding rate vector $H^k(t)$, in terms of $H^j(t-1)$ and $Y(t)$ is

$$H_i^k(t) = \begin{cases} H_i^j(t-1) + h_{i,t-1}, & \text{if } Y_i(t) = 0; \\ \sum_{l \in P_i} r_{il} H_l^k(t), & \text{if } Y_i(t) = 1 \text{ and } P_i \neq \emptyset; \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

Finally, the cost of the minimum cost solution is

$$f_T = \min_{k=1, \cdots, s_T} \{ f_T[H^k(T)] \}.$$

To illustrate the updating of the new state definition, we again consider the example of Figure 2. In this problem, let the costs be stationary with h_i denoting the one period holding cost per unit of item i.

Since $Y_i(1) = 1$ for all i, we have $H_i(1) = 0$ for all i. To update for period 2, we observe that items 3 and 4 had no setups, giving $H_4(2) = h_4$ and $H_3(2) = h_3 + H_4(1) = h_3$, while items 1 and 2 did have setups, yielding $H_2(2) = 2H_4(2) = 2h_4$ and $H_1(2) = H_2(2) + H_3(2) = 2h_4 + h_3$. Observe that the last value arises because one unit each of items 2 and 3 are required to produce item 1. There is no inventory holding cost due to item 2 directly; however, for each unit of item 2 two units are required of item 4 which were produced in

period 1. Hence, there is a cost of holding inventory of item 4. In addition, there is the cost of holding item 3 from period 1 for the item 1 setup in period 2.

Now consider period 5. The reader can verify that $H(4) = (2h_2 + 3h_4,\ 2h_2 + 2h_4,\ h_4,\ h_4)$. We then have $H_4(5) = 0$ since a setup occured for item 4 in period 5 and item 4 has no predecessors. Also $H_3(5) = h_3 + H_3(4) = h_3 + h_4$, and $H_2(5) = H_4(5) = 0$ since item 2 has item 4 as its only predecessor and both have setups in period 5. Finally, $H_1(5) = h_1 + H_1(4) = h_1 + 2h_2 + 3h_4$.

We now have the necessary definitions to discuss the decision process in more detail. This procedure might be viewed as an implicit enumeration scheme. As such, the primary concern is that of limiting growth of the state space. Next we discuss a pruning procedure to accomplish this.

3. A Pruning Technique

We present a pruning technique which identifies and eliminates unnecessary states, considerably reducing the size of the state space. Although more sophisticated pruning techniques can be implemented in which states in one period prune states in other periods, we only consider pruning of states in the same period. This limits the information required for the pruning to information already available for (and required by) the dynamic programming procedure. First we present a definition of pruning.

Definition:

Let $A(t)$ and $B(t)$ be two distinct DP states. Then $A(t)$ is said to prune $B(t)$ iff the optimal solution through period T restricted to using $A(t)$ is no more costly than the optimal solution restricted to using $B(t)$.

There are two simple requirements which are sufficient for $A(t)$ to prune $B(t)$. First, the cost to reach $A(t)$ from the initial state is no more that the cost to reach B(t) from the initial state. The second requirement loosely relates to the "cost to go," or the cost to complete the problem from period $t + 1$ onward. The requirement is that for each item i the cost of meeting future demand using only the setups indicated in $A(t)$ up to time t is

less than or equal to the cost of meeting future demand using only the setups in $B(t)$ up to time t. If both requirements are satisfied, removing $B(t)$ does not eliminate any uniquely optimal solution, so it can be pruned.

To check for the second requirement, we need to capture the effect of constraining the solution to using state $A(t)$ or $B(t)$ on the cost to satisfy demand of each item i until its first production run occurring after period t. We also need to capture the cost due to a potential increase of demand (of any item) in period t which might result from batching at a successor stage in period t to satisfy demand beyond period t, or from independent demand beyond period t.

The cost of satisfying demand $d_{i,t+k}$ for any non-negative k using $A(t)$ is

$$d_{i,t+k}\left\{H_i[A(t)] + \sum_{j=t}^{t+k-1} h_{ij}\right\},$$

and using $B(t)$ is

$$d_{i,t+k}\left\{H_i[B(t)] + \sum_{j=t}^{t+k-1} h_{ij}\right\}.$$

Observe that the summations are equivalent as is the constant term $d_{i,t+k}$, so it is sufficient to compare $H_i[A(t)]$ and $H_i[B(t)]$. Note also that these are the respective costs of using the setups in $A(t)$ and $B(t)$ to satisfy a unit of demand in period t. Thus, the only information we need to determine the minimum of the two costs above are the net holding cost rates $H_i[A(t)]$ and $H_i[B(t)]$.

The pruning procedure checks whether

1) the minimum cost of reaching $A(t)$ is no more than the minimum cost of reaching $B(t)$, that is, $f_t[A(t)] \leq f_t[B(t)]$, and
2) the net inventory holding cost rate for item i in $A(t)$ is no more than the net inventory holding cost rate for item i in $B(t)$ for each i, that is, $H_i[A(t)] \leq H_i[B(t)]$ for all i.

Next we prove that if these conditions hold, then $A(t)$ prunes $B(t)$.

Theorem 1. *Let $A(t)$ and $B(t)$ be any two distinct states in S_t. If $f_t[A(t)] \leq f_t[B(t)]$ and $H[A(t)] \leq H[B(t)]$, then $A(t)$ prunes $B(t)$.*

Proof:

Let $f_{t+k}[A(t+k)|A(t)]$ be the cost of the minimum cost strategy to state $A(t+k)$ from the initial state restricted to using the state $A(t)$ at time t. A sufficient condition for $A(t)$ to prune $B(t)$ is for the cost of a minimum cost strategy to <u>any</u> future state $B(t+k)$ restricted to using the state $B(t)$ to be greater than or equal the cost of a minimum cost strategy to <u>some</u> state $A(t+k)$ restricted to using the state $A(t)$ for all $k > 0$. That is, for <u>each</u> state $B(t+k)$, $k > 0$, there exists some state $A(t+k)$ such that $f_{t+k}[B(t+k)|B(t)] \geq f_{t+k}[A(t+k)|A(t)]$. Given $f_t[A(t)] \leq f_t[B(t)]$, this condition is satisfied if we can find a state $A(t+k)$ for each state $B(t+k)$ where the minimum cost to reach $A(t+k)$ from $A(t)$ is no more than the minimum cost to reach $B(t+k)$ from $B(t)$ for all $k = 1, 2, \cdots, T-t$

We will show the assumptions of Theorem 1 allow us to satisfy this sufficient condition for a particular state $A(t+k)$: the state of period $t+k$ which is reached from state $A(t)$ using the same sequence of policies used to reach state $B(t+k)$ from state $B(t)$.

We divide the proof into two parts: 1) a proof that the holding cost inequality $H_i[A(t+k)] \leq H_i[B(t+k)]$ holds for all items i and $k = 1, 2, \cdots, T-t$, and 2) a proof that the cost inequality $C[A(t), A(t+k)] \leq C[B(t), B(t+k)]$ holds for $k = 1, 2, \cdots, T-t$, where $C[A(t), A(t+k)]$ is the cost to go from state $A(t)$ to $A(t+k)$ following the same sequence of policies which leads from $B(t)$ to $B(t+k)$. We prove these results by induction on k.

Part 1. Assume the holding cost inequality, $H[A(t+m-1)] \leq H[B(t+m-1)]$, holds for all $m = 1, \cdots, k$. It obviously holds for $m = 1$ as it is an assumption of the theorem. We will prove $H[A(t+k)] \leq H[B(t+k)]$. We consider two cases: $Y_i(t+k) = 0$ and $Y_i(t+k) = 1$ for each item i taken in order so that all predecessor net holding rates in period $t+k$ have been updated prior to updating the same for item i.

Case 1. If $Y_i(t+k) = 0$ then

$$H_i[A(t+k)] = H_i[A(t+k-1)] + h_{i,t+k-1},$$

and

$$H_i[B(t+k)] = H_i[B(t+k-1)] + h_{i,t+k-1}.$$

Obviously, $H_i[A(t+k-1)] \leq H_i[B(t+k-1)]$ implies $H_i[A(t+k)] \leq H_i[B(t+k)]$ for this case.

Case 2. If $Y_i(t+k) = 1$ and item i has no predecessors, then $H_i[A(t+k)] = H_i[B(t+k)] = 0$, so the inequality is true for all components with no predecessors.

Now we inductively show the inequality is true for the general case where i has predecessors.

Since $Y_i(t+k) = 1$, then

$$H_i[A(t+k)] = \sum_{j \in P_i} r_{ij} H_j[A(t+k],$$

and

$$H_i[B(t+k)] = \sum_{j \in P_i} r_{ij} H_j[B(t+k)].$$

Let N_0 denote the set of items with no predecessors and define (recursively) N_l as the set of items with at least one predecessor in N_{l-1} and *all* predecessors among the sets $N_0, N_1, \cdots, N_{l-1}$. We have already shown the inequality holds for $i \in N_0$. Assuming that it holds for $i \in N_k$ for $k = 1, 2, \cdots, l$, we will show the inequality also holds for N_{l+1}.

Consider the items in N_{l+1}. Their predecessors are contained in N_k, $k = 0, 1, \cdots, l$, so for all $j \in P_i$ where $i \in N_{l+1}$, we have $H_j[A(t+k)] \leq H_j[B(t+k)]$. This implies

$$\sum_{j \in P_i} r_{ij} H_j[A(t+k)] \leq \sum_{j \in P_i} r_{ij} H_j[B(t+k)],$$

which, by definition of H_i gives

$$H_i[A(t+k)] \leq H_i[B(t+k)].$$

Hence the inequality holds for $l+1$. By induction on l, the inequality holds for all l and therefore for all items i. Since the holding cost inequality is true for $k = 0$ it follows by induction on k that the holding cost inequality holds for $k = 1, 2, \cdots, T-t$.

Part 2. We prove $C[A(t), A(t+k)] \leq C[B(t), B(t+k)]$ by induction on k. Assuming $C[A(t), A(t+m)] \leq C[B(t), B(t+m)]$ is true for $m = 0, 1, \cdots, k-1$, we will prove the cost inequality is true for $m = k$. Obviously, this holds for $m = 0$ as both costs are zero. By definition of the cost of a sequence of policies,

$$C[A(t), A(t+k)] = C[A(t), A(t+k-1)] + C[A(t+k-1), A(t+k)], \quad \text{and}$$
$$C[B(t), B(t+k)] = C[B(t), B(t+k-1)] + C[B(t+k-1), B(t+k)],$$

where $C[A(t+k-1), A(t+k)]$ is the cost of making in state $A(t+k-1)$ the same decision as made in state $B(t+k-1)$. Since $C[A(t), A(t+k-1)] \leq C[B(t), B(t+k-1)]$ by assumption, it is sufficient to show that $C[A(t+k-1), A(t+k)] \leq C[B(t+k-1), B(t+k)]$ to prove the case for $m = k$.

By definition,

$$C[A(t+k-1), A(t+k)] = \sum_i d_{i,t+k} H_i[A(t+k)] + Y_i(t+k) K_{i,t+k}, \quad \text{and}$$

$$C[B(t+k-1), B(t+k)] = \sum_i d_{i,t+k} H_i[B(t+k)] + Y_i(t+k) K_{i,t+k}.$$

The terms $Y_i(t+k)K_{i,t+k}$ in both equations are identical. In part 1, we proved that $H_i[A(t+k)] \leq H_i[B(t+k)]$, hence $d_{i,t+k} H_i[A(t+k)] \leq d_{i,t+k} H_i[B(t+k)]$ and so $C[A(t+k-1), A(t+k)] \leq C[B(t+k-1), B(t+k)]$. Thus the cost inequality is true for the case $m = k$. Since the inequality inequality holds for $m = 0$, $C[A(t), A(t+k)] \leq C[B(t), B(t+k)]$ for $k = 0, 1, \cdots, T-t$ by induction on k. ∎

Theorem 1 allows us to do pruning with <u>one</u> state in a period since the state will remain unpruned when done. However, to exploit the full power of pruning, it is necessary to consider all states as potential pruning states, or <u>full pruning</u>. Hence, we need to show that full pruning can be done so that for every state which is pruned an unpruned state exists which can still prune it. It turns out that the pruning operation is transitive which immediately implies that pruning can be safely performed in any desired order.

Theorem 2. *Pruning is transitive: if U prunes V and V prunes W then U prunes W.*

Proof:

Let U, V, and W be three states of period t where U prunes V and V prunes W. By the pruning conditions, $f_t[U] \leq f_t[V]$ and $f_t[V] \leq f_t[W]$, hence $f_t[U] \leq f_t[W]$. Also, $H_i[U] \leq H_i[V]$ and $H_i[V] \leq H_i[W]$ for all items i, hence $H_i[U] \leq H_i[W]$ for all items i. Thus pruning conditions are satisfied for U to prune W. ∎

3.1 A Linear Bound on the Number of States

In this section we show that under relatively mild conditions there exists a uniform upper bound on the number of unpruned states at any time t. Although this bound is not of practical significance because it is a very loose bound, it is of theoretical significance because it indicates that the maximum number of unpruned states at time t is independent of t.

Theorem 3. *Let $A^*(t)$ denote the state with production setups for all items in period t. If*

(i) all revised inventory holding costs are uniformly bounded below by $h > 0$ (i.e., there is no speculative motive for holding inventory),

(ii) the per period independent demand for all finished products is uniformly bounded below by $d > 0$ in each period,

(iii) all setup costs are finite and uniformly bounded above by K, and

(iv) all items require at least one unit of each immediate predecessor ($r_{ij} \geq 1$ for $j \in P_i$),

then $2^{N^2K/hd}$ is an upper bound on the number of unpruned states in any period t if $A^*(t)$ is used as the basis for pruning in period t.

Proof:

We first show that $A^*(t)$ prunes all states $B(t)$ containing a setup prior to some period τ. To do this, we first show that the minimum cost to reach $B(t)$ from the initial state has a lower bound which is larger than an upper bound on the minimum cost to reach $A^*(t)$ from the initial state. With these cost bounds and the fact that $H[A^*(t)] = 0$ we show

that $A(t)$ prunes $B(t)$. Finally, we show that $t - \tau$ is uniformly bounded from below by a quantity that is independent of t, thereby giving an upper limit on the number of unpruned states at any time t.

Let f_t be the minimum of $f_t[A(t)]$ over all unpruned states $A(t)$. Since $A^\bullet(t)$ has production setups for all items in period t, we must have

$$\begin{aligned} f_t[A^\bullet(t)] &= f_{t-1} + \sum_i K_{it} \\ &\leq f_{t-1}[B(t-1)] + NK, \end{aligned}$$

where $B(t-1)$ is such that $B(t-1) \overset{Y(t)}{\rightarrow} B(t)$ in the minimum cost strategy to reach $B(t)$ from the initial state.

Consider all item setups in $B(t)$ which directly or indirectly satisfy a finished product demand in period t. Suppose the earliest such setup occurs in period τ for some item i. Since $B(t)$ indicates only the production runs to be used to satisfy independent demand in period t (or beyond), the item's production quantity must be at least d units to satisfy the minimum possible demand in period t for the finished product it directly or indirectly supplies. Hence, an inventory of at least d (in the form of item i or some successor of item i) must be held from period τ through period $t-1$ at a minimum cost per unit per period of h. Hence

$$f_t[B(t)] \geq f_{t-1}[B(t-1)] + (t-\tau)hd,$$

where the inequality is further weakened by ignoring the cost of any production setups in period t to reach state $B(t)$. By comparing the bounds on $f_t[A^\bullet(t)]$ and $f_t[B(t)]$ we have $f_t[A^\bullet(t)] \leq f_t[B(t)]$ if

$$\begin{aligned} hd(t-\tau) &\geq NK, \quad \text{or} \\ (t-\tau) &\geq NK/hd \end{aligned}$$

since $hd > 0$. By definition of $A^\bullet(t)$ the holding cost rates $H[A^\bullet(t)]$ are zero and $H[A^\bullet(t)] \leq H[B(t)]$ since all holding costs are non-negative. Thus the conditiions are satisfied for $A^\bullet(t)$ to prune $B(t)$.

Let

$$\Delta t = NK/hd,$$

then $A^*(t)$ will prune any state $B(t)$ with a production run (for any item) scheduled in period $t - \Delta t$ or earlier. Observe that Δt is independent of t. Since the number of policies in each period is 2^N, the maximum number of unpruned states in any period is at most $2^{N\Delta t}$, which is also independent of t. ∎

The conditions on h and d can be relaxed with somewhat more complicated relationships among the h_{it} and d_{it} values. In essence, all that is required is that demands and inventory holding costs not all be zero prior to period t.

The bound described above is a very loose bound, since the number of dynamic programming states does not grow as 2^N per period as many different sequences of policies may lead to the same state, nor is the state $A^*(t)$ the only state used in pruning other states. Furthermore, with the assumptions of no speculative motive and no capacity constraints, nested schedules in which a production run of an item occurs only if at least one of its successors also schedules a production run are usually (although not always) optimal, and nearly all unnested schedules tend to be pruned rapidly. Thus, while there is a worst case bound on the number of unpruned states in a period which is constant for all t, experimental results (which we discuss later) suggest that the maximum number of states is much smaller, even when the conditions of the theorem are not satisfied.

4. Detection of Forecast Horizons

One of the primary advantages of a forward dynamic programming procedure is that it can detect forecast horizons when they exist. In doing so, it permits one to implement current decisions with confidence that the decision will still be optimal even after additional data is collected for periods beyond the forecast horizon. Since many lot-sizing procedures are necessarily implemented using some type of rolling horizon framework, even experimental evidence that forecast horizons do exist for these problems and estimates of the length of these horizons (as a function of the problem data) are useful.

For the problem described above, forecast horizons can be detected in two ways. First, if at any time t there is only one unpruned state and that state has all non-zero elements, then that state is optimal as are all decisions which led to that state. The reason that we require all elements to be non-zero is that a zero element indicates that the batch for the item in question may not be complete. In other words, consideration of demand beyond time t may cause the optimal batch size to increase. Thus, the information through time t is not sufficient to determine the optimal batch size of that item to be produced in the current period.

Another way to detect a forecast horizon is to examine the set of policies in period t that are part of an optimal strategy leading to an unpruned state of a later period $t+k$. If this set contains a single policy then the policy must be optimal in period t. To extend this concept further, we can also detect a forecast horizon if the set of optimal strategies to all unpruned states in period $t+k$ use a common state in period t. When this happens, the common state and all decisions leading to it must be optimal for the infinite horizon problem.

We have found experimentally that forecast horizons often exist, and that (rather amazingly) they are sometimes detected because only one unpruned state remains (i.e., the state with production setups for all items). We report these and other computational results in the next section.

5. Computational Results

There were several objectives of our computational study. First, we wanted some assurance that the state space would be "manageable" and computational times would be acceptable. Second, we wished to obtain an indication of whether forecast horizons existed for these multi-stage systems, and if so, what order of magnitude they were.

We used the algorithm to solve a variety of four-item problems with the product structure depicted if Figure 1. The base case had $d_{1t} = 500$ for all t, $d_{it} = 0$ for $i = 2,3,4$ and all t. Costs were constant over time with $K_i = 200$, 300, 400, 900 and $h_i = 1.0$, 0.3, 0.3, 0.2 for $i = 1,2,3,4$, respectively. We also set $r_{i,j} = 1$ for all applicable

TABLE 1.

Problem Data and Experimental Results

Case	Statistics for Unpruned States mean	Std. Dev.	Maximum	CPU* time
Base case	6.03	1.43	12	5.34
Discounting 0.9 per period.	7.55	2.23	16	6.87
Demand decline 10% per period.	69.17	59.90	213	168.33
Demand growth 10% per period.	3.07	2.17	8	3.19
Requirements change $r_{21} = 1$, $r_{31} = 2$, $r_{42} = 3$, $r_{43} = 4$.	3.76	1.12	7	3.50
Lumpy demand $P(d_1 = 0) = .25$; otherwise U[0,1500]. (10 problems solved.)	20.91	25.03	152	mean = 24.64 s.d. = 9.34
Time-Varying costs All costs uniform between half and and double the base case costs. (10 problems solved.)	10.04	4.95	34	mean = 8.03 s.d. = 1.397

* In seconds. Problems solved on an Apollo DN330 minicomputer.

i, j pairs. Each problem had a 30 period study horizon.

We investigated the effects of discounting, growth and decline of demand over time, and different $r_{i,j}$ values. We also randomly generated some problems with time-varing "lumpy" demand (some zeros), and time-varying costs. The measures of performance we were most concerned about were the average and maximum number of unpruned states and the computing time. The algorithm was coded in FORTRAN and executed on an Apollo DN330 minicomputer which we estimate to be at least ten to fifteen times slower than most

TABLE 2.

Forecast Horizons

Case	period
Base case	n/a
Discounting	8
Demand decline	n/a
Demand growth	7
Requirements change	7
Lumpy demand	13,7,9,5,12,7,13,8,6,13
Time-Varying costs	6,9,6,8,9,12,13,12,7

modern mainframe computers.

The problem data and results appear in Table 1. Most problems could be solved within a reasonable amount of time and with amazingly few unpruned states in any period. The main exception was the problem with rapid decline in demand. This case is not suprising — the same type of result occurs in single-product lot-sizing problems. This behavior is also suggested by the sensitivity to d of the upper bound on the number of unpruned states found in Theorem 3.

Average computational times for the case of lumpy demand tended to be larger than the other runs, but this is mitigated by the fact that forecast horizons were readily found as indicated in Table 2. Considering only results until the first forecast horizon was detected for each problem, the maximum number of states declines to 93 and and the average CPU time declines by 75%.

There was little difference between the discounting case and the base case in most of our measures. This is also suggested by the upper bound of Theorem 3 since the bound is sensitive to the ratio of the setup and holding costs of a period which is independent of discounting. However, with discounting a forecast horizon was detected in period 8 of the problem.

The case with time varying costs produced similar forecast horizons as the case of lumpy demand. However, the computation times and number of unpruned states per period were significantly less.

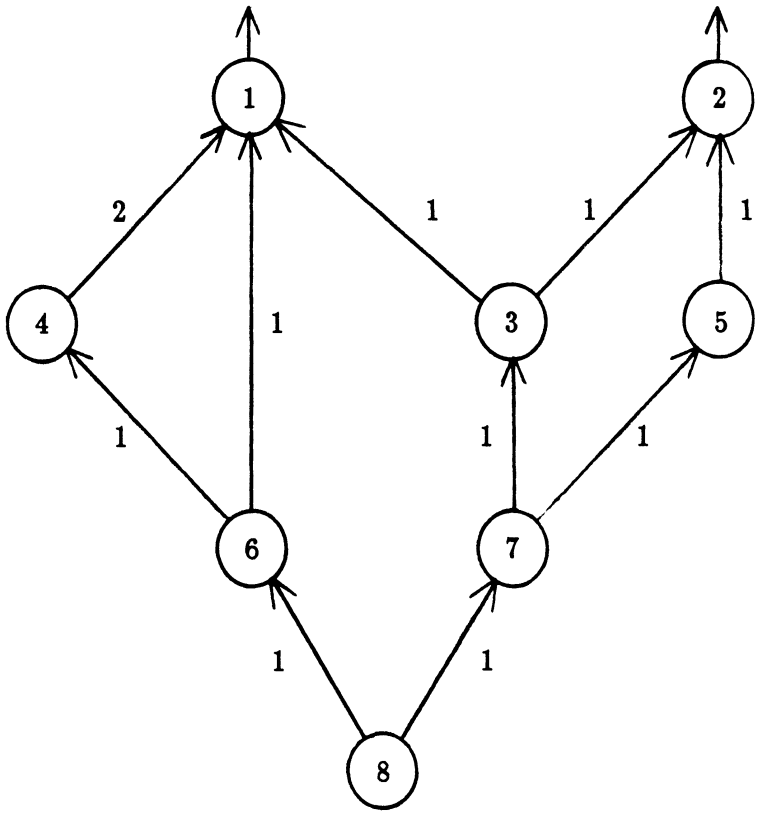

Figure 1. Network Representation of an Eight Item Product Structure.

In two cases, the base case and the case with declining demand, forecast horizons were not detected in 30 periods. This may indicate that (1) they do not exist, (2) a long study horizon is required, or (3) there are alternate optima which which cannot prune one another. We suspect that the latter was true for some of the problems which, after the initial period, had only 2 unpruned states in most periods.

It is possible that a more sophisticated pruning procedure using states in one period to prune states in another period, or relaxing the requirement that the pruning state match each subsequent setup out of the state to be pruned could lead to earlier detection of forecast horizons.

We also solved one eight-item problem which is illustrated in Figure 3. Costs were constant with K_i = 300, 400, 700, 600, 200, 50, 100, 1000 and h_i = 2.0, 1.0, 0.4, 1.0, 0.5, 0.5, 0.2, 0.05 for i = 1 through 8, respectively. We note that one of the echelon holding costs is negative (for item 1). Demand for item 1 was randomly generated using U[170,230] and similarly for item 2 using U[340,460]. All other items have no independent demand.

We used a horizon of 40 periods for this problem. The total CPU seconds on an Apollo DN330 was 1.7 hours, and the time to detect the first forecast horizon (in period 21) was 44 minutes. The mean, standard deviation, and maximum number of unpruned states were 112.9, 32.3, and 186, respectively.

A problem of this size and complexity may be near the practical limit using the current implementation. It is conceivable, however, that computation time could be reduced considerable with more effective pruning or by limiting consideration to nested policies.

6. Summary and Discussion

We have developed a forward dynamic programming procedure for general multi-stage lot-sizing problems which is shown to be computationally tractable for many problems. It was found that planning horizons were discovered (relatively quickly) in problems with time varying problem data, in particular in problems with "lumpy" demand. It is principally for such demand patterns that this procedure would provide the most benefit in real applications (versus models using constant demand approximations). Problems with most other demand and cost patterns, with the exception of rapidly declining demand, required little computing time and had very few unpruned states at any point in time.

Further research is needed to incorporate more extensive pruning, possibly with pruning of not only states, but also pruning certain decisions for individual items. Pruning using a limited "look ahead" to consider future data may also reduce the number of states. Research is also needed to incorporate capacity constraints. Nonetheless, the approach proposed in the paper appears to have promise for solving a wide array of lot-sizing problems.

A new area for research is the study of lot-sizing for problems where the product structure or a component requirement of an item is time dependent. Such problems are important, for example, when an engineering redesign of an item results in new requirements for component items or when an alternative sequence of operations is used to manufacture a finished product. These situations can be modelled by simply making the requirement coefficients, r_{ij}, period dependent. Doing this will not invalidate the decision process nor the pruning technique. Hence, the DP algorithm presented here, with obvious modifications,

can be used to study general uncapacitated multi-stage lot-sizing problems whose product structures and requirement coefficients are time dependent.

References

1. Blackburn, J. D. and H. Kunreuther (1974), "Planning Horizons for the Dynamic Lot Size Model with Backlogging," *Managemant Science 21(3),* 251-255.

2. Billington, J. B., McClain, J., O., and L. J. Thomas (1983) "Mathematical Programming Approaches to Capacity Constrained MRP Systems: Review, Formulation and Problem Reduction," *Management Science 29(10),* 1126-1141.

3. Maxwell, W. L., and J. A. Muckstadt (1985), "Establishing Consistent and Realistic Order Intervals in Production-Distrubution Systems," *Operations Research 33(6),* 1316-1341.

4. Wagner, H. M. and T. H. Whitin (1958), "Dynamic Version of the Economic Lot Size Model," *Management Science 5(1),* 89-96.

5. Zabel, E. (1964) "Some Generalizations of an Inventory Planning Horizon Theorem," *Management Science 10(3),* 465-471.

6. Zangwill, W. I. (1969) "A Backlogging Model and a Multi-Echelon Model of a Dynamic Economic Lot Size Production System - A Network Approach," *Management Science 15(9),* 506-527