MANAGEMENT SCIENCE
Vol. 37, No. 8, August 1991
Printed in U.S.A.

A SIMPLE FORWARD ALGORITHM TO SOLVE GENERAL DYNAMIC LOT SIZING MODELS WITH n PERIODS IN $0(n \log n)$ OR $0(n)$ TIME*

AWI FEDERGRUEN AND MICHAL TZUR†
Graduate School of Business, Columbia University, New York, New York 10027

This paper is concerned with the general *dynamic lot size* model, or (generalized) Wagner-Whitin model. Let n denote the number of periods into which the planning horizon is divided. We describe a simple forward algorithm which solves the general model in $0(n \log n)$ time and $0(n)$ space, as opposed to the well-known shortest path algorithm advocated over the last 30 years with $0(n^2)$ time.

A *linear*, i.e., $0(n)$-time and space algorithm is obtained for two important special cases: (a) models *without* speculative motives for carrying stock, i.e., where in each interval of time the per unit order cost increases by less than the cost of carrying a unit in stock; (b) models with nondecreasing setup costs.

We also derive conditions for the existence of *monotone* optimal policies and relate these to known (planning horizon and other) results from the literature.

(DYNAMIC LOT SIZING MODELS; DYNAMIC PROGRAMMING; COMPLEXITY)

This paper is concerned with the *dynamic lot size* model, one of the most frequently employed deterministic single-item inventory planning models. This model was introduced by Wagner and Whitin (1958) and is therefore often referred to as the Wagner-Whitin model (*W-W model*): it specifies a horizon divided into finitely many (say n) periods each with a known demand which must be satisfied. An unlimited amount may be ordered (produced) in each period. The cost structure consists of fixed-plus-linear order (or production) costs and holding costs assumed to be proportional with the end-of-the-period inventory levels. All parameters, i.e., demands, setup costs, variable replenishment and holding cost rates, may differ from period to period.

Two distinct rationales prevail for maintaining inventories in systems with deterministic demands and unlimited replenishment opportunities:

(I) the *cycle stock motive*: economies of scale in the replenishment costs provide an incentive for order quantities to cover more than a single period's demand;

(II) the *speculative motive* (see Chand and Morton 1986): even in the absence of economies of scale, it may be advantageous to order some future period's demand in the current period, if the future cost of ordering a unit exceeds the cost of ordering this unit now and carrying it until the future period.

In this paper we describe a simple algorithm which solves the *general* dynamic lot size model in $0(n \log n)$ time and $0(n)$ space, as opposed to the well-known shortest path algorithm advocated over the last 30 years with $0(n^2)$ time. A *linear*, i.e., $0(n)$-time and space algorithm is obtained for two important special cases:

(a) models *without* speculative motives for carrying stock, i.e., instances in which in each interval of time, the per unit order cost increases by less than the cost of carrying a unit in stock over this interval (*constant* variable order cost rates represent a special case of such models; Wagner and Whitin, for example, originally confined themselves to this case);

(b) instances with nondecreasing setup costs.

Numerical experiments, reported in §6, confirm that use of the above algorithms results

* Accepted by L. Joseph Thomas; received January 10, 1990. This paper has been with the authors 7 months for 2 revisions.

† Also affiliated with The Wharton School, University of Pennsylvania, Philadelphia, PA 19104.

0025-1909/91/3708/0909$01.25

in major savings in computational time. Indeed problems with 5000 time periods ($n = 5000$) require no more than 1.5 CPU seconds (in average) when executed on an IBM 4381. The algorithm outperforms efficient implementations of the Wagner-Whitin method for $n \geq 20$.

Our procedures consist of iterative updates of a list of candidate last order periods as the planning horizon t is incremented to $(t + 1)$ $(t = 1, \ldots, n - 1)$. The $n \log n$ complexity term in the most general procedure is obtained from the effort to insert or delete an element in this list which in the worst case may be of size n, but in practice is very small indeed. In our numerical experience with problems with up to 5000 periods, the list was never observed to contain more than five elements! (See §6.) A bounded list size and hence linear complexity can in fact *rigorously* be shown if the parameters are bounded from above and below by positive constants. One can therefore argue that our procedure is in practice of linear time even in those rare settings where the worst case bound is $0(n \log n)$.

These results show that exact solution of dynamic lot sizing models is computationally attractive, even when used as a subroutine for material requirement planning systems or systems with tens of thousands of items, see, e.g., Orlicky (1975, pp. 132–133), Smith (1978) and Bitran et al. (1984). This conclusion contrasts prior perceptions based on the exclusive availability of the Wagner-Whitin method. Indeed, considerable effort has been devoted to the study of a variety of *heuristic* methods, all with significant optimality gaps and with comparable complexity as the algorithms of this paper. See, e.g., the Least Unit Cost heuristic, the Part-Period Balancing heuristic, the Economic Order Quantity heuristic, as well as Silver and Meal (1973), Axsäter (1982, 1985), Peterson and Silver (1979), Bitran et al. (1984).

The proposed algorithms are all the more important when the dynamic lot sizing model is used as an approximation of nonstationary generalizations of the *continuous time* Economic Order Quantity model, in which cumulative demands are given by a general nondecreasing function of time and cost parameters exhibit general time-dependencies. See e.g. Barbosa and Friedman (1978), Donaldson (1977), Resh et al. (1976) and Friedman and Winter (1980). A close approximation via the discrete-time dynamic lot sizing model may require a fine time grid and thus result in a horizon with a large number of periods.

Many hierarchical planning problems for multi-item multi-stage production systems with explicit representation of setup costs are formulated as mixed integer programs and solved via Lagrangean relaxation, see Graves (1982). These decompose into many W-W models, each of which needs to be solved for numerous combinations of the Lagrange multipliers. Bitran and Tirupati (1989)'s recent survey on hierarchical planning models states that a comparison between the approaches *with* explicit representation of setup costs and those *without* fixed costs, suggests that the former are more expensive but "likely to provide more cost effective schedules." The new algorithms reduce the complexity of the Lagrangean relaxation methods by a factor n and may all but eliminate any comparative computational disadvantages. Other examples of planning models in which the W-W model arises as a subproblem, include the multi-item capacitated lot-size problem, see e.g. Bitran and Matsuo (1986), Eppen and Martin (1987), Martin (1987), and the references therein.

It is well known (see Wagner and Whitin) that in the W-W model optimal order strategies exist under which orders are placed *if and only if* inventory equals zero. A zero-inventory ordering strategy is completely determined by the specification of the last order period $l(t)$ preceding any given horizon t $(t = 1, \ldots, n)$. Thus, for any $1 \leqq l \leqq t \leqq n$, let $F(l, t)$ = the minimum cost in the first t periods, if the final setup is performed in period l.

Our algorithms are based on the key observation that for any pair of periods $k < l$ the difference function $\Delta_{k,l}(t) \overset{\text{def}}{=} F(k, t) - F(l, t)$ is *monotone* (in $t \geqq l$). This difference function is in fact monotone decreasing (increasing) if (no) speculative motives exist for carrying inventories from period k to period l, i.e., if the variable cost of ordering a unit in period k and carrying it till period l is less than (exceeds) the variable ordering cost in period l.

It is easily verified that the increasingness property of the difference functions $\Delta_{k,l}(\cdot)$ $(l > k)$ in models without speculative motives for carrying inventories also implies that a *monotone optimal* policy exists, i.e., a policy in which $l(t)$, the last period with a setup in a horizon of t periods, is *nondecreasing* in t. Topkis (1968, 1978) and Topkis and Veinott (1972) present the first general existence conditions for monotone optimal policies in dynamic programs, see also Chapter 8 in Heyman and Sobel (1984). See p. 511 of the important paper by Whitney (1935) for an early antecedent.

The existence of a monotone optimal policy was in fact used for the derivation of *planning horizon* theorems and procedures, see e.g. Wagner and Whitin (1958), Zabel (1964), Eppen et al. (1969), Thomas (1970), Blackburn and Kunreuther (1974), Lundin and Morton (1975), Bensoussan et al. (1983), Chand (1982), Chand, Sethi and Proth (1990) and Chand, Sethi and Sorger (1989). A period L is called a *forecast horizon* if the decisions for the first $l \geqq 1$ periods in the optimal solution of a problem with horizon length L are not affected by the model parameters (e.g. demands, costs) for periods beyond the horizon L. Such a period l is called a *planning horizon*. Planning horizon procedures allow saving the effort of forecasting unnecessary future data.

The above monotonicity properties have, to our knowledge, never been exploited for the design of efficient solution methods. Several articles have recently appeared in the Computer Science literature describing efficient algorithms for the solution of dynamic programs in which *all* of the difference functions $\Delta_{k,l}(\cdot)$ are *increasing* or in which *all* are *decreasing*. See e.g. Hirshberg and Larmore (1987), Wilber (1988), Aggarwal et al. (1987), Galil and Giancarlo (1989) and Miller and Myers (1988). The Computer Science literature has been motivated by applications of sequence comparison in, for example, molecular biology (e.g. Waterman 1984), geology (e.g. Smith and Waterman 1980) and speech recognition (e.g. Kruskal and Sankoff 1983).

As pointed out above, our algorithm applies to a setting where some of the difference functions may be increasing and others decreasing. Note that $0(n^2)$ operations are required for the mere evaluation of the costs on the arcs in the network corresponding with the W-W model. It is therefore all the more striking that an *optimal* policy may be determined in at most $0(n \log n)$ time.

After completion of this paper we have become aware of two alternative and independently obtained $0(n \log n)$ solution methods by Wagelmans et al. (1989) and Aggarwal and Park (1990). The former is a backward algorithm whose derivation is based on geometric arguments. The latter is a recursive procedure in which a problem with a given horizon is solved by two recursive calls to subproblems of half the original size and a third problem which is solvable in linear time via one of the above-mentioned algorithms for increasing or decreasing difference functions. The constant factor in the time bound is therefore quite large, see also Galil and Giancarlo (1989) commenting on a similar recursive procedure in Wilber (1988). The worst case complexity of our algorithm is $3n \log n + 0(n)$ and can be argued to be $0(n)$ in practice under the assumption of parameters that are bounded from above or below by positive constants, see Proposition 1. No alternative linear time procedure appears to exist for the prevalent case where the setup costs are nondecreasing over time.

We complete this introduction with an outline of the remainder of this paper. In §1 we introduce the notation and derive some preliminary results. Our $0(n \log n)$ general algorithm is derived in §2. §3 and §4 are devoted to the two above-mentioned special cases in which the complexity of the algorithm reduces to $0(n)$: nondecreasing setup costs (§3) and models without speculative inventory motives (§4). In §5 we discuss conditions for the existence of monotone policies. §6 contains our numerical results.

1. Notation and Preliminaries

The dynamic lot size model with a horizon of n periods, is specified by the following parameters:

d_i = demand in period i ($i = 1, \dots, n$);

K_i = setup cost in period i ($i = 1, \dots, n$);

c_i = variable per unit order cost in period i ($i = 1, \dots, n$);

h_i = cost of carrying a unit of inventory at the end of period i ($i = 1, \dots, n$).

We assume, without loss of generality, that the starting inventory in period 1 and the ending inventory in period n equal *zero*.

We use the following auxiliary notation:

$D(i) = \sum_{k=1}^{i} d_k$ represents the cumulative demand in periods $1, \dots, i$;

$H(i) = \sum_{k=1}^{i} h_k$ denotes the cumulative cost of carrying a unit from period 1 through period i ($i = 1, \dots, n$).

For all $i < j$, let

$h_{ij} = h_i + h_{i+1} + \cdots + h_{j-1}$;

$c_{ij} = c_i + h_{ij}$ (the variable cost of ordering a unit in period i and carrying it till period j);

$\tilde{C}(i) = c_{in} - h_{1n} = c_i - H(i-1)$. (Thus, $\tilde{C}(i) - \tilde{C}(1)$ represents the differential in variable costs between ordering a unit in period i, versus ordering and carrying it from the very first period.)

$S(i, j)$ = the total inventory carrying cost under zero-inventory ordering in periods $i, \dots, j$ when placing an order in period i to cover demands through period j
$= \sum_{r=i}^{j-1} h_r \sum_{k=r+1}^{j} d_k = \sum_{r=i}^{j-1} h_r(D(j) - D(r))$,

$S(i) = S(1, i)$.

The following identities are used below for $i < k < j$:

$$\begin{aligned}
S(i,j) &= \sum_{r=i}^{j-1} h_r \sum_{R=r+1}^{j} d_R = \sum_{r=i}^{k-1} h_r \sum_{R=r+1}^{j} d_R + \sum_{r=k}^{j-1} h_r \sum_{R=r+1}^{j} d_R \\
&= \sum_{r=i}^{k-1} h_r \sum_{R=r+1}^{k} d_R + \sum_{r=i}^{k-1} h_r \sum_{R=k+1}^{j} d_R + S(k,j) \\
&= S(i,k) + \sum_{r=i}^{k-1} h_r(D(j) - D(k)) + S(k,j) \\
&= S(i,k) + S(k,j) + [D(j) - D(k)][H(k-1) - H(i-1)]; \qquad (1a)
\end{aligned}$$

$$\begin{aligned}
S(i,j) &= \sum_{r=i}^{j-1} h_r \sum_{R=r+1}^{j} d_R = \sum_{r=i}^{k-1} h_r \sum_{R=r+1}^{j} d_R + \sum_{r=k}^{j-1} h_r \sum_{R=r+1}^{j} d_R \\
&= \sum_{r=i}^{k-1} h_r \sum_{R=r+1}^{k} d_R + \sum_{r=i}^{k-1} h_r \sum_{R=k+1}^{j} d_R + h_k \sum_{R=k+1}^{j} d_R + \sum_{r=k+1}^{j-1} h_r \sum_{R=r+1}^{j} d_R \\
&= S(i,k) + \sum_{r=i}^{k-1} h_r \sum_{R=k+1}^{j} d_R + S(k+1,j) \\
&= S(i,k) + S(k+1,j) + [D(j) - D(k)][H(k) - H(i-1)]. \qquad (1b)
\end{aligned}$$

In particular choosing $i = 1$ and $k = j - 1$ in (1a) we get

$$S(j) = S(j-1) + d_j H(j-1), \qquad j \geqq 2; \qquad S(1) = 0. \qquad (1c)$$

Likewise, choosing $i = 1$ in (1a) we obtain for all $1 < k < j$

$$S(k,j) = S(j) - S(k) - [D(j) - D(k)]H(k-1). \qquad (1d)$$

As pointed out in the Introduction, it is well known since Wagner and Whitin that optimal policies exist under which orders are placed if and only if inventory equals zero, and such *zero-inventory ordering policies* are completely determined by the specification of the last order period $l(t)$ preceding any given horizon t ($t = 1, \ldots, n$). We give a (short) proof for the optimality of zero-inventory ordering policies, so as to keep this paper self-contained and because its perturbation argument is needed in the proof of Lemma 3.

LEMMA 1 (Wagner and Whitin 1958). *There exists an optimal zero-inventory ordering policy.*

PROOF. Consider a policy which orders in periods with positive starting inventory. We construct a zero-inventory policy with lower or equal costs. Under the given policy, let period l be the *first* period with positive starting inventory (I_l) in which an order is placed, and let period $i < l$ be the order period preceding period l. (Period i is well-defined since period 1 is an order period with zero starting inventory.) Let X_i (X_l) denote the order quantity in period i (l). Note that $X_i \geqq I_l$ since period i has starting inventory zero.

It is clearly feasible to increase the order quantity in period i by X_l units and to cancel the order in period l. Either this perturbation reduces total costs, or $c_{il} \geqq c_l$ and a cost reduction is achieved by *reducing* the order quantity in period i by I_l units. (This alternative perturbation is also feasible since $X_i \geqq I_l$.) In both cases we obtain a revised policy with lower or equal costs, and with one less period in which an order is placed while its starting inventory is positive. The desired zero-inventory ordering policy may thus be obtained after *finitely* many of the above perturbations. □

Let $F(t)$ = minimum cost in the first t periods, $t = 1, \ldots, n$ and recall that $F(l, t)$ = minimum cost in the first t periods, if the final setup is performed in period $l \leqq t$ ($t = 1, \ldots, n$).

To determine whether for a given horizon t, some period l is a better choice to be the last setup period than some other period k, we first derive some properties of the difference function $\Delta_{k,l}(t) = F(k, t) - F(l, t)$.

Note that

$$F(l, t) = F(l-1) + K_l + S(l, t) + c_l[D(t) - D(l-1)], \tag{2}$$

$$F(k, t) = F(k-1) + K_k + S(k, t) + c_k[D(t) - D(k-1)]. \tag{3}$$

Substitute $S(k, t) = S(k, l-1) + S(l, t) + [D(t) - D(l-1)][H(l-1) - H(k-1)]$, (see (1b)) and $[D(t) - D(k-1)] = [D(t) - D(l-1)] + [D(l-1) - D(k-1)]$ into (3) and subtract (2) from the resulting equation to get

$$\Delta_{k,l}(t) = A(k, l) + (c_{k,l} - c_l)D(t)$$

where, using (1d):

$$\begin{aligned} A(k, l) &= F(k-1) + K_k - F(l-1) - K_l + S(k, l-1) \\ &\quad + c_k[D(l-1) - D(k-1)] + D(l-1)(c_l - c_{k,l}) \\ &= F(k-1) + K_k - F(l-1) - K_l + S(l-1) - S(k) \\ &\quad - H(k-1)[D(l-1) - D(k)] + c_k[D(l-1) - D(k-1)] \\ &\quad + D(l-1)(\tilde{C}(l) - \tilde{C}(k)) \\ &= F(k-1) + K_k - F(l-1) - K_l + S(l-1) - S(k) \\ &\quad + \tilde{C}(k)[D(l-1) - D(k-1)] + d_k H(k-1) \\ &\quad + D(l-1)(\tilde{C}(l) - \tilde{C}(k)). \end{aligned} \tag{4}$$

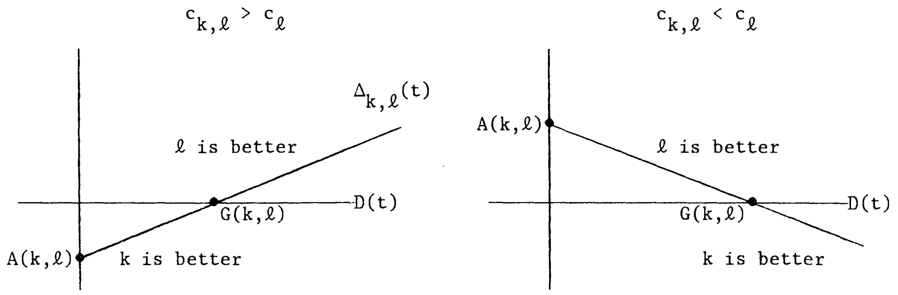

FIGURE 1. Comparative Performance of Periods l and k as Last Setup Periods.

We conclude that $\Delta_{k,l}(t)$ is a *linear* function of $D(t)$ with coefficients that are independent of t. (The final expression for $A(k, l)$ in (4) is given because it allows for efficient evaluation of these quantities, as needed in the algorithms below.)

In view of its linearity in $D(t)$, the difference function $\Delta_{k,l}(\cdot)$ has at most one, easily computable root $G(k, l)$ and the question whether period k is to be preferred to period l as a last setup period may therefore be described by a simple comparison of the cumulative demand $D(t)$ with this root, see Figure 1.

$$G(k, l) = \begin{cases} A(k, l)/(\tilde{C}(l) - \tilde{C}(k)) & \text{if } \tilde{C}(l) \neq \tilde{C}(k), \\ +\infty & \text{if } \tilde{C}(l) = \tilde{C}(k) \text{ and } A(k, l) \leqq 0, \quad k < l, \\ -\infty & \text{if } \tilde{C}(l) = \tilde{C}(k) \text{ and } A(k, l) > 0. \end{cases} \tag{5}$$

(Note that $c_{k,l} - c_l = c_k + h_{k,l} - c_l = c_k + H(l-1) - H(k-1) - c_l = \tilde{C}(k) - \tilde{C}(l)$.)

We conclude:

LEMMA 2. *Fix $k < l \leqq t$:*

(a) $\Delta_{k,l}(t) = A(k, l) + (c_{k,l} - c_l)D(t)$. *Thus, $\Delta_{k,l}(t)$ is a linear function of $D(t)$ with coefficients that are independent of t!*

(b) *If $c_{k,l} > c_l$ then $\Delta_{k,l}(\cdot)$ is increasing; $\Delta_{k,l}(t) \geqq 0$ if and only if $D(t) \geqq G(k, l)$.*

(c) *If $c_{k,l} < c_l$ then $\Delta_{k,l}(\cdot)$ is decreasing; $\Delta_{k,l}(t) \geqq 0$ if and only if $D(t) \leqq G(k, l)$.*

(d) *If $c_{k,l} < c_l$ then $\Delta_{k,l}(\cdot)$ is constant and $\Delta_{k,l}(t) \geqq 0$ if and only if $D(t) \geqq G(k, l)$.*

It is useful to extend the definition of $G(k, l)$ to arbitrary pairs (k, l) in the following symmetric way:

$$G(l, k) = G(k, l); \qquad k < l.$$

2. The General Algorithm

As pointed out above, solution of the W-W model reduces to determining a sequence $\{l(t): t = 1, \ldots, n\}$ with $l(t)$ an optimal period to perform the *last* setup when minimizing the cost in the first t periods, i.e., $F(l(t), t) = F(t)$ $(t = 1, \ldots, n)$. Our proposed algorithm is, like the classical shortest path procedure, a *forward* algorithm with *sequential* determination of a pair $(l(j), F(j))$ for $j = 1, \ldots, n$.

Assume therefore that at the beginning of the jth iteration, $\{(l(k), F(k)): 1 \leqq k \leqq j - 1\}$ are known. The proposed algorithm constructs a list containing all periods that, among the first j periods, are optimal terminal order points for some horizon $t \geqq j$. While striving for the smallest such list, it is to be recognized that the future demands $d_{j+1}, \ldots, d_n$ have not been inspected at this point, and it is inefficient to do so. We therefore treat all *future* demands $d_{j+1}, \ldots, d_n$ as unknown parameters so that all future cumulative

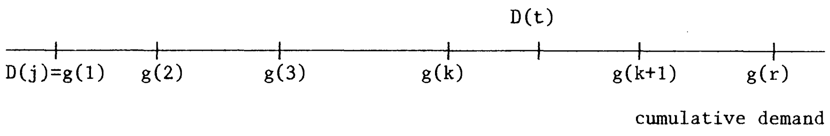

FIGURE 2.

demands may adopt any potential value $D \geqq D(j)$. Thus, let $\Omega(j) = \{1 \leqq l \leqq j$: there exists a horizon $t \geqq j$ with a potential cumulative demand $D \geqq D(j)$ for which l is the *lowest* index with $F(l, t) = \min_{1 \leq i \leq j} F(i, t)\}$.

In other words, $\Omega(j)$ contains a period l only if for some horizon with potential cumulative demand $\geqq D(j)$ it is a better terminal order period than any of the periods $1, \ldots, j$ and a *strictly* better terminal order period than any of the lower indexed periods $1, \ldots, l-1$; $\Omega(j)$ is a *minimal* set in this sense. Clearly, $\Omega(j)$ contains a (globally) optimal terminal order period for the horizon $t = j$. We refer to $\Omega(j)$ as the jth Minimal Optimal Predecessors list.

Our list specifies, in fact, a sequence of critical values $g(1), \ldots, g(r)$ such that the kth element of the list is the unique optimal last setup period for any horizon $t \geqq j$ with potential cumulative demand $g(k) < D < g(k+1)$, see Figure 2. It is conceivable that none of the *actual* cumulative demands $D(j+1), \ldots, D(n)$ is contained in the interval $(g(k), g(k+1))$ in which case the kth element of the list could, in principle, be eliminated. It is *this* possibility which requires us to use *potential* cumulative demands in the definition of $\Omega(j)$. To identify whether this is the case or not requires however a significant additional amount of work which is $0(n \log n)$ itself. This is unnecessary and does not appear to be justified in particular since the sets $\{\Omega(j)\}$ tend to be extremely small, see the Introduction and §6.

Our proposed algorithm consists of *efficient* iterative updates of the sets $\Omega(j)$ ($j = 1, \ldots, n$) and a *simple* identification of an optimal terminal order period $l(j)$ from among the collection $\Omega(j)$ by maintaining the periods in a sequence which guarantees that the first element in $\Omega(j)$ is an optimal last order period for the horizon $t = j$. Note that $\Omega(1) = \{1\}$ and $\Omega(j) \subseteq S^* \overset{\text{def}}{=} (\Omega(j-1) \cup \{j\})$.

We now exhibit a simple procedure to identify which, if any, of the elements of $\Omega(j-1) \cup \{j\}$ need to be eliminated to obtain $\Omega(j)$.

THEOREM 1 (Main theorem: *Characterization of the Minimal Optimal Predecessors lists*). *Fix $j \in \{1, \ldots, n\}$. Let $S \subseteq \{1, \ldots, j\}$ be a collection of periods which contains $\Omega(j)$, i.e., $\Omega(j) \subseteq S \subseteq \{1, \ldots, j\}$. Assume the elements of S are numbered in nonascending order of their $\tilde{C}$-values, i.e., $S = \{i_1, \ldots, i_r\}$ with $\tilde{C}(i_1) \geqq \tilde{C}(i_2) \geqq \cdots \geqq \tilde{C}(i_r)$. (Periods with equal $\tilde{C}$-values are ranked in ascending order of their period indices, i.e., if $\tilde{C}(i_k) = \tilde{C}(i_{k+1})$ then $i_k < i_{k+1}$, $k = 1, \ldots, r-1$.) Let $g(1) = D(j)$ and $g(l) = G(i_l, i_{l-1})$ for $l = 2, \ldots, r$ (so that i_l dominates i_{l-1} as a last order period when the cumulative demand in period j or later is bigger than $g(l)$ and vice versa).*

(a) *$S = \Omega(j)$ i.e., S is the j*th *Minimal Optimal Predecessor list if and only if*

$$g(1) < g(2) < \cdots < g(r) < \infty. \tag{6}$$

(b) *If (6) holds, then $i_1 = l(j)$.*

(c) (i) *If $g(2) \leqq D(j)$ then i_1 may be eliminated from the collection of optimal predecessors S, i.e., $(S \backslash \{i_1\}) \supseteq \Omega(j)$;* (ii) *For $k = 2, \ldots, r-1$, if $g(k+1) \leqq g(k)$ then $(S \backslash \{i_k\}) \supseteq \Omega(j)$;* (iii) *If $g(r) = \infty$ then $(S \backslash \{i_r\}) \supseteq \Omega(j)$.*

PROOF. See Appendix.

Theorem 1(b) shows that if $\Omega(j)$ is available as a *list* with its elements ranked in nonascending order of their $\tilde{C}$-values and ties broken according to the period index (see

Theorem 1), the *first* element in the list qualifies as an *optimal* last order period for a horizon of length j. Moreover, Theorem 1(c) suggests a simple procedure for *updating* the ranked list $\Omega(j+1)$ from the ranked list $\Omega(j)$. For i_{k-1} (i_{k+1}) is to be preferred over i_k when the cumulative demand in period $(j+1)$ or later is less (bigger) than $g(k)$ ($g(k+1)$). Thus, if $g(k+1) \leqq g(k)$, i_k is *always* dominated by i_{k-1} or i_{k+1}.

As mentioned above, $\Omega(1) = \{1\}$ and $\Omega(j+1) \subseteq S = (\Omega(j) \cup \{j+1\})$, $j \geqq 1$. Let $\Omega(j) = \{i_1, \ldots, i_r\}$. In view of Theorem 1, parts (a) and (c), the ranked list $\Omega(j+1)$ may thus be obtained from the ranked list $\Omega(j)$ by the following two steps: first delete any period i from the beginning of the list if $g(2)$ is less than or equal to the new cumulative demand $D(j+1)$; next insert period $(j+1)$ in its proper place (according to its $\tilde{C}$-value). Let $\Omega'(j)$ denote the ranked list obtained after these two steps have been performed.

Assume period $(j+1)$ is inserted between periods i_p and i_{p+1} $(1 \leqq p \leqq r-1)$, i.e., $\tilde{C}(i_p) \geqq \tilde{C}(i_{p+1})$. (The case where period $j+1$ is added at the beginning or the end of the list, is analogous and, if anything, simpler.) Computing $G(i_p, j+1)$ and $G(j+1, i_{p+1})$ via (5) we encounter two cases:

Case I.

$$G(i_{p-1}, i_p) < G(i_p, j+1) < G(j+1, i_{p+1}) < G(i_{p+1}, i_{p+2}). \tag{9}$$

Case II. At least one of the three inequalities in (9) fails to hold.

In Case I, period $(j+1)$ is an element of $\Omega(j+1)$ and the ranked list $\Omega'(j)$ is the ranked list $\Omega(j+1)$, see part (a) of Theorem 1.

In Case II, at least one of the elements in $\Omega'(j)$ is to be eliminated. For example if the left (right) most inequality in (9) is violated then period i_p (i_{p+1}) is to be eliminated and if the middle inequality fails to hold, period $(j+1)$ is itself to be eliminated. In the latter case, it is easily verified from the definition of $\Omega(j)$ and $\Omega(j+1)$ that the update is completed with the discarding of period $(j+1)$ from $\Omega'(j)$. If i_p or i_{p+1} are eliminated, period $(j+1)$ gets a new neighbor in the ranked list requiring an additional evaluation of the $G(\cdot,\cdot)$ function, with the potential of additional deletions until a ranked list satisfying (6) is obtained. In view of Theorem 1(a) that list represents $\Omega(j+1)$.

Before presenting the formal algorithm we derive an additional test on the basis of which period $(j+1)$ may be eliminated in the process of constructing $\Omega(j+1)$ from $\Omega(j)$ $(j \geqq 1)$. This test is based on a *single* comparison of K_j and $K_{i_{p+1}}$ and is therefore to be preferred to the above described elimination tests which require at least one evaluation of the $G(\cdot,\cdot)$ function in addition to a comparison. It also provides the foundation for the simplified algorithm in §3 which applies when the setup costs are nondecreasing.

LEMMA 3. *Let* $1 \leqq i < j \leqq n$. *If* $K_j \geqq K_i$ *and* $\tilde{C}(j) \geqq \tilde{C}(i)$ *then* $F(j, t) \geqq \min\{F(l, t)\colon 1 \leqq l < j\}$ *for any horizon* $t \geqq j$ *with potential cumulative demand* D *such that* $D \geqq D(j)$.

PROOF. Consider an inventory policy for periods $1, \ldots, t$ in which period j is the last order period and with cost $F(j, t)$. Under this policy, let X_j denote the order quantity in period j. It is clearly feasible to increase the order quantity in period i by X_j units and decrease the order quantity in period j to zero. This perturbation allows us to save the order cost $[K_j + c_jX_j]$ in period j at the expense of an additional *variable* cost $(c_i + h_{ij})X_j$ for ordering X_j units in period i and carrying them till period j, as well as a possible additional setup cost K_i in period i. Let $\hat{F}(t)$ be the cost of the revised policy. Thus

$$\hat{F}(t) - F(j, t) \leqq (K_i - K_j) + (c_i + h_{ij} - c_j)X_j$$

$$= (K_i - K_j) + (\tilde{C}(i) - \tilde{C}(j))X_j \leqq 0.$$

	$N(\cdot)$	$\tilde{C}(\cdot)$	$g(\cdot)$
FIRST	N(FIRST)	$\tilde{C}(N(\text{FIRST}))$	$g(1)$
FIRST + 1	N(FIRST + 1)	$\tilde{C}(N(\text{FIRST} + 1))$	$g(2)$
⋮			
LAST	N(LAST)	$\tilde{C}(N(\text{LAST}))$	g(LAST − FIRST + 1)

FIGURE 3. A Minimal Optimal Predecessor List $\Omega(\cdot)$.

If the revised policy is a zero-inventory ordering policy it is one with one of the periods $\{1, \ldots, j-1\}$ as the last order period, i.e., $F(j, t) \geqq \hat{F}(t) \geqq \min \{F(l, t)\colon 1 \leqq l < j\}$. If it is not, the perturbations of the proof of Lemma 1 may be applied to construct a zero-inventory ordering policy with costs lower than or equal to $\hat{F}(t) \leqq F(j, t)$ and a last order period $l(t) \leqq j-1$. □

COROLLARY 1. *Fix $j = 2, \ldots, n$. Let $\Omega(j-1) = \{i_1, \ldots, i_r\}$ with g-values $g(1), \ldots, g(r)$ and assume period j is potentially to be inserted between i_p and i_{p+1} $(p < r)$ (i.e., $\tilde{C}(i_p) \geqq \tilde{C}(j) > \tilde{C}(i_{p+1})$). If $K_j \geqq K_{i_{p+1}}$ then $\Omega(j) = \Omega'(j-1) \setminus \{j\}$.*

PROOF. Immediate from Lemma 3 and the definition of $\Omega(j)$ and $\Omega(j-1)$. □

We are now ready for the formal description of the algorithm. We maintain at each stage a ranked list $\Omega = \{N[\text{FIRST}], N[\text{FIRST} + 1], \ldots, N[\text{LAST}]\}$ such that, for $j = 1, \ldots, n$, $\Omega = \Omega(j)$ at the end of the jth iteration. The records in this list are thus numbered as FIRST, FIRST + 1, . . . , LAST for appropriate values of FIRST and LAST. The record labeled k (FIRST $\leqq k \leqq$ LAST) contains *three* numbers, see Figure 3: the period index $N(k)$, $\tilde{C}(N(k))$ and $g(k)$.

Recall that at any given stage of the update from $\Omega(j-1)$ to $\Omega(j)$ $(j = 2, \ldots, n)$ only one of the following three elements of the ranked list may be deleted: (i) the first element, with index FIRST; (ii) the period just *prior* to period j, whose index we refer to as *index* p and (iii) the period just following period j (which therefore has index $p + 1$). (If j belongs at the top (bottom) of the list we set p equal to (FIRST − 1) (LAST).) We therefore distinguish between these three deletion procedures:

(i) *DELTOP*: this procedure deletes the *first* record of the list and sets FIRST: = FIRST + 1;

(ii) *DELUP* (p): this procedure deletes the record with index p, *increments* the record indices $\{\text{FIRST}, \ldots, p-1\}$ by one and sets FIRST: = FIRST + 1;

(iii) *DELDOWN* (p): this procedure deletes the record with index $p + 1$, *decreases* the record indices $\{p+2, \ldots, \text{LAST}\}$ by one and sets LAST: = LAST − 1.

Finally, we need the insertion procedure.

INSERT (j, p): this procedure reduces the record indices $\{\text{FIRST}, \ldots, p\}$ by *one*, sets FIRST: = FIRST − 1 and puts period j in a record with index p.

If the list Ω were maintained as a simple array, then $0(n)$ fetch-and-store operations would be required in every application of the DELUP, DELDOWN, and INSERT procedures. This would affect the asymptotic complexity even though in most computer systems such fetch-and-store operations are considerably cheaper than additions, and often two orders of magnitude cheaper than multiplications.

The problem may for example be overcome by maintaining the list Ω as a *balanced binary tree* see e.g. Tarjan (1983). This is a binary tree in which a *balance condition* is imposed, forcing the depth of an n-node tree to be $0(\log n)$. This requires rebalancing the tree after (or during) each update operation. The access time in a balanced binary tree is $0(\log n)$. We can also rebalance such a tree after an insertion or deletion in $0(\log n)$ time, see Lemma 4.1 in Tarjan (1983). A balanced binary tree is one kind of *binary search tree* in which a set of items is totally ordered by a given key.

```
Algorithm
Step 0: (Initialization)
        $F(0) := 0$; $F(1) := K_1 + c_1 d_1$; $N(1) := 1$; $S(1) := 0$; $\tilde{C}(1) := c_1$
        $D(1) := d_1$; $H(1) := h_1$; $l(1) := 1$; FIRST := LAST := 1;
                                                  $g$(FIRST) := $D(2)$;
        for $j = 2, n$ do
        begin
            $D(j) := D(j-1) + d_j$;      $-H(j) := H(j-1) + h_j$;
            $\tilde{C}(j) := c_j - H(j-1)$;    $S(j) := S(j-1) + d_j H(j-1)$
        end;
Step 1: (Iterative Step)
        For $j := 2, n$ do
        begin
          while $g$(FIRST+1) $\leqq D(j)$ do DELTOP
          $g$(FIRST) := $D(j)$;
          if $\tilde{C}(j) > \tilde{C}(N$(FIRST)) then begin $p$ := FIRST−1; gnew := $D(j)$ end
          else begin $p$ := max{FIRST $\leqq l \leqq$ LAST: $\tilde{C}(N(l)) \geqq \tilde{C}(j)$};
                    gnew := $G(j, N(p))$ end;
          if (($p$ = LAST or $K_j < K_{N(p+1)}$) and gnew $< \infty$) then
          begin
              if ($p$ = FIRST and gnew $\leqq g(p)$) then begin FIRST := FIRST+1;
              $g$(FIRST) := $D(j)$ end
              while ($p >$ FIRST and gnew $\leqq g(p)$) do
                    begin DELUP($p$); gnew := $G(j, N(p))$ end
              if ($p$ = LAST) then begin INSERT ($j,p$); $g(p)$ := gnew end
              else begin
                $x := G(N(p+1), j)$;
                if ($x >$ gnew) then
                                begin
                                INSERT ($j,p$); $g(p)$ := gnew; $g(p+1) := x$;
                                while ($p+1 <$ LAST and $g(p+2) \leqq g(p+1)$) do
                                  begin DELDOWN($p$); $g(p+1) := G(N(p+1), j)$ end
                                end
              end
            end
          $F(j) := F(N$(FIRST)$, j)$ (see (2))
          $l(j) := N$(FIRST)
        end
```

Complexity of the Algorithm

We now evaluate the complexity of the Algorithm. We first calculate all $D(\cdot)$, $H(\cdot)$, $\tilde{C}(\cdot)$ and $S(\cdot)$ values in $0(n)$ time. Once the ranked list $\Omega(j+1)$ has been updated from the list $\Omega(j)$, $l(j+1)$ is determined as the first period in the list and $F(j+1)$ is obtained from (2) in constant time. This part of the computation effort is therefore clearly *linear* in n. Most of the work is therefore associated with successive updates of the Ω-lists. Note, however, that insertion of period j in the ranked list $\Omega(j-1)$ involves $0(\log_2 j)$ comparisons to obtain the proper place for period j and (at most) two evaluations of the $G(\cdot,\cdot)$ function.

The remainder of the work is associated with *deletions*. A deletion occurs after a *single* comparison of a pair of K-values or a pair of G-values may necessitate the evaluation of *one* additional G-value and invokes a certain amount of work associated with the deletion of a record in a ranked list. The latter requires $0(\log j)$ operations provided an appropriate data-structure is employed, see above. Each period $1, \ldots, n$ is deleted *at most once* in the course of the algorithm. We conclude that both the work associated with insertions and that associated with deletions and hence the complexity of the entire algorithm is $0(n \log n)$!

THEOREM 2. *The Algorithm solves the dynamic lot sizing problem, with* $0(n \log n)$ *elementary operations and* $0(n)$ *space requirements.*

PROOF. The proof is immediate from Theorem 1, Corollary 1, the discussion above and the following observations:

Step 1 starts for any given value of j with the elimination of any record l in $\Omega(j-1)$ with $g(l+1) \leqq D(j)$. The element which appears at the top of the list after this round of eliminations, gets $D(j)$ as its g-value in accordance with the convention of Theorem 1. Step 1 proceeds with determining period j's place in the list. If its place is *before* the end of the list ($p <$ LAST) and $K_j \geqq K_{N(p+1)}$ it follows from Corollary 1 that no further updates are needed. Otherwise, period j *may* be included in $\Omega(j)$ and Step 1 proceeds with the possible sequential deletion of records $p, p-1, \cdots$ until the $g(\cdot)$-value of the record to be occupied by period j (*gnew*) is larger than the g-value of its predecessor ($g(p)$).

If $p =$ LAST, i.e., period j belongs at the end of the list, j needs to be inserted there, we have a list satisfying (6), and the update is completed. If $p <$ LAST, we compute the value that record $(p+1)$ would get if period j were inserted ($x = G(N(p+1), j)$). If $x \leqq$ *gnew*, $j \notin \Omega(j)$ and the update is completed by the definition of $\Omega(j)$ and $\Omega(j-1)$; otherwise, we insert period j into record p with its last computed $g(\cdot)$-value (*gnew*). Clearly,

$$g(\text{FIRST}) < g(\text{FIRST}+1) < \cdots < g(p). \tag{10}$$

The update is completed with the possible sequential deletion of records $p+1, p+2, \cdots$ until

$$g(p+1) < g(p+2) < \cdots < g(\text{LAST}). \tag{11}$$

At the end of this last sequence of deletions, we must have $g(p) < g(p+1)$ and hence full monotonicity of the g-values in the entire list, i.e., (6) in view of (10) and (11). For, if $g(p) \geqq g(p+1)$ we would eliminate period j as well as some period $l \in \Omega(j-1)$, which therefore has the property that $F(l, t) < \min \{F(k, t)\colon 1 \leqq k \leqq j-1\}$ for some horizon t with potential cumulative demand $D \geqq D(j)$. But, since $j \notin \Omega(j)$, this period l must be included in $\Omega(j)$, which leads to a contradiction. □

The above complexity analysis allows for arbitrary parameter combinations. It is however more realistic to assume that set-up costs are bounded from above and holding cost rates and demands from below in which case the Algorithm is easily shown to have *linear* time complexity. Assume all demands are rational, hence integer after appropriate scaling.

PROPOSITION 1. *Assume there exists an integer* $M \geqq 1$, *and constants* h_*, K^* *such that* $(d_i + \cdots + d_{i+M}) \geqq 1$, $h_i \geqq h_*$, $c_i \geqq c_*$, $K_i \leqq K^*$ *and* $c_i \leqq c^*$ *for all* $i = 1, \ldots, n$.
(a) *the size of the lists* Ω *is bounded from above by* $K^*/h_* + (c^* - c_*)/h_* + M$; (b) *the Algorithm requires* $0(n)$ *operations.*

PROOF. (a) Consider a period j with $d_j \geqq 1$, and let $i < j$.

$$F(j, j) < F(i, j) \qquad \text{if} \qquad c_{ij} d_j > K_j + c_j d_j \tag{12}$$

Under (12), $c_{ij} > c_j$ and by Lemma 2(b), $\Delta_{i,j}(\cdot)$ is increasing so that $F(j, t) < F(i, t)$ for *all* $t \geqq j$ and $i \notin \Omega(j)$. But, if

$$i < j - \left[\frac{(c^* - c_*)}{h_*} + \frac{K^*}{h_*}\right],$$

then

$$i < j - \left[\frac{(c^* - c_*)}{h_*} + \frac{K^*}{h_* d_j}\right]$$

which implies that $(c_* + (j - i)h_*)d_j > K^* + c^* d_j$, and hence (12) and $i \notin \Omega(j)$. This implies that

$$|\Omega(j)| \leqq \frac{(c^* - c_*)}{h_*} + \frac{K^*}{h_*}.$$

Now let j_1 and j_2 be consecutive periods with strictly positive demands. By the assumptions $d_{j_1} \geqq 1$, $d_{j_2} \geqq 1$ and $j_2 - j_1 \leqq M$. By the analysis above

$$|\Omega(j_1)| \leqq \frac{K^*}{h_*} + \frac{(c^* - c_*)}{h_*}, \qquad |\Omega(j_2)| \leqq \frac{K^*}{h_*} + \frac{(c^* - c_*)}{h_*}$$

and hence

$$|\Omega(t)| \leqq \frac{K^*}{h_*} + \frac{(c^* - c_*)}{h_*} + M \qquad \text{for all } t = j_1 + 1, \dots, j_2 - 1.$$

(b) Since the list Ω is of bounded size, the effort to delete or insert elements in that list is 0(1), and the overall complexity of the Algorithm is *linear*, see the proof of Theorem 2. □

3. Nondecreasing Setup Costs

In this section we consider the special but frequently prevalent case where the setup costs are nondecreasing, i.e., $K_1 \leqq \cdots \leqq K_n$. The proposed simplification of the general Algorithm results immediately from the following corollary of Lemma 3:

COROLLARY 2. *Assume* $K_1 \leqq K_2 \leqq \cdots \leqq K_n$. *Fix* $1 < j \leqq n$. *Let* $\Omega(j-1) = \{i_1, \dots, i_r\}$ *with g-values* $g(1), \dots, g(r)$.

(a) *If* $\tilde{C}(j) \geqq \tilde{C}(N(\text{LAST}))$, $\Omega(j) = \Omega'(j-1)\setminus\{j\}$.

(b) *If* $\tilde{C}(j) < \tilde{C}(N(\text{LAST}))$, $\Omega(j) \subseteq \Omega'(j-1)$ (*i.e.*, $j \in \Omega(j)$) *and period j is inserted by the Algorithm at the end of its list.*

We thus obtain the following simplified version of Step 1 of the Algorithm.

```
Step 1 for Nondecreasing Setup Costs.
For $j = 2,n$ do
  begin
  while ($g$(FIRST+1) $\leqq D(j)$) do DELTOP;
  $g$(FIRST): = $D(j)$;
  if ($\tilde{C}(j) < \tilde{C}(N$(LAST))) do begin
                              while ($G(j,N$(LAST)) $\leqq g$(LAST)) do DELUP(LAST);
                              INSERT($j$,LAST); $g$(LAST): = $G(j,N$(LAST));
                              end
  $F(j)$: = $F(N$(FIRST),$j$)
  $l(j)$: = $N$(FIRST)
  end
```

In view of Corollary 2(b) no effort needs to be expended to determine at what position in the list any new period should be inserted. Recall that this part requires 0(log n) operations in the general algorithm. Moreover records are deleted only from the top or the bottom of the list and inserted only at the bottom of the list. Such deletion and insertions may all be performed in *constant* time, maintaining the list as an ordinary array. We conclude that the complexity of the modified algorithm (with Step 1 for Nondecreasing Setup Costs) is $0(n)$ only!

In addition to allowing for a simplified solution method, Corollary 2 also shows that an optimal strategy exists in which the last order period $l(t)$ in a horizon of t periods, is nondecreasing in t.

COROLLARY 3. *Assume $K_1 \leqq K_2 \leqq \cdots \leqq K_n$. The Algorithm determines a monotone optimal policy, i.e., a policy with $l(j) \leqq l(j+1)$ for $1 \leqq j < n$.*

PROOF. Since new periods get inserted at the bottom of the list (if at all), it follows that the period index which is stored in the first record of this list (N(FIRST)) can never decrease in the process of the Algorithm. Recall from Step 1 that an optimal last order period $l(j)$ (for the first j periods) is determined by setting $l(j) = N(\text{FIRST})$ at the end of Step 1. □

Corollary 3 was first observed in Eppen et al. (1969).

4. Problems without Speculative Inventory Motives

In this section we consider problems with no speculative motive for carrying inventories, i.e., $c_{ij} \geqq c_j$ or $\tilde{C}(i) \geqq \tilde{C}(j)$ for all $1 \leqq i < j \leqq n$. The following corollary is immediate.

COROLLARY 4. *Assume $\tilde{C}(1) \geqq \tilde{C}(2) \geqq \cdots \geqq \tilde{C}(n)$. Period j is inserted by the Algorithm at the end of its list.*

We thus obtain the following simplified version of Step 1 of the Algorithm which is identical to "Step 1 for Nondecreasing Setup Costs" except that the test ($\tilde{C}(j) < \tilde{C}(N(\text{LAST}))$) can be omitted:

```
Step 1 for Models Without Speculative Inventory Motive.
For $j = 2, n$ do
  begin
  while ($g(\text{FIRST} + 1) \leqq D(j)$) do DELTOP;
  $g(\text{FIRST}) := D(j)$;
  while ($G(j, N(\text{LAST})) \leqq g(\text{LAST})$) do DELUP(LAST);
    INSERT($j$, LAST); $g(\text{LAST}) := G(j, N(\text{LAST}))$;
  $F(j) := F(N(\text{FIRST}), j)$;
  $l(j) := N(\text{FIRST})$;
  end
```

We conclude that the modified algorithm has complexity $0(n)$ as in the case of nondecreasing setup costs!

We also obtain the existence of a *monotone* optimal policy in complete analogy to Corollary 3.

COROLLARY 5. *Assume $\tilde{C}(1) \geqq \tilde{C}(2) \geqq \cdots \geqq \tilde{C}(n)$. The Algorithm determines a monotone optimal policy, i.e., a policy with $l(j) \leqq l(j+1)$ for all $1 \leqq j < n$.*

It follows from Lemma 2 that all difference functions are nondecreasing in this case; the algorithms by Wilber (1988), Galil and Giancarlo (1989) and Miller and Myers (1989) may thus be used as alternatives to the above procedure.

5. Monotone Optimal Policies, Related Work

Wagner and Whitin's classical solution method for the dynamic lot sizing problem, consists of determining a shortest path in a network (N, A) with $N = \{i: 1 \leqq i \leqq n\}$, $A = \{(i,j): 1 \leqq i \leqq j \leqq n\}$ and arc lengths $\{\gamma(i,j): i \leqq j\}$ given by

$\gamma(i, j)$ = the total cost under zero-inventory ordering in the interval of periods $\{i, i, i + 1, \ldots, j\}$ when placing an order in period i to cover the demand in this interval; $(i \leqq j)$.

An arc-cost function is called *submodular* if

$$\gamma(i, k) + \gamma(j, l) \leqq \gamma(i, l) + \gamma(j, k), \qquad \text{i.e.,}$$

$$\gamma(i, k) - \gamma(j, k) \leqq \gamma(i, l) - \gamma(j, l) \qquad \text{for all } i \leqq j < k \leqq l.$$

Submodularity of the arc-cost function is clearly *equivalent* to the difference functions $\Delta_{k,l}(\cdot)$ $(k < l)$ being *monotone increasing*. It is immediate from Lemma 2(a) that the absence of speculative motives for carrying inventories, i.e., $\tilde{C}(1) \geqq \tilde{C}(2) \geqq \cdots \geqq \tilde{C}(n)$ implies that the difference functions $\Delta_{k,l}(\cdot)$ $(k < l)$ are monotone increasing and hence that the arc-cost function γ is submodular.

Topkis (1968), Topkis and Veinott (1972) and Topkis (1978) developed a general framework for the existence of monotone optimal policies; for deterministic dynamic programs which can be formulated as shortest path problems their general sufficient condition reduces to submodularity of the arc cost function. (See also Anily and Federgruen (1991) for a simple self-contained proof of the existence of monotone optimal policies under submodular arc cost functions.) The existence of monotone optimal policies was first shown (by *implicit* use of the submodularity property) in Wagner and Whitin (1958) for the case of *constant* variable order costs, and in Eppen et al. (1969) for the more general case where $c_{ij} \geqq c_j$ $(i < j)$, i.e., $\tilde{C}(1) \geqq \tilde{C}(2) \geqq \cdots \geqq \tilde{C}(n)$.

Finally the treatments in Topkis (1968), Topkis and Veinott (1972), Topkis (1978) and Heyman and Sobel (1984) may give the impression that submodularity of the arc-cost function is the *only* general sufficient condition under which the existence of monotone optimal policies can be established. In §3 we have exhibited another class of dynamic lot sizing models (with nondecreasing setup costs and otherwise general parameters) under which monotone optimal policies exist. The following example shows however that submodularity may fail to hold for that class of problems.

EXAMPLE. Let $n = 4$; $d_1 = d_2 = d_3 = 1$ and $d_4 = 10$; $h_i = 0.1$ $(i = 1, \ldots, 4)$; $c_1 = K_1 = 1$; $c_2 = K_2 = 10$ and c_3, c_4, K_3, K_4 arbitrary. One can easily verify that $\gamma(1, 2) = 3.1$; $\gamma(2, 4) = 132.1$; $\gamma(1, 4) = 17.3$ and $\gamma(2, 2) = 20$. Therefore $\gamma(1, 2) + \gamma(2, 4) > \gamma(1, 4) + \gamma(2, 2)$ contradicting submodularity.

See also Anily and Federgruen (1991) for another general class of dynamic programs corresponding with so-called *extremal* partitioning problems, in which monotone optimal policies exist while submodularity of the corresponding arc cost functions may be violated.

6. Numerical Results

In this section we provide a numerical evaluation of the efficiency of the Algorithm. Table 1 exhibits the performance of the Algorithm on a sample of 20 test problems, with n varying from 500 to 5000 as indicated in the first column of the table. In all 20 problems, one period demand, holding and variable order cost rates and setup costs are all generated as uniform random variables on the integers of a given interval. The minimum and maximum values of these intervals are specified in Table 1. Column 6 specifies the CPU time of the Algorithm when executed on an IBM 4381 in Fortran, and measured in seconds.

We report (as a benchmark) in column 7 the CPU times required by Evans' (1985) efficient implementation of the classical (Wagner-Whitin) dynamic programming algorithm.

For the smallest of our test problems (with $n = 500$), the new algorithm is approximately 3 times faster. For $n = 5000$, it is two orders of magnitude cheaper.

TABLE 1

n	c	d	h	k	$0(n \log n)$-algorithm	$0(n^2)$-algorithm
500	(1, 20)	(1, 5)	(1, 10)	(1, 200)	0.47	1.49
500	(1, 50)	(1, 5)	(1, 10)	(1, 200)	0.46	1.47
500	(1, 50)	(1, 10)	(1, 10)	(1, 200)	0.47	1.48
500	(1, 20)	(1, 10)	(1, 5)	(1, 200)	0.49	1.52
500	(1, 10)	(1, 20)	(1, 5)	(1, 100)	0.47	1.57
1000	(1, 20)	(1, 5)	(1, 10)	(1, 100)	0.59	4.92
1000	(1, 50)	(1, 5)	(1, 20)	(1, 100)	0.59	4.72
1000	(1, 50)	(1, 10)	(1, 20)	(1, 100)	0.59	4.71
1000	(1, 20)	(1, 10)	(1, 20)	(1, 200)	0.59	4.92
1000	(1, 10)	(1, 20)	(1, 20)	(1, 200)	0.61	5.01
2000	(1, 20)	(1, 5)	(1, 10)	(1, 100)	0.82	18.31
2000	(1, 50)	(1, 5)	(1, 20)	(1, 100)	0.79	18.00
2000	(1, 50)	(1, 10)	(1, 20)	(1, 100)	0.80	17.56
2000	(1, 20)	(1, 10)	(1, 20)	(1, 200)	0.84	18.41
2000	(1, 10)	(1, 20)	(1, 20)	(1, 200)	0.83	18.70
5000	(1, 20)	(1, 5)	(1, 10)	(1, 100)	1.51	109.84
5000	(1, 50)	(1, 5)	(1, 20)	(1, 100)	1.50	108.69
5000	(1, 50)	(1, 10)	(1, 20)	(1, 100)	1.46	108.04
5000	(1, 20)	(1, 10)	(1, 20)	(1, 200)	1.53	112.81
5000	(1, 10)	(1, 20)	(1, 20)	(1, 200)	1.53	114.61
(1)	(2)	(3)	(4)	(5)	(6)	(7)

As pointed out in the introduction, the list Ω has five or less elements in all of the above problem instances and at any stage of the algorithm.

Appendix. Proof of Theorem 1

Note first that if $i_2 < i_1$ and $\tilde{C}(i_2) \leqq \tilde{C}(i_1)$ then $c_{i_2,i_1} \leqq c_{i_1}$.

Assume first that (6) holds. The minimality of the set S is established by showing that (i) $F(i_1, t) < F(l, t)$ for all $l \in S\setminus\{i_1\}$ when $D(t) < g(2)$; (ii) for $k = 2, \ldots, r-1$: $F(i_k, t) < F(l, t)$ for all $l \in S\setminus\{i_k\}$ when $g(k) < D(t) < g(k+1)$ and (iii) $F(i_r, t) < F(l, t)$ for all $l \in S\setminus\{i_r\}$ when $D(t) > g(r)$. We verify (ii). (Verification of (i) and (iii) is analogous.) Fix k, with $2 \leqq k \leqq r-1$ and $g(k) < D(t) < g(k+1)$, see Figure 2. Apply Lemma 2 sequentially to the pairs $\{i_{k-1}, i_k\}, \{i_{k-2}, i_{k-1}\}, \ldots, \{i_1, i_2\}$ to conclude that:

$$F(i_k, t) < F(i_{k-1}, t) < F(i_{k-2}, t) < \cdots < F(i_1, t) \tag{7}$$

Similarly, applying Lemma 2 sequentially to the pairs $\{i_k, i_{k+1}\}, \ldots, \{i_{r-1}, i_r\}$ one obtains:

$$F(i_k, t) < F(i_{k+1}, t) < \cdots < F(i_r, t). \tag{8}$$

The inequalities (7) and (8) together establish (ii) and (i)–(iii) establish the *minimality* of the set S. (i) also establishes part (b), since $D(j) < G(i_1, i_2) = g(2)$ and since only periods $1, \ldots, j$ may be used as last order periods for a horizon of length j.

Assume now that (6) fails to hold. To show that S fails to be minimal and hence to complete the proof of part (a), it suffices to prove part (c).

(c)(i): It follows from Lemma 2 that $F(i_2, t) \leqq F(i_1, t)$ for *all* t, with $D(t) \geqq g(2)$.

(c)(ii): Apply Lemma 2 to conclude that $F(i_{k+1}, t) \leqq F(i_k, t)$ for all horizons t with $D(t) \geqq g(k+1)$ and $F(i_{k-1}, t) \leqq F(i_k, t)$ for all horizons t with $D(t) \leqq g(k)$ and hence for all horizons with $D(t) \leqq g(k+1)$.

(c)(iii). If $g(r) = G(i_{r-1}, i_r) = +\infty$ then $\tilde{C}(i_r) = \tilde{C}(i_{r-1})$ by (5) and hence, by the numbering convention for the elements of S, $i_r > i_{r-1}$. Moreover, $A(i_{r-1}, i_r) \leqq 0$ so that $F(i_{r-1}, t) \leqq F(i_r, t)$ for all horizons t. □

References

AGGARWAL, A., M. M. KLAWE, S. MORAN, P. SHOR AND R. E. WILBER, "Geometric Applications of a Matrix-Searching Algorithm," *Algorithmica*, 2 (1987), 209-233.

——— AND J. K. PARK, "Improved Algorithms for Economic Lot-Size Problems, "Working paper, IBM Thomas J. Watson Research Center, Yorktown Heights, NY, (1990).

ANILY, S. AND A. FEDERGRUEN, "Structured Partitioning Problems," *Oper. Res.*, 31 (1991), 130-149.

Axsäter, S., "Worst Case Performance for Lot Sizing Heuristics," *European J. Oper. Res.*, 9 (1982), 339–343.

———, "Performance Bounds for Lot Sizing Heuristics," *Management Sci.*, 31 (1985), 634–640.

Barbosa, L. C. and M. Friedman, "Deterministic Inventory Lot Size Models—A General Root Law," *Management Sci.*, 24 (1978), 819–826.

Bensoussan, A., J. Crouhy and J. M. Proth, *Mathematical Theory of Production Planning*, Advanced Series in Management, North-Holland, Amsterdam, 1983.

Bitran, G., T. L. Magnanti and H. H. Yanasse, "Approximation Methods for the Uncapacitated Dynamic Lot Size Problem," *Management Sci.*, 30 (1984), 1121–1140.

——— and H. Matsuo, "The Multi-Item Capacitated Lot Size Problem: Error Bounds of Manne's Formulations," *Management Sci.*, 32 (1986), 350–359.

——— and D. Tirupati, "Hierarchical Production Planning," MIT Sloan School Working Paper #3017-89-MS, MIT, Cambridge, MA 02139, 1989.

Blackburn, J. D. and H. Kunreuther, "Planning Horizons for the Dynamic Lot Size Model with Backlogging," *Management Sci.*, 21 (1974), 251–255.

Chand, S., "Lot Sizing for Products with Finite Demand Horizon and Periodic Review Policy," *European J. Oper. Res.*, 11 (1982), 145–148.

——— and T. E. Morton, "Minimal Forecast Horizon Procedures for Dynamic Lot Size Models," *Naval Res. Logist. Quart.*, 33 (1986), 111–122.

——— S. Sethi and J. M. Proth, "Existence of Forecast Horizons in Undiscounted Discrete-Time Lot Size Models," *Oper. Res.*, 38 (1990), 884–892.

———, ——— and G. Sorger, "Forecast Horizons in the Discounted Dynamic Lot Size Model," University of Toronto Working Paper, Toronto, Canada, 1989.

Donaldson, W. A., "Inventory Replenishment Policy for a Linear Trend in Demand—An Analytical Solution," *Oper. Res. Quart.*, 28 (1977), 663–670.

Eppen, G. D., F. J. Gould and B. P. Pashigian, "Extensions of Planning Horizon Theorem in the Dynamic Lot Size Model," *Management Sci.*, 15 (1969), 268–277.

——— and R. K. Martin, "Solving Multi-Item Capacitated Lot Sizing Problems Using Variable Redefinitions," *Oper. Res.*, 35 (1987), 832–848.

Evans, J. R., "An Efficient Implementation of the Wagner-Whitin Algorithm for Dynamic Lot-Sizing," *J. Oper. Management*, 5 (1985), 229–235.

Friedman, M. and J. Winter, "An Asymptotic Solution of Inventory Lot Size Models with Homogeneous Time-Dependent Demand Functions," *SIAM J. Alg. Discr. Meth.*, 1 (1980), 300–314.

Galil, Z. and R. Giancarlo, "Speeding Up Dynamic Programming with Applications to Molecular Biology," *Theoret. Computer Sci.*, 64 (1989), 107–118.

Graves, S., "Using Lagrangian Techniques to Solve Hierarchical Production Planning Problems," *Management Sci.*, 28 (1982), 260–275.

Heyman, D. and M. Sobel, *Stochastic Models in Operations Research* Vol. II, McGraw-Hill, New York, 1984.

Hirshberg, D. S. and L. L. Larmore, "The Least Weight Subsequence Problem," *SIAM J. Comput.*, 16 (1987), 628–638.

Kruskal, J. B. and D. Sankoff (Eds.), *Time Warps, String Edits, and Macromolecules*: The Theory and Practice of Sequence Comparison, Addison-Wesley, New York, 1983.

Lundin, R. A. and T. E. Morton, "Planning Horizons for the Dynamic Lot Size Model: Zabel vs. Protective Procedures and Computational Results," *Oper. Res.*, 23 (1975), 711–734.

Martin, R. K., "Generating Alternative Mixed-Integer Programming Models Using Variable Redefinitions," *Oper. Res.*, 35 (1987), 820–831.

Miller, W. and E. W. Myers, "Sequence Comparison with Concave Weighting Functions," *Bulletin Math. Biology*, 50 (1988), 97–120.

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

Peterson, R. and E. Silver, *Decision Systems for Inventory Management and Production Planning*, John Wiley, New York, 1979.

Resh, M., M. Friedman and L. C. Barbosa, "On a General Solution of the Deterministic Lot Size Problem with Time Proportional Demand," *Oper. Res.*, 24 (1976), 718–725.

Silver, E. A., "A Simple Inventory Replenishment Decision Rule for a Linear Trend in Demand," *J. Oper. Res. Soc.*, 30 (1979), 71–75.

——— and H. C. Meal, "A Heuristic for Selecting Lot-Size Quantities for the Case of a Deterministic Time-Varying Demand Rate and Discrete Opportunities for Replenishment," *Production and Inventory Management*, 14 (1973), 64–74.

Smith, D., "Material Requirements Planning," in *Studies in Operations Management*, A. Hax (Ed.), North-Holland, Amsterdam (1978).

Smith, T. F. and M. S. Waterman, "New Stratigraphic Correlation Techniques," *J. Geology*, 88 (1980), 451–457.

TARJAN, R., *Data Structures and Network Algorithms*, CBMS-NSF Regional Conf. Ser. Appl. Math., Vol. 44, (1983).

THOMAS, L. J., "Price-Production Decisions with Deterministic Demand," *Management Sci.*, 16 (1970), 747–750.

TOPKIS, D. T., "Ordered Optimal Solutions," Ph.D. dissertation, Stanford University, Stanford, CA, 1968.

———, "Minimizing a Submodular Function on a Lattice," *Oper. Res.*, 26 (1978), 305–321.

———, "Applications of Minimizing a Subadditive Function on a Lattice, unpublished manuscript, 1978.

——— AND A. F. VEINOTT, JR., "Isotone Solutions of Extremal Problems on a Lattice," unpublished manuscript, 1972.

WAGELMANS, A., S. VAN HOESEL AND A. KOLEN, "Economic Lot-Sizing: An $0(n \log n)$-Algorithm That Runs in Linear Time in the Wagner-Whitin Case," CORE Discussion Paper no. 8922, Université Catholique de Louvain, Louvain-la-Neuve, Belgium, (1989).

WAGNER, H. M. AND T. M. WHITIN, "Dynamic Version of the Economic Lot Size Model," *Management Sci.*, 5 (1958), 89–96.

WATERMAN, M. S., "General Methods of Sequence Comparison," *Bulletin Math. Biology*, 46 (1984), 473–501.

WHITNEY, H., "On the Abstract Properties of Linear Dependence," *Amer. J. Math.*, 57 (1935), 509–533.

WILBER, R. E., "The Concave Least Weight Subsequence Problem Revisited," *J. Algorithms*, 9 (1988), 418–425.

ZABEL, E., "Some Generalization of Inventory Planning Horizon Theorem," *Management Sci.*, 10 (1964), 465–471.