A Dynamic Programming Algorithm for Dynamic Lot Size Models with Piecewise Linear Costs

HSIN-DER CHEN, DONALD W. HEARN, and CHUNG-YEE LEE
Industrial and Systems Engineering Department, University of Florida, Gainesville, Florida 32611, U.S.A.

(Received: 12 January 1993; accepted: 10 August 1993)

Abstract. We propose a new algorithm for dynamic lot size models (LSM) in which production and inventory cost functions are only assumed to be piecewise linear. In particular, there are no assumptions of convexity, concavity or monotonicity. Arbitrary capacities on both production and inventory may occur, and backlogging is allowed. Thus the algorithm addresses most variants of the LSM appearing in the literature. Computational experience shows it to be very effective on NP-hard versions of the problem. For example, 48 period capacitated problems with production costs defined by eight linear segments are solvable in less than 2.5 minutes of Vax 8600 cpu time.

Key words. Lot size models, dynamic programming.

1. Introduction

The single item dynamic lot size model (LSM) can be described as follows. For a finite time horizon T, demand for a single commodity of each production period is known. These demands may be satisfied by productions in the current or future periods, by inventories from previous periods, or by purchases from outside suppliers. There may be certain kinds of capacity restraints on production, inventories or purchases. We consider costs which are incurred by producing products, purchasing products, or holding inventories. Here the holding costs may occur because of either positive or negative inventory, with the latter representing backlogging. The LSM seeks a feasible production plan with the total cost as small as possible.

In this paper, we propose a dynamic programming algorithm for dynamic lot size models where the production and inventory cost functions are only assumed to be piecewise linear. In particular, there are no assumptions of convexity, concavity or monotonicity. Thus the algorithm addresses most variants of the LSM appearing in the literature.

The literature of the LSM originates with Wagner and Whitin (1958) who considered models with fixed setup costs and linear holding costs. There was no capacity constraint in their original model and backlogging was not allowed. They gave an $O(T^2)$ dynamic programming algorithm. Zangwill (1966) provided a network approach (minimal cost network flow problems) for the LSM with concave cost functions, and later included backlogging in this model.

Florian and Klein (1971) characterized some properties, which relate to *regeneration points*, of the optimal solutions for the LSM with concave cost

© 1994 *Kluwer Academic Publishers. Printed in the Netherlands.*

functions and with capacity on production. They proposed an $O(T^4)$ algorithm for the case of constant production capacities. Love (1973) gave an $O(T^3)$ algorithm for certain problems with piecewise concave cost functions and with capacity on both production and inventory. Jagannathan and Rao (1973) have similar results to Love's, however the production cost was neither concave nor convex and the inventory cost was linear in their model. Swoveland (1975) extended Jagannathan and Rao's results to the cases with piecewise linear production and holding costs. Extensions of Florian and Klein's results are given by Lambrecht and Vander Eecken (1977) for problems with arbitrary capacities and Lambert and Luss (1982) who studied the cases where capacities were multipliers of a constant value.

The basic capacitated LSM has received much attention because it is more tractable than other general models. In this model backlogging is not allowed, there are capacity constraints on production, holding costs are linear and production costs have a fixed-charged structure. Based on Bitran and Yanasse (1982), the notation for this problem is in a form of $\alpha/\beta/\gamma/\delta$ where α, β, γ and δ represent setup cost, holding cost, production cost and capacity type, respectively. The values of these parameters may be G, C, ND, NI and Z for arbitrary pattern, constant, non-decreasing, non-increasing and zero over production periods. When $\delta = \infty$, it represents an uncapacitated LSM. The capacitated LSM, even in many special cases, is NP-hard (Florian *et al.*, 1980; and Bitran and Yanasse, 1982). Several special cases with polynomial computational time have been discovered. Bitran and Yanasse (1982) showed that the time complexity of $NI/G/NI/ND$, $NI/G/NI/C$, $C/Z/C/G$ and $ND/Z/ND/NI$ are $O(T^4)$, $O(T^3)$, $O(T \log T)$ and $O(T)$, respectively. Chung and Lin (1988) improved the time complexity of $NI/G/NI/ND$ to $O(T^2)$. Baker *et al.* (1978) proposed a tree-search solution algorithm for the general problem $G/G/G/G$ and Florian *et al.* (1980) provided a discrete dynamic programming algorithm with computational complexity $O(D_T C_T)$ where D_T and C_T are cumulative demand and capacity, respectively. Kirca (1990) offered improvements to this discrete dynamic program. Chung *et al.* (1990) implemented a branch-and-bound algorithm for the problem $G/G/NI/G$.

In addition to the above literature, certain variations of the LSM have been treated by other authors. For example, Lippman (1969) considered the LSM with the decision variables being the number of trucks to deliver orders. This idea was extended in the works of Sethi and Chand (1981) and Chand and Sethi (1983) for the capacitated LSM. Multi-source versions of the capacitated LSM are treated by Lee and Zipkin (1989) who allowed make-or-buy decisions and gave an $O(T^5)$ algorithm for the case of equal production capacities. And Erenguc and Tufekci (1987) and Erenguc and Aksoy (1990) have implemented branch-and-bound algorithms for the capacitated LSM with piecewise linear production cost functions.

Of the above papers, only a few report computational results with cpu times. To indicate the state-of-the-art, we cite those and quote the cpu times of the most difficult problem in each: Kirca (1990) solved a $G/G/G/G$ problem with 18 periods in 328 seconds on a CORDATA-ATP-8-Q20; Chung *et al.* (1990) solved one of their $G/C/C/G$ problems with 96 periods in 103.47 seconds on a VAX 11/750; Erenguc and Tufekci (1987) solved a 10 period capacitated LSM with 4 segments in the production cost functions in 4.65 seconds on an IBM3081/D32, and Erenguc and Aksoy (1990) solved a similar problem with 12 periods and 2 segments in 0.284 seconds on an IBM3090/400.

In Section 4 we give extensive computational results, including Vax 8600 cpu times, for the method of this paper. From these, it can be estimated that our cpu times for the above four cases would be less than 0.08, 2.91, 0.34 and 0.09 seconds, respectively. The Vax 8600 is rated at 0.48 megaflops by Dongarra (1989) while the Vax 11/750, IBM 3081 and IBM3090 have ratings of 0.12, 2.1 and 16, respectively. (The CORDATA-ATP-8-Q20 is not rated.) Thus, even allowing for the different computers, we feel this study confirms the effectiveness of the method relative to those proposed thus far.

Our algorithm is motivated by the approach of Wagelmans, Van Hoesel and Kolen (1992) who have recently improved the complexity of the original Wagner–Whitin ($G/G/G/\infty$) model to $O(T \log T)$. (Independently, Federgruen and Tzur (1991) and Aggarwal and Park (1990) have obtained the same complexity with different approaches.) It also represents a significant generalization and extension of our work on the capacitated LSM in Chen, Hearn and Lee (1994).

The algorithm results from a *continuous* version of the standard dynamic programming model. The state variable, i.e., the cumulative production level, is treated as continuous. Under the assumption that the cost functions (production and inventory) are piecewise linear, we develop an efficient method to update the optimal value function. While the computational effort depends on the number of segments in the optimal value function, which can increase exponentially with T, extensive testing has shown the method to be very effective over a wide range of problem parameters.

2. Notation, Formulation and Model Variants

We need the following notation (for each period t) to describe the generic single item dynamic lot size model.

d_t = demand
x_t = production level
D_t = cumulative demand, that is, $D_t = \Sigma_{i=1}^{t} d_i$
X_t = cumulative production, that is, $X_t = \Sigma_{i=1}^{t} x_i$
$\mathscr{X}_t$ = the set of feasible production levels
$\mathscr{R}_t$ = the set of feasible cumulative production levels
$P_t(x_t)$ = cost function for producing x_t units
$H_t(X_t)$ = cost function for carrying $X_t - D_t$ units inventory at the end of

period t

Without loss of generality, we assume that $X_0 = 0$, $D_0 = 0$ and $X_T = D_T$. The generic single item dynamic lot size model can be formulated as follows.

$$(P1)\ Z^* = \min_{(x_1,\dots,x_T)} \sum_{t=1}^{T} (P_t(x_t) + H_t(X_t))$$

such that

$$\begin{aligned}
&X_0 = 0\,,\\
&X_T = D_T\,,\\
&X_t = X_{t-1} + x_t\,, \quad t = 1, \dots, T\\
&x_t \in \mathscr{X}_t\,, \quad t = 1, \dots, T\\
&X_t \in \mathscr{R}_t\,, \quad t = 1, \dots, T\,.
\end{aligned}$$

Note that the cost functions, $P_t(x_t)$ and $H_t(X_t)$, as well as the feasible sets, $\mathscr{X}_t$ and $\mathscr{R}_t$, in the above model are not specified. By specifying these functions or sets, we can define variants of the single item dynamic lot size model. We need the following additional notation for these variants.

c_t = capacity on production in period t.
w_t = capacity on inventory in period t.
C_t = cumulative capacity on production, that is, $C_t = \Sigma_{i=1}^{t} c_i$.
K_t = fixed set-up cost in period t.
p_t = unit production cost in period t.
β_t = unit purchasing cost in period t.
h_t = unit holding cost in period t.
b_t = unit backlogging cost in period t.
$\delta(x_t) = 0$ if $x_t = 0$, or 1 if $x_t > 0$.

(M1) THE WAGNER–WHITIN MODEL

In the Wagner–Whitin model, the unit production cost is constant ($p_t = p$), backlogging is not allowed, and there is no capacity restraint on production or inventory. Thus we define

$$\begin{aligned}
&P_t(x_t) = K_t\delta(x_t) + px_t\\
&H_t(X_t) = h_t(X_t - D_t)\\
&\mathscr{X}_t = [0, \infty)\\
&\mathscr{R}_t = [D_t, D_T]
\end{aligned}$$

(M2) THE UNCAPACITATED DYNAMIC LOT SIZE MODEL

The uncapacitated dynamic lot size model is similar to the Wagner–Whitin model except it has time varying unit production costs. That is,

$$P_t(x_t) = K_t\delta(x_t) + p_t x_t$$

(M3) THE DYNAMIC LOT SIZE MODEL WITH CAPACITY ON PRODUCTION

The dynamic lot size model with capacity on production is similar to (M2) except

$$\mathscr{X}_t = [0, c_t]$$
$$\mathscr{R}_t = [D_t, \min\{D_T, C_t\}]$$

(M4) THE DYNAMIC LOT SIZE MODEL WITH CAPACITY ON INVENTORY

The dynamic lot size model with capacity on inventory is similar to (M2) except

$$\mathscr{R}_t = [D_t, \min\{D_T, w_t + D_t\}]$$

(Note that (M3) and (M4) can be combined to have capacities on both production and inventory.)

(M5) THE DYNAMIC LOT SIZE MODEL WITH BACKLOGGING

In models with backlogging, the costs of carrying negative inventories are included, and cumulative production level, X_t, can be less than the cumulative demand in the previous period. For example, (M4) with backlogging is obtained with the following definitions:

$$P_t(x_t) = K_t\delta(x_t) + p_t x_t$$
$$H_t(X_t) = \begin{cases} b_t(D_t - X_t) & \text{if } X_t < D_t \\ h_t(X_t - D_t) & \text{if } X_t \geqslant D_t \end{cases}$$
$$\mathscr{X}_t = [0, \infty)$$
$$\mathscr{R}_t = [0, \min\{D_T, w_t + D_t\}]$$

(M6) THE MULTI-SOURCE DYNAMIC LOT SIZE MODEL

The multi-source model means that there is more than one way to satisfy demands. For example, we may produce in-house or buy from outside suppliers. Or we may have several production lines with independent setups or different production costs.

The difference between the models with single source and those with multiple sources is only in the production cost function. For instance, in the make-or-buy cases, suppose that the in-house capacity is c_t in period t, and let p_t, β_t be the unit production cost and the unit purchasing cost, respectively. There is a fixed charge for each case, say, K_t is the setup cost for production and α_t is the setup cost for purchasing in period t. Also, β_t is assumed to be greater than p_t, and there exists e_t, a break-even point between producing and purchasing, that is, $K_t + p_t e_t = \alpha_t + \beta_t e_t$ and $e_t < c_t$. Then the production cost function of this model is

$$P_t(x_t) = \begin{cases} \alpha_t \delta(x_t) + \beta_t x_t & \text{if } 0 \leqslant x_t \leqslant e_t \\ K_t + p_t x_t & \text{if } e_t < x_t \leqslant c_t \\ K_t + p_t c_t + \alpha_t + \beta_t (x_t - c_t) & \text{if } c_t < x_t \,. \end{cases}$$

Notice that above function is not necessarily convex or concave.

(M7) THE MULTI-WAREHOUSE DYNAMIC LOT SIZE MODEL

In this model, the inventory is held in several warehouses. Each warehouse has its own capacity, and a fixed charge must be paid if it is used. Very general versions of this model can be constructed, but in the interest of brevity, we give one simple example. Suppose that there are n identical warehouses with capacity w_t and setup cost W_t in period t. Then the multi-warehouse dynamic lot size model has the following holding cost function.

$$H_t(X_t) = \begin{cases} 0 & \text{if } X_t - D_t = 0 \\ iW_t + h_t(X_t - D_t) & \text{if } (i-1)w_t < X_t - D_t \leqslant iw_t \,, \\ & i = 1, \ldots, n \end{cases}$$

Notice that above function is also not necessarily convex or concave.

3. A Continuous Dynamic Programming Approach

Let $F_t(X_t)$ be the optimal value function of (P1) considering only the first t periods with X_t as the final cumulative production level, that is,

$$F_t(X_t) = \min_{(x_1, \ldots, x_t)} \sum_{k=1}^{t} (P_k(x_k) + H_k(X_k))$$

such that

$$\begin{aligned} & X_0 = 0 \\ & X_k = X_{k-1} + x_k \,, \quad k = 1, \ldots, t \\ & x_k \in \mathscr{X}_k \,, \quad k = 1, \ldots, t \\ & X_k \in \mathscr{R}_k \,, \quad k = 1, \ldots, t-1 \,. \end{aligned}$$

And define

$$F_0(X_0) = \begin{cases} 0 & \text{if } X_0 = 0 \\ \infty & \text{otherwise} . \end{cases}$$

Then (P1) can be solved by dynamic programming with the following recursive equations:

$$\begin{aligned} F_t(X_t) = H_t(X_t) + \min\{F_{t-1}(X_{t-1}) + P_t(x_t) \mid X_t = X_{t-1} + x_t , \\ X_{t-1} \in \mathcal{R}_{t-1}, x_t \in \mathcal{X}_t\} , \quad t = 1, \ldots, T \end{aligned} \tag{1}$$

and

$$Z^* = F_T(D_T) .$$

The conventional approach for (1) assumes that the state variable X_t and the control variable x_t are discrete. Thus, it yields a pseudo-polynomial time algorithm with computational complexity $O(D_T\bar{X})$ where $\bar{X}$ is the number of total elements in $\mathcal{X}_t$ over all t. For instance, the complexity is $O(D_TC_T)$ for the problems in (M3). Since D_T and $\bar{X}$ can be very large numbers, researchers usually see this approach only as a last resort.

Our approach is different from the conventional one by treating X_t and x_t as continuous variables. Since X_t is continuous, there are an infinite number of states in our approach. In order to resolve the difficulty incurred by infinite states, the following assumption is needed:

ASSUMPTION. *$P_t(x_t)$ and $H_t(X_t)$ are piecewise linear functions with finitely many segments for any t.*

Since all given cost functions are piecewise linear, it is reasonable to expect that $F_t(X_t)$ is also a piecewise linear function for any t. The following theorem will be proven later.

THEOREM 1. *$F_t(X_t)$ is a piecewise linear function with finitely many pieces for any t.*

Since $F_{t-1}(X_{t-1})$ is piecewise linear, its domain can be split into several intervals, say

$$\mathcal{R}_{t-1} = \bigcup_i \mathcal{R}_{t-1}^{(i)} ,$$

such that, in each interval $\mathcal{R}_{t-1}^{(i)}$, $F_{t-1}(X_{t-1})$ is only a line segment. In a similar way, the domain of $P_t(x_t)$ can also be split into several intervals, that is,

$$\mathcal{X}_t = \bigcup_j \mathcal{X}_t^{(j)} ,$$

such that, $P_t(x_t)$ is only a single line segment in each interval $\mathscr{X}_t^{(j)}$.

Now, consider minimizing in the right hand side of (1) segment by segment instead of point by point. Define

$$F_t^{(i,j)}(X_t) = \min\{F_{t-1}(X_{t-1}) + P_t(x_t) \mid X_t = X_{t-1} + x_t, \\ X_{t-1} \in \mathscr{R}_{t-1}^{(i)}, x_t \in \mathscr{X}_t^{(j)}\} \quad \forall t, i, j . \tag{2}$$

Therefore, the recursive equations in (1) can be rewritten as

$$F_t(X_t) = H_t(X_t) + \min\{F_t^{(i,j)}(X_t) \mid \forall i, j\} \quad t = 1, \dots, T . \tag{3}$$

The function value of $F_t^{(i,j)}(X_t)$ is infinite if X_t is out of its domain.

In the following sections we will show $F_t^{(i,j)}(X_t)$ for any i and j is a piecewise linear function, and it can be written in closed form.

Given an $\mathscr{R}_{t-1}^{(i)}$ and an $\mathscr{X}_t^{(j)}$, suppose that

$$\mathscr{R}_{t-1}^{(i)} = [a_1^{(i)}, a_2^{(i)}] ,$$
$$\mathscr{X}_t^{(j)} = [b_1^{(j)}, b_2^{(j)}] ,$$

and

$$F_{t-1}(X_{t-1}) = R^{(i)} + r^{(i)} X_{t-1} , \quad \text{for } X_{t-1} \in \mathscr{R}_{t-1}^{(i)} ,$$
$$P_t(x_t) = P^{(j)} + p^{(j)} x_t \quad \text{for } x_t \in \mathscr{X}_t^{(j)} .$$

Then for each i, j, (2) becomes

$$F_t^{(i,j)}(X_t) = \min\{R^{(i)} + r^{(i)} X_{t-1} + P^{(j)} + p^{(j)} x_t \mid \\ X_t = X_{t-1} + x_t, a_1^{(i)} \leq X_{t-1} \leq a_2^{(i)}, b_1^{(j)} \leq x_t \leq b_2^{(j)}\} . \tag{4}$$

We use a continuous version of the *reaching* concept (Denardo, 1982) to evaluate the above recursive equation, and say that X_t can be reached from X_{t-1} if $b_1^{(j)} \leq X_t - X_{t-1} \leq b_2^{(j)}$.

Consider the minimization in (4). Given an X_t, after substituting $X_{t-1} = X_t - x_t$ or $x_t = X_t - X_{t-1}$, we are to minimize

$$R^{(i)} - (r^{(i)} - p^{(j)}) x_t + P^{(j)} + r^{(i)} X_t$$

or

$$R^{(i)} + (r^{(i)} - p^{(j)}) X_{t-1} + P^{(j)} + p^{(j)} X_t$$

over all possible production levels x_t, or over all X_{t-1} from which X_t can be reached. So, if $r^{(i)} \leq p^{(j)}$, it is optimal to make x_t as small as possible, or to make X_{t-1} as large as possible. That is, $x_t = b_1^{(j)}$ when $X_t \in [a_1^{(i)} + b_1^{(j)}, a_2^{(i)} + b_1^{(j)}]$; and

$X_{t-1}=a_2^{(i)}$ when $X_t \in [a_2^{(i)}+b_1^{(j)}, a_2^{(i)}+b_2^{(j)}]$. On the other hand, in case of $r^{(i)}>p^{(j)}$, x_t should be as large as possible and X_{t-1} as small as possible. That is, $X_{t-1}=a_1^{(i)}$ when $X_t \in [a_1^{(i)}+b_1^{(j)}, a_1^{(i)}+b_2^{(j)}]$; and $x_t=b_2^{(j)}$ when $X_t \in [a_1^{(i)}+b_2^{(j)}, a_2^{(i)}+b_2^{(j)}]$. The above discussion can be summarized in the following algebraic expressions:
If $r^{(i)} \leqslant p^{(j)}$, then

$$F_t^{(i,j)}(X_t)=\begin{cases} R^{(i)}-(r^{(i)}-p^{(j)})b_1^{(j)}+P^{(j)}+r^{(i)}X_t\,, \\ \text{for } a_1^{(i)}+b_1^{(j)} \leqslant X_t \leqslant a_2^{(i)}+b_1^{(j)} \\ R^{(i)}+(r^{(i)}-p^{(j)})a_2^{(i)}+P^{(j)}+p^{(j)}X_t\,, \\ \text{for } a_2^{(i)}+b_1^{(j)} < X_t \leqslant a_2^{(i)}+b_2^{(j)}\,, \end{cases}$$

otherwise,

$$F_t^{(i,j)}(X_t)=\begin{cases} R^{(i)}+(r^{(i)}-p^{(j)})a_1^{(i)}+P^{(j)}+p^{(j)}X_t\,, \\ \text{for } a_1^{(i)}+b_1^{(j)} \leqslant X_t \leqslant a_1^{(i)}+b_2^{(j)} \\ R^{(i)}-(r^{(i)}-p^{(j)})b_2^{(j)}+P^{(j)}+r^{(i)}X_t\,, \\ \text{for } a_1^{(i)}+b_2^{(j)} < X_t \leqslant a_2^{(i)}+b_2^{(j)}\,, \end{cases}$$

or in another form, if $r^{(i)} \leqslant p^{(j)}$, then

$$F_t^{(i,j)}(X_t)=\begin{cases} F_{t-1}(X_t-b_1^{(j)})+P_t(b_1^{(j)})\,, \\ \text{for } a_1^{(i)}+b_1^{(j)} \leqslant X_t \leqslant a_2^{(i)}+b_1^{(j)} \\ F_{t-1}(a_2^{(i)})+P_t(X_t-a_2^{(i)})\,, \\ \text{for } a_2^{(i)}+b_1^{(j)} < X_t \leqslant a_2^{(i)}+b_2^{(j)}\,, \end{cases} \tag{5}$$

otherwise,

$$F_t^{(i,j)}(X_t)=\begin{cases} F_{t-1}(a_1^{(i)})+P_t(X_t-a_1^{(i)})\,, \\ \text{for } a_1^{(i)}+b_1^{(j)} \leqslant X_t \leqslant a_1^{(i)}+b_2^{(j)} \\ F_{t-1}(X_t-b_2^{(j)})+P_t(b_2^{(j)})\,, \\ \text{for } a_1^{(i)}+b_2^{(j)} < X_t \leqslant a_2^{(i)}+b_2^{(j)}\,. \end{cases} \tag{6}$$

We interpret the above segments as follows. The first segment of (5) and the second segment of (6) are created by fixing the production levels at $b_1^{(j)}$ and $b_2^{(j)}$, respectively. On the other hand, the second segment of (5) and the first segment of (6) are created by fixing the cumulative production level in period $t-1$, X_{t-1}, at $a_2^{(i)}$ and $a_1^{(i)}$, respectively.

We have shown that $F_t^{(i,j)}(X_t)$ in (3) for any i and j is a piecewise linear function with two segments. Therefore $F_t(X_t)$ in (3) can be computed in two steps: first, evaluating the minimum of $F_t^{(i,j)}(X_t)$ over all i, j, then adding this result to $H_t(X_t)$.

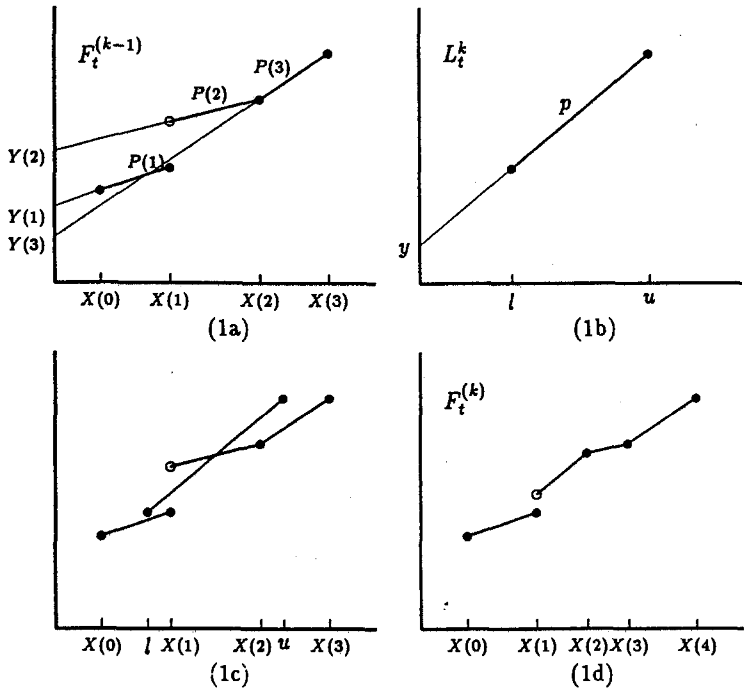

Fig. 1.

The first step is illustrated in Figure 1. Let there be in total s line segments from all of the $F_t^{(i,j)}(X_t)$ over all i and j. Denote L_t^k as the kth line segment for $1 \leq k \leq s$ (taken in any order). Any L_t^k is uniquely defined by (l, u, p, y) = (lower bound, upper bound, slope, intercept); see Figure 1(b).

Starting with $F_t^{(1)}(X_t) = L_t^1$ for $X_t \in [l, u] \cap \mathcal{R}_t$, the function $F_t^{(s)}(X_t)$ for $X_t \in \mathcal{R}_t$ can be obtained by a *segment by segment* update procedure. In Figure (1a), $F_t^{(k-1)}$ represents F_t after $(k-1)$ updates. It is defined by its breakpoints $X(0)$, $X(1)$, $X(2)$, $X(3)$ and by its segments which have slopes $P(1)$, $P(2)$, $P(3)$ and intercepts $Y(1)$, $Y(2)$, $Y(3)$. $F_t^{(k-1)}$ and L_t^k are both included in Figure (1c), then $F_t^{(k)}$ is obtained by moving through the breakpoints, including l and u, and comparing the function values with those of the segment. The result is given in Figure (1d).

The second step is to sum two piecewise linear functions, $H_t(X_t)$ and $F_t^{(s)}(X_t)$. After sorting all the breakpoints of these two functions, there is only a single segment for each function between two such consecutive breakpoints. Therefore, the summation of these two functions can be calculated easily to give $F_t(X_t)$.

For a detailed discussion of evaluating lower envelopes and recovering an optimal solution for the model (M3), see Chen *et al.* (1994), which contains a pseudocode and a numerical example.

The computational effort of our approach is a function of the time horizon, T, the number of segments in the cost functions and the number of segments in the optimal value functions. Although the number of segments may increase at an exponential rate with T theoretically, it need not be more than D_T in practice since we can always delete segments between two consecutive integer values of X_t without losing the optimal solution. So, our approach can not be worse than the conventional discrete approach for problems with integer data.

We conclude this section by proving Theorem 1.

Proof of Theorem 1. The theorem will be proven by mathematical induction. When $t = 0$, $F_0(X_0)$ is a piecewise linear function with one segment (the origin) by definition. Now assume that $F_{t-1}(X_{t-1})$ is a piecewise linear function with finite segments for $t \geq 1$. From (3), formulas (5), (6) and the assumption that $H_t(X_t)$ is a piecewise linear function with finite segments, it can be seen that $F_t(X_t)$ is also a piecewise linear function with finite segments. ■

4. Computational Experience

In this section we examine the efficiency of our approach on the capacitated LSM with multiple piece production cost functions. Suppose there are M pieces in the production cost function of each period. There is a fixed charge (setup cost), a slope (unit production cost), and a capacity (length of such piece) for each piece. Linear holding costs are considered in this model and no backlogging is allowed.

We modified the problem pattern in Baker *et al.* (1978) to randomly create test problems.

The demand is given in a pattern as follows.

$$d_t = 200 + \sigma z_t + a \sin\left[\frac{2\pi}{b}(t + b/4)\right] \tag{7}$$

where

σ = standard error of demand,
z_t = i.i.d. standard normal random deviates,
a = amplitude of the seasonality component,
b = length of seasonal cycle in periods.

Five problems were created in each of the following four combinations: (1) $\sigma = 67$, $a = 0$, (2) $\sigma = 237$, $a = 0$, (3) $\sigma = 67$, $a = 125$, $b = T$ and (4) $\sigma = 67$, $a = 125$, $b = 12$.

We tested the problems with $T \in \{12, 24, 48, 96,\}$ and $M \in \{1, 2, 4, 8\}$.The unit holding cost, h_t for each t, is uniformly distributed between 0.5 and 1.5. The unit production cost, $p_t^{(j)}$ for each period t and each piece j, is uniformly distributed between 10 and 30. The capacity for each period and each piece is uniformly distributed in $(0.5C/M,\ 1.5C/M)$ where $C \in \{400, 800, 1200, 1600\}$. These average total capacities C satisfy (average) demand of 2, 4, 6 and 8 periods, respectively. The corresponding setup costs are $K = 400$, 1600, 3600, 6400 which

Table I. Maximal segments in $F_t(X_t)$ over time horizon

T	12	24	48	96
2^T	4.1×10^3	1.7×10^7	2.8×10^{14}	7.9×10^{28}
actual number	607	1704	2988	6102

come from the continuous EOQ (economic order quantity) model. The setup cost for each period and each piece is uniformly distributed in $(0.5K/M,\ 1.5K/M)$.

A total of 5120 feasible problems were created and tested. The program of our algorithm is coded in Fortran and run on a VAX 8600.

We are interested in both cpu time and the number of segments of the optimal value function. The results are listed in two tables: one for cpu time and the other for the maximal number of segments. Since these tables are large, they are placed in the Appendix. Each cell in these tables represents 20 different demands of the corresponding category. Both average and the maximal (in the parentheses) values of the 20 problems are listed. Note that the tables show that these computational results are very stable since the maximal values are within twice the average values in almost every case.

As expected, the number of segments of the optimal value function is very small. Even though this number can be more than 2^T for a T period problem, it is only a fraction of its theoretical value as shown in Table I. The fact that the number of segments is small ensures that problems as large as 96 periods and 8 segments in the production cost function can be solved in less than 100 K bytes of memory since about 10 bytes are required for each.

The results in the Appendix also show that the problems with smaller setup costs are easier for our algorithm than those with larger setup costs. On the other hand, the problems with less capacity are harder than those with more capacity. Out of the 16 possible combinations of K and C, the case with $K = 6400$ and $C = 400$ is the hardest. Table II lists the average cpu time of this hardest case. In the case of $T = 96$ and $M = 1$, it takes only 1.97 second on average to find the optimal solution. We feel these results demonstrate the effectiveness of the method relative to the prior literature. For example, the largest problem previously solved to optimality, with $M = 1$, only has $T = 18$ (Kirca, 1990).

Finally, the increases in the cpu times in Table II are given in Table III and IV as a function of T and M, respectively.

In these tables an entry of 8 indicates a cubic growth rate. The entries in the

Table II. Average cpu time* ($K = 6400$, $C = 400$)

	$T = 12$	$T = 24$	$T = 48$	$T = 96$
$M = 1$	0.01	0.06	0.40	1.97
$M = 2$	0.05	0.27	1.84	12.49
$M = 4$	0.19	1.59	9.95	67.76
$M = 8$	1.38	10.89	75.69	512.44

*seconds on Vax 8600.

Table III. Increases in cpu time with respect to T

T	$12 \to 24$	$24 \to 48$	$48 \to 96$
$M = 1$	6.0	6.7	4.9
$M = 2$	5.4	6.8	6.8
$M = 4$	8.4	6.3	6.8
$M = 8$	7.9	7.0	6.8

Table IV. Increases in cpu time with respect to M

M	$T = 12$	$T = 24$	$T = 48$	$T = 96$
$1 \to 2$	5.0	4.5	4.6	6.3
$2 \to 4$	3.8	5.9	5.4	5.4
$4 \to 8$	7.3	6.8	7.6	7.6

two tables range from 3.8 to 8.4, demonstrating that the growth rates range from 1.93 to 3.07. Thus it is reasonable to claim that, for data similar to ours, the cpu time increases at a rate which is approximately cubic in either T or M.

5. Conclusions

In summary, the method proposed in this paper applies to many NP-hard versions of the dynamic lot size model and the computational results show that it is highly effective. Computational effort only grows at a cubic rate in either T or M for the problems tested. This means that the algorithm can be used as a subproblem in decomposition approaches to multi-item problems – this is one extension we are currently exploring.

The computer code used in our experiments is available for research purposes from the authors. The distribution disk includes a code for generation of test data.

Acknowledgement

The authors express appreciation to the referees whose comments and suggestions improved the paper.

Appendix

Table A.1. Cpu time (seconds on a VAX 8600)

K	C	$T = 12$				$T = 24$			
		$M = 1$	$M = 2$	$M = 4$	$M = 8$	$M = 1$	$M = 2$	$M = 4$	$M = 8$
400	400	0.01[a]	0.02	0.09	0.33	0.03	0.11	0.39	1.68
		(0.01)[b]	(0.04)	(0.12)	(0.49)	(0.05)	(0.14)	(0.54)	(2.37)
	800	0.01	0.01	0.05	0.17	0.02	0.06	0.20	0.83
		(0.01)	(0.03)	(0.07)	(0.23)	(0.03)	(0.07)	(0.24)	(1.05)
	1200	0.00	0.01	0.03	0.11	0.01	0.04	0.14	0.51
		(0.01)	(0.02)	(0.05)	(0.16)	(0.02)	(0.06)	(0.19)	(0.67)
	1600	0.00	0.01	0.02	0.08	0.01	0.03	0.10	0.36

		(0.01)	(0.02)	(0.04)	(0.10)	(0.01)	(0.04)	(0.14)	(0.46)
1600	400	0.01	0.03	0.13	0.68	0.04	0.18	0.83	4.66
		(0.02)	(0.05)	(0.25)	(1.16)	(0.06)	(0.26)	(1.19)	(7.03)
	800	0.00	0.02	0.07	0.32	0.02	0.09	0.42	1.96
		(0.01)	(0.04)	(0.11)	(0.48)	(0.04)	(0.13)	(0.67)	(2.52)
	1200	0.00	0.01	0.05	0.18	0.01	0.05	0.22	1.04
		(0.02)	(0.02)	(0.06)	(0.27)	(0.02)	(0.08)	(0.31)	(1.66)
	1600	0.01	0.01	0.03	0.12	0.01	0.04	0.13	0.62
		(0.01)	(0.02)	(0.06)	(0.17)	(0.02)	(0.06)	(0.18)	(0.81)
3600	400	0.01	0.04	0.20	0.90	0.05	0.23	1.03	8.29
		(0.02)	(0.06)	(0.33)	(1.51)	(0.08)	(0.36)	(1.84)	(18.87)
	800	0.01	0.02	0.09	0.37	0.03	0.13	0.60	3.64
		(0.02)	(0.04)	(0.18)	(0.60)	(0.05)	(0.20)	(0.93)	(6.37)
	1200	0.01	0.02	0.06	0.22	0.02	0.07	0.35	1.46
		(0.02)	(0.03)	(0.10)	(0.30)	(0.03)	(0.11)	(0.49)	(2.00)
	1600	0.00	0.01	0.04	0.15	0.01	0.05	0.20	0.90
		(0.01)	(0.02)	(0.07)	(0.24)	(0.03)	(0.08)	(0.29)	(1.29)
6400	400	0.01	0.05	0.19	1.38	0.06	0.27	1.59	10.89
		(0.02)	(0.09)	(0.34)	(3.93)	(0.08)	(0.44)	(2.71)	(39.42)
	800	0.01	0.03	0.10	0.54	0.04	0.16	0.80	4.10
		(0.02	(0.04)	(0.16)	(1.11)	(0.06)	(0.27)	(1.41)	(5.97)
	1200	0.01	0.01	0.06	0.29	0.02	0.09	0.42	2.06
		(0.02)	(0.03)	(0.10)	(0.43)	(0.04)	(0.17)	(0.70)	(3.62)
	1600	0.00	0.01	0.04	0.16	0.02	0.06	0.28	1.29
		(0.01)	(0.02)	(0.07)	(0.25)	(0.03)	(0.08)	(0.43)	(1.98)

a: average; b: worst case.

Table A1. Cpu time (seconds on a VAX 8600) (*continued*)

K	*C*	$T=48$				$T=96$			
		$M=1$	$M=2$	$M=4$	$M=8$	$M=1$	$M=2$	$M=4$	$M=8$
400	400	0.11[a]	0.41	1.64	7.15	0.42	1.64	7.26	32.72
		(0.13)[b]	(0.51)	(1.94)	(8.94)	(0.55)	(1.99)	(8.11)	(38.85)
	800	0.07	0.25	0.89	3.73	0.29	1.05	3.97	16.47
		(0.10)	(0.35)	(1.07)	(4.29)	(0.33)	(1.22)	(4.86)	(20.30)
	1200	0.04	0.16	0.59	2.23	0.18	0.68	2.59	9.92
		(0.09)	(0.21)	(0.78)	(2.85)	(0.22)	(0.90)	(3.05)	(12.87)
	1600	0.04	0.12	0.43	1.67	0.13	0.47	1.80	6.95
		(0.07)	(0.16)	(0.52)	(1.96)	(0.18)	(0.55)	(2.56)	(7.93)
1600	400	0.19	0.82	4.38	28.27	0.79	3.60	20.15	128.24
		(0.31)	(1.01)	(6.16)	(44.79)	(1.07)	(4.92)	(26.19)	(171.49)
	800	0.11	0.41	1.73	9.47	0.44	1.72	8.31	46.71
		(0.16)	(0.55)	(2.17)	(13.70)	(0.51)	(2.28)	(10.23)	(63.35)
	1200	0.06	0.24	0.99	4.45	0.26	1.04	4.22	22.84
		(0.09)	(0.29)	(1.17)	(5.88)	(0.34)	(1.23)	(5.01)	(33.48)
	1600	0.05	0.17	0.67	2.95	0.19	0.70	2.94	13.93
		(0.08)	(0.23)	(0.80)	(3.92)	(0.25)	(0.84)	(3.51)	(18.09)
3600	400	0.28	1.24	8.38	51.62	1.37	7.18	40.48	334.90
		(0.41)	(1.70)	(13.07)	(102.52)	(1.89)	(9.51)	(58.75)	(559.83)
	800	0.15	0.65	3.52	19.91	0.65	3.04	16.96	119.41
		(0.23)	(0.90)	(5.76)	(29.38)	(0.78)	(3.63)	(26.28)	(156.69)
	1200	0.08	0.36	1.61	10.90	0.37	1.64	7.74	51.35
		(0.13)	(0.48)	(1.96)	(14.73)	(0.47)	(2.37)	(9.33)	(63.94)
	1600	0.05	0.23	0.99	5.35	0.25	1.02	4.67	28.29
		(0.07)	(0.34)	(1.26)	(7.80)	(0.31)	(1.24)	(5.54)	(36.93)
6400	400	0.40	1.84	9.95	75.69	1.97	12.49	67.76	512.44
		(0.65)	(2.54)	(23.88)	(145.98)	(2.91)	(27.91)	(94.17)	(766.11)
	800	0.19	0.82	4.98	39.08	0.93	4.88	31.11	210.95
		(0.27)	(1.07)	(7.14)	(59.82)	(1.19)	(7.73)	(45.10)	(281.85)
	1200	0.10	0.48	2.25	18.73	0.50	2.42	12.43	88.28
		(0.16)	(0.66)	(2.84)	(38.88)	(0.69)	(3.14)	(17.62)	(112.00)
	1600	0.07	0.30	1.48	8.68	0.30	1.45	7.76	50.86
		(0.10)	(0.47)	(2.34)	(11.49)	(0.40)	(1.87)	(9.50)	(62.02)

a: average; b: worst case.

Table A2. Maximal segments of $F_t(X_t)$ over time horizon

K	C	$T = 12$				$T = 24$			
		$M = 1$	$M = 2$	$M = 4$	$M = 8$	$M = 1$	$M = 2$	$M = 4$	$M = 8$
400	400	12.4[a]	26.9	55.1	102.2	22.6	53.9	106.4	216.3
		(22)[b]	(41)	(79)	(133)	(34)	(74)	(144)	(271)
	800	8.4	16.1	31.0	56.5	15.9	30.5	59.3	188.9
		(13)	(21)	(41)	(85)	(23)	(43)	(74)	(144)
	1200	6.4	11.9	21.7	38.5	11.1	22.0	43.8	79.8
		(9)	(17)	(29)	(55)	(14)	(27)	(66)	(103)
	1600	5.6	8.8	15.6	28.3	10.1	17.1	32.2	58.5
		(8)	(12)	(22)	(35)	(14)	(21)	(42)	(77)
1600	400	15.8	38.1	81.3	178.8	33.5	83.2	203.9	429.8
		(21)	(63)	(124)	(273)	(59)	(121)	(295)	(565)
	800	8.8	19.0	40.8	94.7	21.3	45.0	115.6	217.8
		(11)	(26)	(62)	(131)	(35)	(74)	(176)	(300)
	1200	7.7	13.8	30.0	56.7	9.6	30.1	65.1	136.7
		(11)	(19)	(44)	(86)	(20)	(46)	(92)	(220)
	1600	5.9	10.6	21.1	37.2	11.3	23.0	45.7	92.0
		(8)	(15)	(40)	(53)	(17)	(34)	(73)	(117)
3600	400	16.7	41.0	114.8	217.0	43.7	110.4	244.9	608.7
		(25)	(74)	(185)	(308)	(76)	(191)	(401)	(1000)
	800	10.4	22.7	56.0	101.3	26.5	61.0	151.6	342.6
		(15)	(36)	(107)	(153)	(42)	(133)	(191)	(575)
	1200	7.5	16.6	34.3	69.0	16.6	37.0	98.6	178.1
		(14)	(28)	(60)	(95)	(27)	(54)	(141)	(251)
	1600	5.9	11.6	25.1	49.5	12.4	28.8	60.3	122.9
		(9)	(24)	(53)	(71)	(18)	(42)	(79)	(183)
6400	400	17.9	48.3	107.5	293.6	49.8	129.3	337.1	714.2
		(28)	(79)	(177)	(607)	(83)	(207)	(580)	(1704)
	800	10.4	24.0	58.3	134.7	28.9	73.3	178.5	350.3
		(16)	(40)	(90)	(229)	(54)	(109)	(288)	(436)
	1200	7.5	16.4	35.7	90.1	18.3	42.7	108.5	222.1
		(12)	(25)	(56)	(186)	(28)	(81)	(167)	(343)
	1600	5.7	11.9	26.3	53.8	14.1	27.6	77.0	159.3
		(8)	(18)	(45)	(86)	(21)	(36)	(127)	(256)

a: average; b: worst case.

Table A2. Maximal segments of $F_t(X_t)$ over time horizon (*continued*)

K	C	$T = 48$				$T = 96$			
		$M = 1$	$M = 2$	$M = 4$	$M = 8$	$M = 1$	$M = 2$	$M = 4$	$M = 8$
400	400	40.5[a]	98.2	212.1	405.6	83.5	190.8	428.8	841.4
		(57)[b]	(129)	(253)	(513)	(117)	(234)	(491)	(937)
	800	28.1	64.2	118.9	235.1	57.0	118.9	244.8	462.0
		(35)	(107)	(156)	(295)	(74)	(142)	(275)	(571)
	1200	20.4	43.0	83.9	158.3	37.0	81.8	168.4	303.8
		(31)	(57)	(111)	(196)	(47)	(99)	(193)	(376)
	1600	16.7	32.5	63.7	128.0	28.8	60.7	124.7	228.1
		(23)	(50)	(83)	(163)	(36)	(73)	(162)	(264)
1600	400	78.5	191.2	460.6	1037.7	153.3	389.5	937.7	2115.8
		(124)	(250)	(635)	(1562)	(209)	(508)	(1167)	(2846)
	800	41.0	102.7	216.1	449.0	85.7	199.6	439.8	936.0
		(56)	(152)	(273)	(595)	(111)	(269)	(641)	(1173)
	1200	28.5	61.5	127.3	280.5	58.0	121.8	265.0	559.0
		(48)	(77)	(169)	(348)	(76)	(151)	(322)	(764)
	1600	21.3	47.0	97.1	203.1	44.3	87.2	196.9	372.7
		(38)	(70)	(113)	(276)	(61)	(103)	(244)	(462)
3600	400	102.7	271.8	727.4	1494.4	243.3	682.0	1537.9	3692.5
		(150)	(409)	(941)	(2323)	(356)	(916)	(2179)	(4879)
	800	56.0	140.8	374.5	733.8	121.3	329.9	775.2	1793.6
		(90)	(191)	(553)	(988)	(151)	(419)	(1129)	(2157)
	1200	35.5	91.3	206.3	490.2	75.7	193.4	430.6	947.3
		(57)	(128)	(248)	(633)	(101)	(270)	(528)	(1208)

	1600	25.3	61.3	139.4	310.6	57.2	124.7	289.7	639.6
		(43)	(84)	(200)	(419)	(70)	(166)	(372)	(831)
6400	400	156.6	384.0	795.0	1922.3	363.0	1046.9	2230.0	4483.7
		(262)	(530)	(1435)	(2988)	(516)	(2039)	(2944)	(6102)
	800	75.2	176.9	463.1	1113.1	179.4	474.0	1145.6	2533.0
		(105)	(239)	(616)	(1586)	(224)	(691)	(1458)	(3671)
	1200	44.8	112.6	257.9	641.0	101.3	260.0	582.4	1401.8
		(68)	(139)	(337)	(984)	(167)	(359)	(830)	(1744)
	1600	33.5	74.7	193.1	389.1	68.4	180.4	411.1	968.1
		(54)	(103)	(258)	(531)	(90)	(232)	(520)	(1251)

a: average; b: worst case.

References

Aggarwal, A. and J. K. Park (1990), Improved Algorithms for Economic Lot-Size Problems, Working paper, IBM Thomas J. Watson Research Center, Yorktown Heights, New York.

Baker, K. R., P. Dixon, M. J. Magazine, and E. A. Silver (1978). An Algorithm for the Dynamic Lot-Size Problem with Time-Varying Production Capacity Constraints, *Management Science* **24**, 1710–1720.

Bitran, G. R. and H. H. Yanasse (1982), Computational Complexity of the Capacitated Lot Size Problem, *Management Science* **28**, 1174–1186.

Chand, S. and S. Sethi (1983), Finite Production Rate Inventory Models with First and Second Shift Setups, *Naval Research Logistics Quarterly* **30**, 401–414.

Chen, H.-D. and C.-Y. Lee (1991), A Simple Algorithm for the Error Bound of the Dynamic Lot Size Model Allowing Speculative Motive, to appear in *IIE Transactions*.

Chen, H.-D., D. W. Hearn, and C.-Y. Lee (1994), A New Dynamic Programming Algorithm for Single Item Capacitated Dynamic Lot Size Model, *Journal of Global Optimization* **4**, 285–300.

Chung, C. S. and C. H. M. Lin (1988), An $O(T^2)$ Algorithm for the NI/G/NI/ND Capacitated Lot Size Problem, *Management Science* **34**, 420–426.

Chung, C.-S., J. Flynn, and C.-H. M. Lin (1990), An Efficient Algorithm for the Capacitated Lot Size Problem, Working paper, College of Business Administration, Cleveland State University, Cleveland, Ohio.

Denardo, E. V. (1982), *Dynamic Programming: Models and Applications*. Prentice-Hall, Englewood Cliffs, New Jersey.

Dongarra, J. J. (1989), Performance of Various Computers Using Standard Linear Equations Softwares, Technique Report CS-89-85, Mathematical Science Section, Oak Ridge National Laboratory, Oak Ridge, TN.

Erenguc, S. S. and S. Tufekci (1987), A Branch and Bound Algorithm for a Single-Item Multi-Source Dynamic Lot Sizing Problem with Capacity Constraints, *IIE Transactions* **19**, 73–80.

Erenguc, S. S. and Y. Aksoy (1990), A Branch and Bound Algorithm for a Single Item Nonconvex Dynamic Lot Sizing Problem with Capacity Constraints, *Computers and Operations Research* **17**, 199–210.

Federguren, A. and M. Tzur (1991), A Simple Forward Algorithm to Solve General Dynamic Lot Sizing Models with n Periods in $O(n \log n)$ or $O(n)$ Time, *Management Science* **37**, 909–925.

Florian, M. and M. Klein (1971), Deterministic Production Planning with Concave Costs and Capacity Constraints, *Management Science* **18**, 12–20.

Florian, M., J. K. Lenstra, and A. H. G. Rinnooy Kan (1980), Deterministic Production Planning: Algorithms and Complexity, *Management Science* **26**, 669–679.

Jagannathan, R. and M. R. Rao (1973), A Class of Deterministic Production Planning Problems, *Management Science* **19**, 1295–1300.

Kirca, Ö. (1990), An Efficient Algorithm for the Capacitated Single Item Dynamic Lot Size Problem, *European Journal of Operational Research* **45**, 15–24.

Lambert, A. and H. Luss (1982), Production Planning with Time-Dependent Capacity Bounds, *European Journal of Operational Research* **9**, 275–280.

Lambrecht, M. and J. Vander Eecken (1978), A Capacity Constrained Single-Facility Dynamic Lot-Size Model, *European Journal of Operational Research* **2**, 132–136.
Lee, S.-B. and P. H. Zipkin (1989), A Dynamic Lot-Size Model with Make-or-Buy Decisions, *Management Science* **35**, 447–458.
Lippman, S. A. (1969), Optimal Invenstory Policy with Multiple Set-Up Costs, *Management Science* **16**, 118–138.
Love, S. F. (1973), Bounded Production and Inventory Models with Piecewise Concave Costs, *Management Science* **20**, 313–318.
Sethi, S. and S. Chand (1981), Multiple Finite Production Rate Dynamic Lot Size Inventory Models, *Operations Research* **29**, 931–944.
Swoveland, C. (1975), A Deterministic Multi-Period Production Planning Model with Piecewise Concave Production and Holding-Backorderer Costs, *Management Science* **21**, 1007–1013.
Wagelmans, A., S. Van Hoesel, and A. Kolen (1992), Economic Lot-Sizing: an $O(n \log n)$ Algorithm that Runs in Linear Time in the Wagner–Whitin Case, *Operations Research* **40**, S145–S156.
Wagner, H. M. and T. M. Whitin (1958), Dynamic Version of the Economic Lot Size Model, *Management Science* **5**, 89–96.
Zangwill, W. (1966), A Deterministic Multi-Period Production Scheduling Model with Backlogging, *Management Science* **13**, 105–119.