Fast Solution and Detection of Minimal Forecast Horizons in Dynamic Programs with a Single Indicator of the Future: Applications to Dynamic Lot-sizing Models

Awi Federgruen • Michal Tzur
Graduate School of Business, Columbia University, New York, New York 10027
Department of Industrial Engineering, Tel Aviv University, Tel Aviv, Israel

In most dynamic planning problems, one observes that an optimal decision at any given stage depends on limited information, i.e. information pertaining to a limited set of adjacent or nearby stages. This holds in particular for planning problems over time, where an optimal decision in a given period depends on information related to a limited future time horizon, a so-called forecast horizon, only. In this paper we identify a general class of dynamic programs in which an efficient forward algorithm can be designed to solve the problem and to identify *minimal* forecast horizons. Such a procedure specifies necessary and sufficient conditions for a stage to arise as a forecast horizon. This class of dynamic programs includes the single-item dynamic lot-sizing model with general concave costs, both with and without backlogging, to which special attention is given.
(*Minimal Forecast Horizons; Fast Solution and Detection; Applications to Lotsizing*)

1. Introduction

In most dynamic planning problems, one observes that an optimal decision at any given stage depends on limited information, i.e. information pertaining to a limited set of adjacent or nearby stages. This holds in particular for planning problems over time (or space), where an optimal decision in a given period (at a given point) depends on information related to a limited future time (space) horizon, a so-called forecast horizon, only.

In this paper we identify a general class of dynamic programs in which an efficient forward algorithm can be designed to solve the problem and to identify *minimal* forecast horizons. Such a procedure specifies necessary and sufficient conditions for a stage to arise as a forecast horizon. A forecast horizon procedure allows for the determination of optimal decisions when accurate information is available over a limited time or space horizon only. This class of dynamic programs includes the single-item dynamic lot-sizing model with general concave costs, both with and without backlogging, to which special attention is given in §§3 and 4. We thus generalize earlier detection procedures of minimal forecast horizons, obtained under (fixed-plus) linear cost structures, see Federgruen and Tzur (1994, 1993), and references therein. In §5 we mention several other problems that belong to the above class of dynamic programs. These include scheduling, equipment replacement, bond refunding, molecular biology, geology, and speech recognition problems.

It is well known that every deterministic dynamic program can be represented as a shortest path problem in an acyclic network. A shortest path is fully characterized by identifying for every node in the network, its optimal predecessor node. It is therefore useful to

0025-1909/95/4105/0874$01.25

characterize for any pair of nodes i, j its associated *difference function*, comparing the optimal cost of reaching any node in the network with i rather than j as its last predecessor. The class of dynamic programs treated in this paper consists of those in which all difference functions can be expressed as functions of a *single* appropriately chosen node-indicator. In the next section (§2) we develop a general algorithm, GENFOR, to solve this class of dynamic programs and to detect minimal forecast horizons.

Assuming that the number of roots of the difference functions is uniformly bounded in n, the number of nodes in the network, the GENFOR algorithm requires $O(n^2)$ determinations of roots of difference functions and $O(n^2 \log^* n)$ additional operations. ($\log^* n$ is a function which grows exceedingly slowly with n; $\log^* n \leq 3$ for $n \leq 3{,}814{,}279$; see §2 for a precise definition.) Recall that standard shortest path methods have similar complexity ($O(n^2)$) while incapable of finding minimal forecast horizons. When all difference functions have at most two roots, the complexity reduces to $O(n^2)$ determinations of roots of difference functions and $O(n^2)$ additional operations. Similarly, if all difference functions have at most a single root (are all nondecreasing or all nonincreasing), the complexity bounds reduce to $O(n \log n)$ ($O(n)$) only. Last, but not least, the complexity is $O(n)$ if a uniform upper bound M prevails for the length of individual arcs on an optimal path (i.e. arc (i, j) is part of the optimal path only if $j - i < M$). We exhibit several important models with this property. Standard shortest path methods can of course be adapted to exploit such upper bounds on the length of individual arcs in the optimal path as well, but this adaptation requires up-front knowledge of the upper bound M. The GENFOR algorithm, on the other hand, reduces automatically to a linear time algorithm without any adaptation or up-front knowledge of an upper bound M.

Our paper does not address conditions for the existence of forecast horizons; it develops efficient algorithms to find minimal forecast horizons whenever they exist. We refer to Bean and Smith (1984), McKenzie (1976), Bean and Smith (1993) and the references therein for important work establishing existence conditions for forecast horizons in general dynamic programs. We postpone our literature review of forecast horizon results for dynamic lot-sizing models to §§3 and 4. Our work is motivated by recent algorithms in the Computer Science literature for dynamic programs of special structure, in particular Galil and Giancarlo (1989).

Forward algorithms, capable of detecting minimal forecast horizons, are related to on-line algorithms, to which an increasing amount of attention is being paid; see Karp (1992) for a recent survey. These are algorithms in which at each stage of a dynamic planning process, a decision needs to be made on the sole basis of information (parameters) that pertain to the current stage only; examples are bin packing problems in which when a new object is to be packed, only information about its parameters (size, dimensions) is available and no information about any objects to be packed in the near future. The focus in designing and analyzing such algorithms is in characterizing their competitive ratio, defined as the maximum over all possible input sequences or expected value over assumed patterns of input sequences of the ratio between the cost incurred by the on-line algorithm and the cost incurred by an optimal off-line algorithm.

The approach taken in this paper is to adopt an intermediate, and in many settings, more realistic view between the two extremes adhered to by on-line algorithms (i.e. no future information) and standard off-line algorithms (i.e. complete information about the future). The view adopted here is that at any stage of the planning process information is available about a limited future horizon of (say m) stages, thus allowing for the execution of m iterations of the forward algorithm with the verification of whether a forecast horizon of length less than or equal to m prevails. If so, initial decisions can be made without any loss of optimality in spite of the limited amount of future information. In summary, the focus here is in answering the question: "how much of the future needs to be known or forecasted" rather than the central question addressed in connection with on-line algorithms: "how much is it worth to know the complete future as opposed to lacking any future information?" Our experience with forecast horizons in dynamic lot-sizing models with linear or fixed plus linear cost structures is that minimal forecast horizons are very short indeed (including no more than two or three order cycles); see Federgruen and Tzur (1994) for details.

2. Dynamic Programs with a Single Indicator for the Future

Consider a given deterministic dynamic program and assume it can be represented as a shortest path problem in an acyclic network with node set $\mathbb{N} = \{0, 1, \ldots, n\}$. We assume that an arc exists between every pair of nodes (i, j) with $i < j$ with a cost value $c(i, j)$. ($c(i, j)$ may be infinite if the transition from i to j is infeasible.)

Most dynamic programming models assume that *all* arc costs $\{c(i, j): 0 \le i < j \le n\}$ are known upfront. In many practical settings, information about these cost values is however obtained progressively as the planning process proceeds. Examples include the production/inventory planning problems discussed in §§3 and 4; these planning models are used in practice in conjunction with a forecasting system which at any point in time generates estimates of cost and sales volumes over a relatively short forecast interval only. The standard view of full parameter information until stage n being available upfront, is therefore inappropriate in these settings.

In general, we need to specify *how much* information about "future" parameters is available as the planning process proceeds. To this end, we introduce a nested sequence of *information sets*:

$$\mathbb{I}(0) \subseteq \mathbb{I}(1) \subseteq \mathbb{I}(2) \subseteq \cdots \subseteq \mathbb{I}(n) = \mathbb{I}^*.$$

The set $\mathbb{I}(t)$ consists of all information available when reaching node t. This set may contain past and future parameter values as well as structural information, e.g. the specific functional form of current and future cost components. To indicate that a specific parameter or functional relationship is known upon reaching node t, we will say that it can be viewed as the outcome of a mapping from the information set $\mathbb{I}(t)$.

DEFINITION 1. A quantity c is said to be $\mathbb{I}$-measurable with respect to an information set $\mathbb{I} \subseteq \mathbb{I}^*$ if $c = c(\mathbb{I})$ i.e., c can be viewed as the outcome of a mapping from $\mathbb{I}$ into $\mathbb{R}$.

Let Φ denote the set of all real-valued functions defined on the real line.

DEFINITION 2. A function $f \in \Phi$ is said to be $\mathbb{I}$-measurable with respect to an information set $\mathbb{I} \subseteq \mathbb{I}^*$ if f can be viewed as the outcome of a mapping ψ from $\mathbb{I}$ to Φ, i.e., $f = \psi(\mathbb{I})$.

We assume that every arc cost $c(i, j)$ is $\mathbb{I}(j)$-measurable ($1 \le i < j \le n$). In other words we merely assume that upon reaching stage j the costs on all arcs entering node j are known. We use the following notation. Let

$F(t)$ = the cost of the shortest path from node 0 to node t; ($t \in \mathbb{N}$).

The shortest path is clearly characterized by specifying for any node t in $\mathbb{N}$ its optimal predecessor node $l(t)$. (If several nodes arise as optimal predecessor nodes, we apply an arbitrary rule to break ties and identify $l(t)$.) Thus, let

$F(k, t)$ = the cost of the shortest path from node 0 to node t, using node k as the predecessor of t; ($1 \le k < t \le n$).

The values $\{F(t): t = 1, \ldots, n\}$ clearly satisfy the recursion

$$F(t) = \min_{k<t} F(k, t) = \min_{k<t} \{F(k) + c(k, t)\}.$$

For certain applications, we need a generalization of this recursion where

$$F(t) = \min_{k \le t} \{E(k) + c(k, t)\} \tag{1}$$

with $E(k)$ an arbitrary $\mathbb{I}(k)$-measurable quantity. In this generalization we may need quantities $c(t, t)$, again assumed to be $\mathbb{I}(t)$-measurable. Examples include the dynamic program required to solve lot-sizing models with backlogging, as described in §4, as well as many problems in the computer science literature, for example the modified edit distance problem discussed in §5. We henceforth consider this generalized recursion while continuing to refer to it as a shortest path problem.

When determining which of a given pair of nodes k and l ($k < l \in \mathbb{N}$) is preferred as the best node preceding node $t \in \mathbb{N}$, it is useful to consider the difference function $\Delta_{k,l}: \mathbb{N} \to \mathbb{R}$ defined by:

$$\Delta_{k,l}(t) = F(k, t) - F(l, t), \quad t \in \mathbb{N}.$$

We consider the class of dynamic programs in which the nodes in $\mathbb{N}$ can be characterized by a *single* indicator $X: \mathbb{N} \to \mathbb{R}$ such that all difference functions $\Delta_{k,l}(\cdot)$ can be written in the form

$$\Delta_{k,l}(t) = \delta_{k,l}(X(t)) \tag{2}$$

where the function $\delta_{k,l}(\cdot)$ is $\mathbb{I}(l)$-measurable and $X(t)$ is $\mathbb{I}(t)$-measurable. In other words, upon reaching node l, for *any* future node $t > l$, the difference function

$\Delta_{k,l}(t)$ can be evaluated given the single value $X(t)$. In particular, upon reaching node l and given the single indicator value $X(t)$, it can be determined whether node l is to be preferred to node k as the predecessor of any "future" node t. For example, in many of the production/inventory planning problems discussed in §§3 and 4, the difference functions can be written as in (2) with $X(t) = D(t)$, the cumulative demand up to time t; in others, this representation can be achieved with $X(t) = \hat{C}(t)$, an aggregate indicator which depends on the variable order cost rate at time t, and the backlogging rate up to that time. There are many planning problems in molecular biology and computer science, for example the modified edit distance problem discussed in §5, in which $X(t) = t$ arises as the indicator.

In §2.1 we describe a list-based solution method which determines for every node t an optimal predecessor $l(t)$. In §2.2 we show that this algorithm can be used, with a minor modification, for the identification of minimal forecast horizons. The algorithm's complexity is analyzed in §2.3.

2.1. A List-based Solution Method

While generally applicable, our proposed list-based solution method is *computationally* competitive to standard shortest path algorithms if the difference functions satisfy the following condition.

CONDITION (C). All difference functions $\delta_{k,l}$ are continuous and there exists an integer $R \geq 1$ such that all difference functions $\delta_{k,l}(\cdot)$ have at most R roots, or $\delta_{k,l}(X(t)) = 0$ for all $X(t)$. (A root is a point where the difference function equals zero and changes signs.)

Several articles have recently appeared in the Computer Science literature, describing efficient algorithms for the solution of dynamic programs in which *all* difference functions are *increasing*, or in which *all* are *decreasing*. These two cases are usually referred to as *concave* and *convex* dynamic programs, respectively. Note that in both cases, condition (C) applies with $R = 1$, see for example Hirshberg and Larmore (1987), Wilber (1988), Aggarwal et al. (1987), Galil and Giancarlo (1989) and Miller and Myers (1988). See Park (1992, Chapter 2) for many more recent examples. In standard dynamic lot-sizing models with fixed-plus-linear order and linear holding costs, one observes that all difference functions are monotone, but some may be increasing while others are decreasing, see Federgruen and Tzur (1991). In other words, Condition (C) with $R = 1$ continues to apply.

If all difference functions are polynomials in $X(t)$, of uniformly bounded degree R, Condition (C) clearly applies as well; more generally, this applies when the difference functions are given as piecewise combinations of convex and concave functions with a uniformly bounded number of pieces, s. (It is easily verified that (C) applies with $R \leq 2s$.) We note that most numerical approximation methods for nonlinear functions employ piecewise polynomial representations, e.g. splines.

Thus, for any pair of nodes $k < l$ let $G_r(k, l)$ denote the rth smallest root of the difference function $\delta_{k,l}(\cdot)$, with the convention that $\delta_{k,l}(x) \leq 0$ for $x \leq G_1(k, l)$. (If necessary, set $G_1(k, l) = -\infty$.) Note that with this convention, $\delta_{k,l}(x) \geq 0$ (i.e. l dominates k as a predecessor of any node t with $X(t) = x$) if x lies between the two roots $G_r(k, l)$ and $G_{r+1}(k, l)$ with r odd.

The list-based algorithm is a forward procedure with one iteration associated with each node. The algorithm constructs in the jth iteration an ordered list of nodes $\Omega(j) = (i_1, \ldots, i_m)$ and an associated ordered set of critical indicator-values

$$G(j) = (-\infty = g(1) < g(2) < \cdots < g(m+1) = +\infty)$$

with the following interpretation:

DEFINITION 3. Fix $0 \leq j \leq n$. Let $\Omega(j) = (i_1, \ldots, i_m)$ and

$$G(j) = (-\infty = g(1) < g(2) < \cdots < g(m+1) = +\infty).$$

Node i_k is an optimal predecessor node, among nodes $0, \ldots, j$, for any later node $t > j$ with an indicator value

$$x = X(t) \in [g(k), g(k+1)] \quad (k = 1, \ldots, m).$$

This definition assumes that, for any node j, no restrictions on the possible values of $X(t)$ for $t > j$ arise from the information contained in the set $\mathbb{I}(j)$. The algorithm is modified to reflect any such restrictions by eliminating from the list $\Omega(j)$ any node i_k if $\mathbb{I}(j)$ implies that $X(t) \in [g(k), g(k+1)]$ is not possible for any $t > j$, see Remark 1 below.

We note that the same node index may appear multiple times in the list $\Omega(j)$. For every $j \geq 0$ the lists $\Omega(j)$

and $G(j)$ provide a complete characterization of which of the nodes $0, \ldots, j$ arises as an optimal predecessor node for any potential later node $t > j$. In particular, we have the following interpretation: $\{i_1, \ldots, i_m\} = \{0 \le l \le j$: there exists a node $t > j$ with a potential indicator-value $x = X(t)$ with $F(l, t) < F(i, t)$, $i = 0, \ldots, j, i \neq l\}$. As will be shown below, $g(k)$ is a root of the difference function δ_{i_{k-1}, i_k} for all $k = 2, \ldots, m$.

Note that, since $X(j)$ is $\mathbb{I}(j)$-measurable, it is known upon reaching node j. We therefore assume that $X(j)$ is known at the beginning of the jth iteration, enabling us to identify an index k such that $X(j) \in [g(k), g(k+1)]$ and hence $i_k = l(j)$ by Definition 3. The remainder of the jth iteration consists of updating the lists $\Omega(j-1) = \{i_1, \ldots, i_m\}$ and

$$G(j-1) = (-\infty = g(1) < g(2) < \cdots < g(m+1) = +\infty)$$

obtained in the $(j-1)$st iteration as follows. Consider a given (say the kth, $1 \le k \le m$) element in the list $\Omega(j-1)$. Let $R_k \le R$ denote the number of roots of the difference function $\delta_{i_k, j}(\cdot)$ and let $G_l = G_l(i_k, j)$, $l = 1, \ldots, R_k$. (Set $G_{R_k+1} = +\infty$.) The following observations follow directly from Definition 3 and the numbering convention for the roots of the difference functions:

$$X(t) = x \in \bigcup \{([g(k), g(k+1)] \cap [G_l, G_{l+1}]): l \le R_k \text{ and } l \text{ is odd}\} \Rightarrow F(j, t) \le F(i, t), \quad i = 0, \ldots, j, \tag{3a}$$

$$X(t) = x \in [g(k), g(k+1)] \backslash \bigcup \{[G_l, G_{l+1}]: l \le R_k \text{ and } l \text{ is odd}\} \Rightarrow F(i_k, t) \le F(i, t), \quad i = 0, \ldots, j. \tag{3b}$$

(If $x \in [G_l, G_{l+1}]$ with l odd, node j dominates node i_k as a last predecessor, and if $x \in [g(k), g(k+1)]$ as well, node i_k dominates every other node $0, \ldots, j-1$, thus verifying (3a). Likewise, if

$$x \in [g(k), g(k+1)] \backslash [G_l, G_{l+1}]$$

for l odd, i_k dominates all nodes $0, \ldots, j-1$, as well as node j.)

Thus, (3) fully characterizes for all $t > j$ with $g(k) \le x = X(t) \le g(k+1)$ which of the indices $0, \ldots, j$ minimizes $\{F(i, t), i = 0, \ldots, j\}$. A full characterization, for all x, can be obtained by applying (3) sequentially to all $k = 1, \ldots, m$. Each such application invokes determining the union of the intersections of pairs of intervals; the kth determination could potentially be sped up by keeping track of intervals on which j is dominated by one of the periods $i_1, \ldots, i_{k-1}$ as determined in the first $(k-1)$ applications of (3). In general, this fails to pay off; see however the case $R = 2$ below. On the other hand, it is worthwhile to update a value UP such that for all $x \ge$ UP, j has been shown to be dominated. In particular, if R_k is even, it follows from (3) that j is dominated for all $x \ge G_{R_k}$ so that UP can be reduced to G_{R_k} (if UP $> G_{R_k}$).

We thus obtain the algorithm below, which employs the following conventions and procedures: Ω^{old} and G^{old} represent the Ω and G lists at the beginning of an iteration. New elements of the updated lists are written into Ω^{new} and G^{new}. $i^{\text{old}}(k)$ and $i^{\text{new}}(k)$ represent the kth element in the lists Ω^{old} and Ω^{new}, respectively.

NEXT represents the next position to be filled in $\Omega^{\text{new}}(\cdot)$; LAST $= |\Omega^{\text{old}}|$. As explained above, UP is the largest value of x that still needs to be considered at the current iteration. We use an $n \times R$ matrix $MR(\cdot, \cdot)$ to store in each iteration j the (at most) R roots of the difference functions $\delta_{i^{\text{old}}(k), j}(\cdot)$ for all $i^{\text{old}}(k)$, k = FIRST, $\ldots$, LAST.

Procedures

```
INSERT ($j$, NEXT, $x$): if $i^{new}$(NEXT-1) $\neq j$ then
    *begin* $i^{new}$(NEXT):=$j$, $g^{new}$(NEXT):=$x$, NEXT:=
    NEXT + 1 *end*.
MOVEALL (NEXT, $k$): while $g^{old}(k+1) < \infty$ *do*
    $i^{new}$(NEXT):=$i^{old}(k)$; $g^{new}$(NEXT):=$g^{old}(k)$;
    $k$:=$k$+1; NEXT:=NEXT+1 *end* $g^{new}$(NEXT):=$\infty$.
```

Each iteration starts with the execution of Step 1. Input to the jth iteration is the information set $\mathbb{I}(j)$. Recall that $\mathbb{I}(j)$ contains all $c(i, l)$ and all $\delta_{i,l}(\cdot)$ for $0 \le i \le l \le j$, as well as $X(l)$ and $E(l)$, $l = 1, \ldots, j$.

Algorithm GENERAL

```
*Step* 0: (*Initialization*)
$F(0)$:=0; $i^{new}(1)$:=0; $g^{new}(1)$:=$-\infty$; $g^{new}(2)$:=$\infty$; $j$:=0;
$k$:=1; LAST:=1; NEXT:=2.
```

```
Step 1:
$j := j+1$;
$l(j) := i_m$ such that $g^{\text{new}}(m) \le X(j) < g^{\text{new}}(m+1)$, $m = 1, \ldots, \text{LAST}$;
$F(j) := E(l(j)) + c(l(j), j)$;
If $j = n$ then STOP.

Step 2:
LAST:=NEXT-1;
for $l$:=1,LAST do begin $i^{\text{old}}(l) := i^{\text{new}}(l)$; $g^{\text{old}}(l) := g^{\text{new}}(l)$;
end
NEXT:=1; UP:=$\infty$;

Step 3:
if ($i^{\text{old}}(k) \ne i^{\text{old}}(l)$ for all $l < k$)
  then begin Compute the roots $G_1, G_2, \ldots G_{R_k}$ of $\delta_{i^{\text{old}}(k),j}(x)$ and store them in $MR(\cdot,\cdot)$; $G_{R_k+1} = \infty$; end
  else get the roots $G_1, G_2, \ldots G_{R_k}$ of $\delta_{i^{\text{old}}(k),j}(x)$ from MR$(\cdot,\cdot)$;
if (($R_k$ is even) and ($G_{R_k} <$ UP)) then UP = $G_{R_k}$.
$p := \min\{1 \le l \le R_k+1 : G_l > g^{\text{old}}(k)\}$.

Step 4:
If ($p$ is odd) then INSERT($i^{\text{old}}(k)$, NEXT, $g^{\text{old}}(k)$);
  otherwise INSERT($j$, NEXT, $g^{\text{old}}(k)$);
if ($G_p > g^{\text{old}}(k+1)$) then go to Step 5.
$g^{\text{old}}(k) := G_p$; $p := p+1$; Repeat step 4.

Step 5:
$k := k+1$;
if $g^{\text{old}}(k) >$ UP then begin MOVEALL(NEXT, $k$); go to Step 1 end.
go to Step 3.
```

At the beginning of iteration j the algorithm generates $l(j)$ and $F(j)$ as outputs, from which the shortest path from node 0 to node j (and its cost function) can be determined.

REMARK 1. As mentioned above, for any $j = 1, \ldots, n$, the information contained in $\mathbb{I}(j) \setminus \mathbb{I}(j-1)$ may imply some new restrictions for the feasible region X of future index values. In such cases the jth iteration needs to start with determining for k = FIRST, ..., LAST the intersection of X and the interval $[g(k), g(k+1)]$, eliminating i_k and $g(k)$ from the list if this intersection is empty. As a result, i_{k-1} is now associated with the interval $[g(k-1), g(k+1)]$ of the previous list; observe that i_{k-1} continues to be the optimal predecessor node among $\{0, \ldots, j\}$ for any *feasible* value $X(t) \in [g(k-1), g(k+1)]$. For example, if $X(t) = t$, it is known upon reaching node l (i.e., it is part of the information set $\mathbb{I}(l)$) that $X(t)$ is an integer $>l$ for all $t > l$. Thus, i_k can be eliminated from $\Omega(j)$ if no such integer is contained in $[g(k), g(k+1)]$. In the examples where $X(t) = D(t)$ (the cumulative demand for an item up to period t), it is known upon reaching node l that $X(t) \ge D(l)$ for all $t > l$ if the demand in every period is nonnegative; this allows for the elimination of all nodes $i_k \in \Omega$ with $g(k+1) \le D(l)$, and replacement of $g(k)$ by $D(l)$ for the (at most unique) element i_k with $g(k) < D(l) \le g(k+1)$. With this modification of the algorithm GENERAL, a revised definition of the list Ω applies:

DEFINITION 3′. Fix $0 \le j \le n$. Let $\Omega(j) = (i_1, \ldots, i_m)$ and

$$G(j) = (-\infty = g(1) < g(2) < \cdots < g(m+1) = +\infty).$$

$[g(k), g(k+1)]$ has a nonempty intersection with X, the feasible region of future indicator values (as specified by $\mathbb{I}(j)$) and node i_k is an optimal predecessor node, among nodes $0, \ldots, j$, for any later node $t > j$ with a *feasible* indicator value

$$x = X(t) \in [g(k), g(k+1)] \quad (k = 1, \ldots, m).$$

2.2. Detecting Minimal Forecast Horizons

A minor modification of algorithm GENERAL allows for the detection of a minimal forecast horizon. A node L is called a *forecast horizon* if the optimal path from node 0 to any node $t > L$ goes through node l (for *any* possible arc costs $\{c_{rs} : s > L\}$). Such a node l is referred to as a *planning* horizon. A node $L^* = \min\{L : L$ is a forecast horizon$\}$ is the *minimal* forecast horizon.

We assume for the sake of notational simplicity, that $l(j)$, the optimal last predecessor of node j, is unique for all $j = 1, \ldots, n$. This assumption implies that a *unique* optimal path exists to every node $j = 1, \ldots, n$. Almost all of the existing literature on forecast horizons employs a similar assumption on the existence of a unique shortest path, throughout. It is easy to extend our results to the general case where *multiple* finite horizon optimal solutions may exist, see Federgruen and Tzur (1994). Let

$q(j)$ = the *first* node after node 0 on the optimal path to j $(j = 1, \ldots, n)$;

$q(0) = 0$.

We first derive the necessary and sufficient condition for a node j to be a forecast horizon.

THEOREM 1. *Fix $j = 1, \ldots, n$. Let $\Omega(j) = \{i_1, i_2, \ldots, i_m\}$. Node j is a forecast horizon if and only if*

$$0 < q^* \equiv q(i_1) = q(i_2) = \cdots = q(i_m). \tag{4}$$

If (4) holds, then q^* is a planning horizon.

PROOF. Assume first that (4) holds. Consider the optimal path to a given node $t > j$, let t^- denote the first node on this path with $t^- > j$, and $p \leq j$ its predecessor. Clearly, $p \in \Omega(j)$ since p is an optimal last predecessor among nodes $0, \ldots, j$ for t^-. We conclude that the optimal path to node t goes through $q(p) = q^*$. Arc $(0, q^*)$ is therefore an optimal initial arc, regardless of the arc costs $\{c(k, l): j < l\} \notin \mathbb{I}(j)$. Therefore j is a forecast horizon and q^* the associated planning horizon.

Conversely, assume that (4) fails to hold but that node j is a forecast horizon nevertheless. This implies that the optimal initial arc is independent of any arc costs $\{c(k, l): j < l\}$. Thus, assume arc $(0, r)$ (for some $r \geq 1$) is the initial arc on the optimal path to every potential future node $t > j$. Since (4) fails to hold, there exists an element $i_k \in \Omega(j)$ with $q(i_k) \neq r$. By definition 3 (or its more general version 3′), there exists an *unknown feasible* future indicator value $X(j + 1) \in (g(k), g(k + 1))$ for which i_k is the predecessor of node $(j + 1)$ on the optimal path to node $(j + 1)$, and arc $(0, q(i_k)) \neq (0, r)$ is the initial arc on this path, a contradiction. □

Theorem 1 suggests an extremely simple procedure for the detection of a minimal forecast horizon and its associated planning horizon. Note that

$$q(j) = \begin{cases} j & \text{if } l(j) = 0, \\ q(l(j)) & \text{otherwise.} \end{cases} \quad (j = 1, \ldots, n) \tag{5}$$

Algorithm GENERAL is thus easily adapted to allow for the detection of a minimal forecast horizon. Having computed and stored the values $q(1), \ldots, q(j - 1)$, $q(j)$ is easily computed via (5). Thus, to verify whether node j is a forecast horizon, it suffices to conclude the jth iteration with a test of whether (4) holds or not. This can be done efficiently by keeping track of the multiplicity of each distinct q-value among all q-values associated with nodes which appear in Ω, and by testing at the end of the jth iteration whether the multiplicity of $q(i_1)$ equals the number of (distinct) nodes in the prevailing list Ω. The multiplicities of the q-values only need to be updated when a node is deleted or added to the list Ω. There are at most $2n$ such deletions and additions of (distinct) nodes in n iterations and the update associated with an addition (deletion) consists of increasing (decreasing) the multiplicity of a *single* q-value by one. We conclude that the forecast horizon test can be performed in each of the first n iterations, with a total of $O(n)$ elementary operations in addition to those required by the algorithm GENERAL, see below. We refer to Algorithm GENERAL combined with the forecast horizon test in each iteration (including the necessary updates of the q-values) as algorithm GENFOR.

In the case of multiple finite horizon optimal solutions, $q(j)$ represents a set of optimal first nodes. In this case j is a forecast horizon if and only if the q-sets associated with the elements in $\Omega(j)$ have a common element. Verifying this condition can again be done in $O(n)$ time, see Federgruen and Tzur (1994) for details.

In some settings it is required or desired to obtain a planning horizon which is at least as large as a prespecified number of nodes. As in Federgruen and Tzur (1994), we refer to such a planning horizon as the *stability horizon*. The above procedure is easily adjusted: Define

$$q_s(j) = \begin{cases} \text{the } \textit{first} \text{ node after node } s \\ \text{on the optimal path to node } j, \\ \qquad j = s + 1, \ldots, n, \\ 0 \quad \text{otherwise.} \end{cases}$$

Clearly $q_0(j) = q(j)$ for all $j = 1, \ldots, n$. In analogy to (5) we obtain:

$$q_s(j) = \begin{cases} j & \text{if } l(j) \leq s \quad (j = s + 1, \ldots, n), \\ q(l(j)) & \text{otherwise.} \end{cases}$$

Thus, (5) with $q(\cdot)$ replaced by $q_s(\cdot)$ represents the necessary and sufficient condition for a node j to be the minimal forecast horizon associated with a stability horizon s.

2.3. Complexity Analysis

We now discuss the complexity of algorithm GENERAL. We confine ourselves to the case where no restrictions on the possible values of $X(t)$ for $t > l$ arise from the information contained in the set $\mathbb{I}(l)$. The complexity analysis can be appropriately adjusted in the alternative case. A single call to the INSERT procedure requires a constant number of elementary operations. The jth iteration requires, in the worst case, calculation of up to R roots of j difference functions δ and $O(R|\Omega(j-1)|)$ additional elementary operations. (Note that Step 3 is repeated at most $|\Omega(j-1)|$ times and an execution of Step 3 is followed by at most R consecutive executions of Step 4.) The total complexity of the algorithm is therefore $O(R\sum_{j=1}^{n}|\Omega(j-1)|)$ elementary operations and $O(n^2R)$ computations of roots of difference functions. (Efficient procedures for the calculation of roots of a general non-linear function can be found e.g. in Leavenworth (1960) and Muller (1956), see also IMSL (1991) and Conte and De Boor (1972) for a reference text.) We now derive an upper bound for the list size.

DEFINITION 4. A substring $(i_{l_1}, \ldots, i_{l_p})$ of $(i_1, \ldots, i_m)$ is called *alternating* if it consists of a pair of nodes a and b such that $(i_{l_1}, \ldots, i_{l_p}) = (a, b, a, b, \cdots)$ for some pair of nodes $1 \le a, b \le j$.

LEMMA 1. *Fix $j \le n$. The ordered set $\Omega(j) = (i_1, \ldots, i_m)$ contains no alternating substrings of length $R+2$.*

PROOF. Consider a substring $(i_{l_1}, \ldots, i_{l_p}) = (a, b, a, b, \cdots)$ for some pair of nodes a, b. Every consecutive pair of nodes $i_{l_r}, i_{l_{r+1}}$ in the substring is associated with a pair of intervals

$$[g(l_r), g(l_r+1)] \quad \text{and} \quad [g(l_{r+1}), g(l_{r+1}+1)]$$

such that $\delta_{a,b}$ (if $a < b$) or $\delta_{b,a}$ (if $b < a$) is negative on one of the two intervals and positive on the other, since in the former l_r dominates as a last predecessor node and in the latter l_{r+1}. By the continuity of δ (see condition (C)) it follows that δ has a root in between these intervals. It follows that every consecutive pair of elements $(i_{l_r}, i_{l_{r+1}})$ is associated with a *distinct* root of the difference function $\delta_{a,b}$ (or $\delta_{b,a}$). The maximum alternating substring is therefore of length $R+1$. □

It follows from Lemma 1 that for some function $\lambda_R(\cdot)$, $|\Omega(j)| \le \lambda_R(j+1)$, the maximum length of a string with up to $j+1$ distinct elements where no consecutive pair of elements is identical and no alternating substring of length $R+2$ exists. Such strings were first introduced by Davenport and Schinzel (1965) and are since referred to as Davenport-Schinzel sequences. Davenport and Schinzel showed that $\lambda_2(j) \le 2j-1$ (see also Theorem 5.2 in Tzur (1992)). Example 1 shows that this bound is tight.

EXAMPLE 1. Consider the following string, containing j distinct elements:

$$(1, 2, 3, \ldots, j-2, j-1, j, j-1, j-2, \ldots, 3, 2, 1).$$

No two consecutive elements in this string are identical, there are no alternating substrings of length 4 and the length of the string is $2j-1$.

Szemerédi (1974) proved that for a given R, $\lambda_R(j) = O(j\log^* j)$ where $\log^* j = \min\{k: \exp_k(1) > j\}$ with $\exp_1(x) = e^x$ and $\exp_k(x) = \exp \exp_{k-1}(x)$ and where the constant factor depends on R. Note that $\log^* j \le 2$ for $j \le 15 < e^e$ and $\log^* j \le 3$ for $j \le 3{,}814{,}279 < e^{e^e}$. In other words, $\log^* j \le 3$ for all practical purposes.

Sharir (1987) proved that for general values of R, $\lambda_R(j) = O(j\alpha(j)^{O(\alpha(j)^{R-3})})$ where $\alpha(j)$ denotes the inverse Ackermann function, an extremely slowly growing function defined as follows:

$$A_1(m) = 2^{m+1}; \quad A_{k+1}(1) = A_k(2);$$
$$A_{k+1}(m+1) = A_k(A_{k+1}(m)).$$

Its functional inverse $\alpha(n) = \min\{k: n \le A_k(k)\}$. For astronomically large values of n it is a tighter bound than that of Szemerédi. Hart and Sharir (1986) showed that $\lambda_R(j)/j$ goes to infinity, i.e., $\lambda_R(j) \ne O(j)$. We thus conclude:

THEOREM 2. *Assume Condition (C) holds for some given integer R.*

(a) Algorithm GENERAL solves the shortest path problem.

(b) For $R > 2$ the algorithm requires $O(n^2\log^* n)$ elementary operations and $O(n^2)$ computations of roots of difference functions δ.

(c) For $R = 2$ the algorithm requires $O(n^2)$ elementary operations and $O(n^2)$ computations of roots of difference functions δ.

PROOF. Immediate from the above explanations. □

For $R = 2$, Algorithm GENERAL may be sped up by exploiting the following observation: Let $\Omega(j-1) = \{i_1,$

$\ldots, i_m\}$ with associated critical values $g(1), \ldots, g(m)$. Once a value G_1 (G_2) is identified as a root $G_1(i_l, j)$ ($G_2(i_l, j)$) for some $1 \le l \le m$, we know that node j cannot arise as an optimal last predecessor node (among the first j nodes) for any future horizon t with $X(t) = x < G_1$ or $x > G_2$. Thus, having characterized optimal predecessor nodes (among nodes $0, \ldots, j$) for all $x \le g(k+1)$ for some k $(1 \le k \le m)$, we know that node j can arise as the optimal last predecessor node only for

$$\mathrm{LOW}(k) = \max\{G_1(i_l, j): l = 1, \ldots, k\} < x < \mathrm{UP}(k) = \min\{G_2(i_l, j): l = 1, \ldots, k\}.$$

The update procedure can thus be terminated as soon as $\mathrm{LOW}(k) > \mathrm{UP}(k)$. Clearly,

$$\mathrm{LOW}(k+1) = \min\{\mathrm{LOW}(k), G_1(i_{k+1}, j)\} \quad \text{and}$$
$$\mathrm{UP}(k+1) = \min\{\mathrm{UP}(k), G_2(i_{k+1}, j)\},$$

allowing for easy updates of the LOW and UP values. The reader is referred to Tzur (1992, Chapter 5) for a formal description of the resulting variant of Algorithm GENERAL.

For $R = 1$, the algorithm may be modified so as to require only $O(n \log n)$ elementary operations and $O(n \log n)$ computations of single root difference functions δ. Note that in this case, by Lemma 1 no node can appear more than once in the list; in particular, $|\Omega(j)| \le \lambda_1(j+1) = j+1$. To update $\Omega(j-1)$ in the jth iteration, it thus suffices to find the single interval $[\underline{g}, \bar{g}]$ (if any) in which node j dominates over nodes $0, \ldots, j-1$ as a last predecessor node. Algorithm GENERAL has complexity $O(n^2)$ even when $R = 1$. In the Appendix we describe how the algorithm can be modified to achieve a complexity of $O(n \log n)$. We thus obtain:

PROPOSITION 1. *Assume Condition* (C) *applies with* $R = 1$.

(*a*) *Algorithm* GENERAL, *modified by replacing Steps 2–5 by the procedure* SRUP *in the Appendix, solves the dynamic programming model and requires $O(n \log n)$ determinations of roots of difference functions and $O(n \log n)$ additional elementary operations.*

(*b*) *If all difference functions $\delta_{k,l}(\cdot)$ are nondecreasing (nonincreasing), algorithm* GENERAL *may be modified to require $O(n)$ determinations of roots of difference functions and $O(n)$ additional elementary operations.*

PROOF. See the Appendix.

Finally, it is worth noting that the complexity of the algorithm consists of $O(n)$ determinations of roots and $O(n)$ additional operations if the number of *distinct* nodes in the list can be shown to be uniformly bounded in n.

COROLLARY 1. *Assume Condition* (C) *applies for some integer R. Assume that for all nodes t ($t = 1, \ldots, n$) an optimal path from node 0 exists in which no arc (i, j) is employed with $j - i > M$, for some constant integer $M \ge 1$. Algorithm* GENERAL *requires $O(n)$ determinations of roots and $O(n)$ additional operations.*

The condition of Corollary 1 can often easily be verified. We do this in §§3 and 4 for the lot-sizing models discussed there, proving a uniform bound for order cycles, merely assuming that certain marginal cost values and demand parameters are uniformly bounded from above or below. Our numerical experience with lot-sizing models indicates that the list size is very small. Observe also that no upfront bound for R or the maximum list size are needed when running the algorithm.

3. Dynamic Lot-sizing Models Without Backlogging: General Concave Order and Holding Cost Functions

In §§3 and 4 we discuss dynamic lot-sizing models without and with backlogging, respectively. We show how these models can be represented as dynamic programs with a single-indicator of the future, thus allowing for the application of Algorithm GENFOR.

The dynamic lot sizing model is one of the most frequently employed deterministic single item inventory planning models. The basic model was introduced by Wagner and Whitin (1958). It specifies a study 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. In the basic model without backlogging, the cost structure consists of fixed-plus-linear order (or production) costs and holding costs. Order costs thus consist of a fixed component incurred whenever an order is placed, and a variable component proportional to the order size.

The holding costs are assumed to be proportional with the end-of-the-period inventory level. All parameters, i.e. demands, setup costs, variable replenishment and holding cost rates may differ from period to period. Wagner and Whitin (1958) developed an $O(n^2)$ shortest path algorithm to solve the problem.

New solution methods, with complexity $O(n \log n)$, were recently developed for the basic problem by Aggarwal and Park (1993), Federgruen and Tzur (1991), and Wagelmans et al. (1992). Federgruen and Tzur (1994) also showed how, with a slight modification of their algorithm, minimal forecast horizons can be detected.

In this section we discuss dynamic lot-sizing models without backlogging in which the order and holding costs are described by general concave functions of the order quantity and inventory level. This cost structure allows for the modeling of general economies of scale as they apply to production, order and inventory costs, e.g., economies of scale due to quantity discounts.

We note that the basic structural properties of an optimal strategy, as employed in the basic model, continue to hold under general concave cost structures. More specifically, a zero inventory ordering policy in which an order is placed in period i only if the ending inventory of period $i - 1$ is zero, remains optimal, see Zangwill (1968) and Denardo (1982). A zero-inventory ordering strategy is completely determined by the specification of $l(t)$, the last period with zero ending inventory preceding any given horizon t ($t = 1, \ldots, n$). On the basis of this property, the Wagner-Whitin shortest path algorithm has been directly generalized to handle general concave cost functions, see Zangwill (1969) and Denardo (1982). In this section we show how the framework developed in §2 can be applied to the dynamic lot-sizing model with general concave cost functions. The resulting list-based algorithms are a generalization of the $O(n \log n)$ solution method by Federgruen and Tzur (1991), for the basic model. The main advantage of these algorithms is their ability to detect minimal forecast horizons, if they exist, in the presence of general nonlinear (concave) cost functions. In the voluminous literature on forecast horizon for lot-sizing models, Bensoussan et al. (1991) make the most general cost assumptions to date, but restrict themselves to piecewise linear (concave) functions only. If the number of pieces required to describe the one-period cost function is bounded, the complexity of their algorithm is $O(n^3)$.

In §3.1 we discuss models without backlogging in which every period's order cost function is piecewise linear (concave) and the holding cost function is arbitrary (concave). In §3.2 we generalize the model to allow for general concave order cost functions.

3.1. Lot-sizing Models Without Backlogging: Piecewise Linear Concave Order Cost and Arbitrary Concave Holding Cost Functions

We use the following notation: For all $i = 1, \ldots, n$, let

d_i = demand in period i; we assume without loss of generality that $d_i \geq 0$;

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

We initially assume that the order cost functions are fixed-plus-linear and discuss the generalization to general piecewise linear concave functions at the end of this subsection. Thus,

K_i = setup cost in period i;

c_i = variable per unit order cost in period i;

$h_i(I_i)$ = holding cost in period i when its ending inventory equals I_i (assume, without loss of generality, that the $h(\cdot)$ functions are normalized such that $h_i(0) = 0$. We also assume that the $h(\cdot)$ functions are nondecreasing.)

As in §2, let

$F(t)$ = minimum total cost in periods $1, \ldots, t$,

$F(l, t)$ = minimum total cost in periods $1, \ldots, t$ given that the last period with zero ending inventory (prior to t) is l ($l < t$).

Thus, $F(t)$ represents the cost of a shortest path from node 0 to node t in a network with node set $\mathbb{N} = \{0, \ldots, n\}$ and with $c(l, t)$, the cost of arc (l, t), the total order and holding costs in periods $l + 1, \ldots, t$, if the ending inventory in periods l and t both equal zero (and nowhere in between). One can easily verify that

$$F(l, t) = F(l) + K_{l+1} + c_{l+1}(D(t) - D(l)) + \sum_{r=l+1}^{t} h_r(D(t) - D(r)). \quad (6)$$

$$\text{Thus, } F(t) = \min_{l<t} F(l, t). \quad (7)$$

For all $k < l$,

$$\Delta_{k,l}(t) = F(k,t) - F(l,t) = F(k) - F(l) + K_{k+1} - K_{l+1} + c_{l+1}D(l) - c_{k+1}D(k) + (c_{k+1} - c_{l+1})D(t) + \sum_{r=k+1}^{l} h_r(D(t) - D(r)).$$

We specify the information sets $\mathbb{I}(t)$ as follows:

$$\mathbb{I}(0) = \{K_1, c_1\} \quad \text{and}$$

$$\mathbb{I}(t) = \mathbb{I}(t-1) \cup \{d_t, h_t(\cdot), K_{t+1}, c_{t+1}\} \quad \text{for } t = 1, \dots, n.$$

In other words, upon reaching node t (which corresponds with the end of period t), both that period's demand and its inventory cost function are known, as well as the cost of any order to be placed at the beginning of period $t + 1$. If a dynamic lot-sizing model is to be solved with a study horizon of n periods, this assumes that at the beginning of period 1, all information in $\mathbb{I}(n)$ is known. The implication of this knowledge base is that at the beginning of period 1, n iterations of the algorithm GENFOR can be executed; if in the course of these n iterations a (no) minimal forecast horizon is detected, this therefore implies that the corresponding (no) planning horizon can be identified at the beginning of period 1. Furthermore, the common practice of determining lot sizes on the basis of a *rolling* horizon of n periods means that at the end of every period j ($j = 1, 2, \cdots$) the information set $\mathbb{I}(j + n)$ is known, allowing for the execution of iterations $j + 1, \dots, j + n$ of the algorithm.

We observe that the difference functions depend on t only via $D(t)$, i.e., $D(t)$ is the *single-indicator* of the future of the dynamic program (7). In other words, the difference functions can be written as in (2) with $X(t) = D(t) \in \mathbb{I}(t)$, as follows:

$$\Delta_{k,l}(t) = \delta_{k,l}(D(t)) \quad \text{where}$$

$$\delta_{k,l}(x) = A(k,l) + (c_{k+1} - c_{l+1})x + \sum_{r=k+1}^{l} h_r(x - D(r)) \quad \text{and}$$

$$A(k,l) = F(k) - F(l) + K_{k+1} - K_{l+1} + c_{l+1}D(l) - c_{k+1}D(k). \tag{8}$$

We note that all $\delta_{k,l}(\cdot)$ functions are $\mathbb{I}(l)$-measurable; moreover, they are *concave* since all $h_r(\cdot)$ functions are ($r = 1, \dots, n$). This implies in particular that the $\delta(\cdot)$ functions have at most two roots, i.e. Condition (C) is satisfied with $R = 2$.

Since one-period demands are assumed to be nonnegative, it follows that for all $t > l$, $X(t) = D(t) \geq D(l)$. In other words, the information contained in the set $\mathbb{I}(l)$ implies a restriction on possible values of future indicators. Our procedures below are based on this observation.

Thus, for any pair of periods $0 \leq k < l \leq n$ let

$G_1(k, l) < G_2(k, l)$ denote the two roots of the function $\delta_{k,l}(\cdot)$ on the half line $[D(l), \infty)$ if two such roots exist. (9a)

If $\delta_{k,l}(\cdot)$ has a single root r^* on the half line $[D(l), \infty)$ we distinguish between the following two cases in defining $G_1(k, l)$ and $G_2(k, l)$:

$$G_1(k,l) = D(l) \quad \text{and} \quad G_2(k,l) = r^* \quad \text{if } \delta_{k,l}(D(l)) \geq 0, \tag{9b}$$

$$G_1(k,l) = r^* \quad \text{and} \quad G_2(k,l) = \infty \quad \text{if } \delta_{k,l}(D(l)) < 0. \tag{9c}$$

Finally, if $\delta_{k,l}(D(t)) \geq 0$ for all $D(t) \geq D(l)$, we define

$$G_1(k,l) = D(l) \quad \text{and} \quad G_2(k,l) = \infty. \tag{9d}$$

If $\delta_{k,l}(\cdot) < 0$ throughout, then

$$G_1(k,l) = G_2(k,l) = \infty. \tag{9e}$$

Since for this model Condition (C) is satisfied with $R = 2$, we have in view of Theorem 2(c):

THEOREM 3. *The algorithm* GENFOR *solves the lot-sizing problem* (7), *and detects a minimal forecast horizon. It consists of the computation of* (*at most two*) *roots of at most* $O(n^2)$ *difference functions and* $O(n^2)$ *additional elementary operations.*

Note that since the difference functions are concave, roots can be determined, e.g., by a simple binary search procedure, starting with the prevailing interval [LOW, UP], see §2.

Theorem 3 characterizes the complexity of the algorithm GENFOR under fully general holding cost functions, and arbitrary order cost and demand parameters. Corollary 1 shows however that the algorithm has *linear* complexity only, if the list size is uniformly bounded. In §3.2 we show this (in a more general setting) under mild conditions with respect to the cost functions and parameter values. For the models considered here, this implies the following proposition: Assume all demands are rational, hence integer after appropriate scaling. For any real-valued function $\phi(x)$, let $\nabla^+\phi(x)$ denote the right-hand derivative of the function (whenever it exists), i.e.,

$$\nabla^+\phi(x) = \lim_{h\downarrow 0} \frac{\phi(x+h) - \phi(x)}{h}.$$

Recall that for every concave function, the right-hand derivative exists everywhere (see Rockafellar 1970).

PROPOSITION 2. *Assume there exists an integer* $M \geq 1$, *and uniform bounds* h_*, K^*, c_* *and* c^* *such that* $(d_i + \cdots + d_{i+M}) \geq 1$, $c_i \geq c_*$, $K_i \leq K^*$, $c_i \leq c^*$ *and* $\underline{\lim}_{x\to\infty} \nabla^+ h_i(x) \geq h_* > 0$ *for all* $i = 1, \ldots, n$.

(*a*) *The size of the list* Ω *is bounded from above by*

$$2\left(\frac{K^*}{h_*} + \frac{(c^* - c_*)}{h_*} + M\right);$$

(*b*) *The algorithm requires* $O(n)$ *operations.*

Piecewise Linear Concave Order Cost Functions. Assume now that for all $i = 1, \ldots, n$, $c_i(x)$, the cost of an order of size x in period i is given by a piecewise linear function, described via P_i pieces. In Federgruen and Tzur (1992) we show that any model with piecewise linear order cost functions can be transformed into an equivalent model with fixed-plus-linear order cost functions only. The transformation consists of substituting each period i by a sequence of P_i periods $(i, 1), \ldots, (i, P_i)$ with appropriately chosen cost functions and parameters.

Let $P_{\max}$ denote the maximum number of pieces required to represent the order cost functions. An optimal schedule can thus be determined, and a minimal forecast horizon can be detected for the general model $(\mathcal{P})$, via $O(n^2P_{\max}^2)$ elementary operations and as many computations of roots of (nonlinear) functions, by applying Algorithm GENFOR to the equivalent model. Under the condition stated in Proposition 1 the complexity is $O(nP_{\max})$.

Bensoussan et al. (1991) recently developed a procedure to detect a maximal planning horizon for a given forecast horizon, for the special case of the model considered here, where all holding cost functions are piecewise linear (concave), just as the order cost functions. The complexity of their procedure is

$$O(n^3H_{\max}\log_2(1 + H_{\max}) + n^2H_{\max}\log_2(1 + P_{\max}) + n^2P_{\max}\log_2(1 + H_{\max}) + nP_{\max}\log_2(1 + P_{\max}))$$

where $H_{\max}$ denotes the maximum number of pieces required to describe the holding cost functions. Our procedure can be transformed to one detecting a maximal planning horizon for a given forecast horizon, increasing the complexity by a factor $O(\log P_{\text{tot}}) = O(\log(nP_{\max}))$. Conversely, the procedure in Bensoussan et al. (1991) can be transformed to one detecting minimal forecast horizons, increasing the complexity by a factor $O(\log n)$.

3.2. Models Without Backlogging: General Concave Order Cost Functions

In this subsection we generalize the model of the previous subsection to allow for general concave order cost functions. Thus, let $c_i(x)$ = cost of placing an order of size x in period i, $(i = 1, \ldots, n)$. (We assume, without loss of generality, that the cost functions are normalized such that $c_i(0) = 0$. We also assume that $c(\cdot)$ is a nondecreasing function.)

We continue to use the same notation for the holding cost functions and demand parameters. We specify the information sets $\mathbb{I}(\cdot)$ as before, except that c_t is now interpreted as a function rather than a scalar. Note that expression (6) for the function $F(l, t)$ continues to apply, provided once again that $c_{l+1}(\cdot)$ is now interpreted as a function. Once again, the difference functions depend on t only via $D(t)$, i.e., $X(t) = D(t) \in \mathbb{I}(t)$ is the single-indicator of the future of the dynamic program (7):

$$\Delta_{k,l}(t) = \delta_{k,l}(D(t)) \quad \text{where}$$

$$\delta_{k,l}(x) = F(k) - F(l) + c_{k+1}(x - D(k)) - c_{l+1}(x - D(l)) + \sum_{r=k+1}^{l} h_r(x - D(r)). \quad (10)$$

As before, $\delta_{k,l}(\cdot)$ is $\mathbb{I}(l)$-measurable. Observe, however, that $\delta_{k,l}(\cdot)$ may now fail to be concave since the term $-c_{l+1}(x - D(l))$ is convex. This implies that the difference function may have more than two roots, i.e., $R > 2$ may occur. The following example, adapted from Bensoussan et al. (1991) shows that the difference functions may in fact have an *infinite* number of roots.

EXAMPLE 2. Let $n = 2$; $d_1 = 0$; $h_1(\cdot) = h_2(\cdot) = 0$; $c_1(x) = 1 - e^{-x}$; and $c_2(x) = 1 - e^{-x}(1 + 0.5 \sin x)$. It is easily verified that $c_1(\cdot)$ and $c_2(\cdot)$ are concave, and $c_1(0) = c_2(0) = 0$. Note that $\delta_{1,2}(x) = 0.5e^{-x} \sin x$ which has infinitely many roots.

In the remainder of this subsection we assume that Condition (C) applies. Note that if all one-period cost functions are (concave and) polynomials of uniformly bounded degree R, the same can be shown to hold for all difference functions, hence the difference functions have at most R roots. Condition (C) clearly continues to apply when all one-period cost functions are described as piecewise combinations of polynomials of uniformly bounded degree, with a uniformly bounded number of pieces. Alternatively, Condition (C) continues to apply if holding cost functions of the above type are combined with order cost functions which are rational, i.e. ratios of two polynomials, of uniformly bounded degree. Even more generally, all one-period cost functions may be chosen as linear combinations of an arbitrary so-called Tchebycheff system. A Tchebycheff system is a set of N continuous functions $u_1, \ldots, u_N$ defined on a closed interval $[a, b]$ such that every function of the form $w(x) = \sum_{i=1}^{N} a_i u_i(x)$ equals zero for at most N zeros. This condition can be verified by evaluating the sign of certain determinants, see Karlin and Studden (1966) for an extensive discussion of such systems and many examples. We conclude from theorem 2:

COROLLARY 3. *The Algorithm* GENFOR *solves the dynamic lot-sizing model with general concave one-period cost functions which satisfy Condition* (C) *for some integer* R, *and detects a minimal forecast horizon. Its complexity consists of* $O(n^2 \log^* n)$ *elementary operations and* $O(n^2)$ *computations of roots of the difference functions* δ.

Corollary 3 characterizes the complexity of the algorithm under fully arbitrary holding cost functions and demand parameters, and general order cost functions. The complexity of each iteration is proportional with the list size Ω. As discussed in §2, the worst case list size is $O(n \log^* n)$. Below we show that the list size is in fact uniformly bounded under mild conditions with respect to the cost functions and demand parameters. As before assume all demands are rational, hence integer after appropriate scaling.

PROPOSITION 3. *Assume there exists an integer* $M \geq 1$, *and uniform bounds* h_*, K^*, c_* *and* c^* *such that* $(d_i + \cdots + d_{i+M}) \geq 1$, $\underline{\lim}_{x\to\infty} \nabla^+ h_i(x) \geq h_*$, $c^* \geq \underline{\lim}_{x\to\infty} \nabla^+ c_i(x) \geq c_* > 0$, *and* $c_i(0^+) \equiv \underline{\lim}_{x\downarrow 0} c_i(x) \leq K^*$ *for all* $i = 1, \ldots, n$.

(*a*) *There exists an optimal solution in which every replenishment interval, i.e., interval between two consecutive order periods, is of length* $\leq T \equiv [K^* + (c^* - c_*)]/h_* + M$.

(*b*) *The size of the set* Ω *is uniformly bounded by* $O(T \log^* T)$ *where the constant factor is a function of* R *only*.

(*c*) *The algorithm requires* $O(n)$ *operations*.

PROOF. (a) Consider an arbitrary period t, let $j + 1$ with $t - M \leq j + 1 \leq t$ denote the last period prior to t with $d_{j+1} \geq 1$ and let $0 < i < j - [K^* + (c^* - c_*)]/h_*$. Let $F_{i,j}(t)$ denote the minimum cost in periods $1, \ldots, t$ if periods i and j are the last two periods prior to period t with zero ending inventory. Then,

$$
\begin{aligned}
F(i, t) - F(j, t) &\geq F(i, t) - F_{i,j}(t) \\
&= \sum_{r=i+1}^{j} \{h_r(D(t) - D(r)) - h_r(D(j) - D(r))\} \\
&\quad + c_{i+1}(D(t) - D(i)) - c_{i+1}(D(j) - D(i)) - c_{j+1}(D(t) - D(j)) \\
&\geq h_*(j - i)(D(t) - D(j)) + c_*(D(t) - D(j)) - c^*(D(t) - D(j)) - K^* \\
&= [D(t) - D(j)][h_*(j - i) - (c^* - c_*)] - K^* > K^* - K^* = 0,
\end{aligned}
$$

where the second inequality follows from the concavity of the $c(\cdot)$ and $h(\cdot)$ functions and the last inequality from $(j-i)h_* > K^* + (c^* - c_*)$ and $D(t) - D(j) \geq d_{j+1} \geq 1$. Thus, if $i < t - T$, it cannot be optimal to use $[i+1, t]$ as a last replenishment interval and hence it cannot be optimal to use it as a replenishment interval altogether.

(b) and (c): By part (a) there can be at most T distinct periods in any list Ω which by Szemerédi (1974) is therefore bounded by $O(T \log^* T)$ with a constant which depends on R only. □

4. Dynamic Lot-sizing Models with Backlogging and General Concave Cost Functions

In this section we consider models with backlogging, allowing for general concave holding, order and backlogging cost functions. In models with backlogging, the cost structure consists of order and holding costs, as well as backlogging costs which are functions of the period's ending backlog size. Zangwill (1969) showed that it is optimal to place a *single* order between any pair of periods with zero starting inventory, and developed an $O(n^3)$ algorithm on the basis of this property. The algorithm determines a shortest path on a network with n nodes and multiple arcs connecting every pair of nodes. Aggarwal and Park (1993) recently obtained an $O(n^2)$ solution method. As far as a detection procedure for forecast horizons is concerned, no such procedures have been obtained that allow for the variable order, holding and backlogging costs to be represented by nonlinear functions.

We show in §4.1 how the general algorithm of §2 can be used to obtain an $O(n^2)$ solution method which allows for the detection of minimal forecast horizons. The method is applicable when holding costs are given by arbitrary concave functions, order costs are piecewise linear (and concave) and backlogging costs are fixed-plus-linear. Models with piecewise linear order costs are first transformed into equivalent models with fixed-plus-linear order costs employing a transformation similar to the one described in §3.1. Fixed-plus-linear order costs are the only ones to allow for a decomposition of the planning problem into a pair of generalized shortest path problems on networks with n nodes and a *single* arc connecting every pair of nodes. This observation is due to Zangwill (1969) and is also the basis of the solution method for the basic model with backlogging (i.e., when order costs are fixed-plus-linear, holding and backlogging costs are linear) in Federgruen and Tzur (1993).

The restriction to fixed-plus-linear backlogging costs is necessary to obtain difference functions for the second of the two dynamic programs, which depend on the horizon length via a single indicator only (in particular, the characteristic $\hat{C}(t)$, see below).

To conclude our discussion on dynamic lot-sizing models with general concave costs, we refer to Federgruen and Tzur (1992) for a simple forward algorithm for the general model, allowing for arbitrary concave one-period cost functions throughout, which requires $O(n^2)$ operations and computations of roots of single variable functions. However, this method doesn't allow for the detection of minimal forecast horizons.

The cost structure in models with backlogging is described by a triple of one-period cost functions:

$c_i(x)$ = cost of placing an order of size x in period i $(i = 1, \ldots, n)$;

$h_i(x)$ = cost of carrying x units of inventory at the end of period i $(i = 1, \ldots, n)$;

$h_i^-(x)$ = cost of a backlog of size x at the end of period i (similar to $h_i(\cdot)$, we assume that the $h_i^-(\cdot)$ functions are nondecreasing).

We assume again, without loss of generality, that $d_i \geq 0$.

4.1. An $O(n^2)$ Solution Method Allowing for the Detection of Minimal Forecast Horizons

Assume now that the functions $c_i(\cdot)$ are piecewise linear, and $h_i(\cdot)$ fixed-plus-linear. We first transform the model into an equivalent model with fixed-plus-linear order costs only. Again, the transformation consists of substituting each period i by a sequence of P_i periods $(i, 1), \ldots, (i, P_i)$ with the same holding cost functions, order cost and demand parameters as in §3.1. and with appropriately chosen backlogging cost functions, see Federgruen and Tzur (1992).

The equivalency between the original and transformed model is again easily established. In the remainder of this subsection we assume therefore, without loss of generality, that the order functions $c_i(\cdot)$ are fixed-plus-linear, i.e., $c_i(x) = K_i + c_i x$ for $x > 0$ and $c_i(0) = 0$.

Let $F_1(t)$ = minimum cost for satisfying demands in periods $1, \ldots, t$ from production in these periods, and $F_2(t)$ = minimum cost for satisfying demands in periods $1, \ldots, t$ from production in these periods, given that a final setup occurs in period t and starting inventory in period t is nonpositive.

The functions $F_1(\cdot)$ and $F_2(\cdot)$ satisfy the following pair of recursions:

$$F_1(t) = \min_{k:1\le k\le t} \left\{ F_2(k) + c_k(D(t) - D(k)) + \sum_{r=k}^{t-1} h_r(D(t) - D(r)) \right\}, \quad t = 1, \ldots, n, \quad (11)$$

$$F_1(0) = 0,$$

$$F_2(t) = \min_{k:0\le k<t} \left\{ F_1(k) + \sum_{r=k+1}^{t-1} [L_r + \hat{h}_r(D(r) - D(k))] + K_t + c_t[D(t) - D(k)] \right\}, \quad t = 1, \ldots, n. \quad (12)$$

Equations (11) and (12) are both generalized shortest path problems of type (1) with $E(k) = F_2(k)$ in (11) and $E(k) = F_1(k)$ in (12). (One easily verifies that both $F_1(k)$ and $F_2(k)$ are $\mathbb{I}(k)$-measurable.)

One easily verifies that the difference functions $\Delta^1_{k,l}(t)$ associated with (11) are of the form $\Delta^1_{k,l}(t) = \delta_{k,l}(X(t))$ with $X(t) = D(t)$ and with all $\delta_{k,l}(\cdot)$ function $\mathbb{I}(l)$-measurable ($k < l \le t$). Similarly, the difference functions $\Delta^2_{k,l}(t)$ associated with (12) are of the form

$$\begin{aligned}\Delta^2_{k,l}(t) &= F_1(k) - F_1(l) + \sum_{r=k+1}^{l} [L_r + \hat{h}_r D(r)] \\ &\quad - \left(\sum_{r=k+1}^{t-1} \hat{h}_r\right) D(k) + \left(\sum_{r=l+1}^{t-1} \hat{h}_r\right) D(l) \\ &\quad - c_t(D(k) - D(l)) \\ &= F_1(k) - F_1(l) + \sum_{r=k+1}^{l} [L_r + \hat{h}_r D(r)] \\ &\quad + \sum_{r=1}^{k} \hat{h}_r D(k) - \sum_{r=1}^{l} \hat{h}_r D(l) \\ &\quad + \hat{C}(t)(D(l) - D(k))\end{aligned}$$

where $\hat{C}(t) = c_t + \sum_{r=1}^{t-1} \hat{h}_r$. ($\hat{C}(t)$ represents the per unit cost of satisfying demand in the first period via production in the tth period.) In other words, $\Delta^2_{k,l}(t) = \delta_{k,l}(\hat{C}(t))$ with $\hat{C}(t) \in \mathbb{I}(t)$ and $\delta_{k,l}(\cdot)$ is $\mathbb{I}(l)$-measurable.

The difference functions for (11) are concave in $X(t) = D(t)$, i.e., for this recursion Condition (C) is satisfied with $R = 2$. The difference functions for (12) are *linear* and nondecreasing in $X(t) = \hat{C}(t)$. We conclude that both dynamic programs (11) and (12) can be solved, respectively, by the algorithm GENERAL, as long as the cost values are computed in the sequence

$$F_2(1), F_1(1), F_2(2), F_1(2), \ldots, F_2(n), F_1(n),$$

i.e., the jth iteration of the algorithm for (12) is followed by the jth iteration of the algorithm for (11), which is followed by the $(j + 1)$st iteration for (12), etcetera. Let $\{\Omega(j), j = 1, \ldots, n\}$ and $\{W(j), j = 1, \ldots, n\}$ denote the lists generated by GENERAL for (11) and (12) respectively. These lists have the following interpretations: $\Omega(j) = \{1 \le l \le j$: l is the optimal last order period for some horizon $t \ge j$ with a potential cumulative demand $D \ge D(j)\}$. $W(j) = \{1 \le i_l \le j$: i_l is an optimal last period with zero starting inventory (given that a final setup occurs in period j and the starting inventory in period j is nonpositive) for some horizon $t \ge j$ with an appropriate backlogged procurement cost rate $\hat{C}\}$.

A minimal forecast horizon can be detected for both dynamic programs *separately*, by appending the test (4) to each iteration. This, however, may fail to generate a *meaningful* forecast horizon L and an associated planning horizon l in the sense that in every scenario $t > L$, it is optimal to have period l be the first period after period 0 with zero ending inventory. It is this quantity which determines the lot-size for the first l periods, the immediate decisions to be implemented.

A first complication is due to period 0 arising as an optimal last period with zero ending inventory for any future period t, if the variable order cost in that period, or some of the backlogging cost rates in periods $n + 1, \ldots, t$ are sufficiently negative. In other words, without any restriction on the parameter values, no planning horizons can ever occur. We therefore assume that a constant $\hat{C}_*$ exists such that $\hat{C}(t) \ge \hat{C}_*$ for all t. (As discussed in §2, any such restriction on future index-values is easily incorporated in the algorithm.)

We now discuss how a meaningful minimal forecast horizon *can* be detected. Let $l(t)$ and $z(t)$ denote the period k which achieves the minimum in (11) and (12) respectively. That is, $l(t)$ is the last order period in the optimal schedule for periods $1, \ldots, t$, and $z(t)$ is the last period with zero ending inventory in the optimal schedule for $1, \ldots, t$ which places an order in period t. (As in §2.3 we assume without loss of generality that both (11) and (12) have a unique optimal solution.) Also, for $t = 1, \ldots, n$ consider the optimal schedule for periods $1, \ldots, t$ (assuming all demands over this horizon need to be satisfied from production in these periods). Let

$$s(t) = \begin{cases} 0 & \text{if under this schedule, no period} \\ & \quad j = 1, \ldots, t-1 \text{ has zero-ending} \\ & \quad\quad \text{inventory,} \\ \text{the first period (after 0)} & \\ \quad \text{with zero-ending inventory otherwise.} & \end{cases}$$

The values $s(t)$ can be computed via the recursion

$$s(t) = s(z(l(t))) \quad (t = 1, \ldots, n).$$

Let $\Omega(j) = \{i_1, \ldots, i_p\}$ and $W(j) = \{m_1, \ldots, m_q\}$. A period j is a forecast horizon with associated planning horizon s^* if and only if the following two conditions are satisfied:

$$s(z(i_1)) = s(z(i_2)) = \cdots = s(z(i_p)) = s^* \geq 1, \quad (13)$$

$$s(m_1) = s(m_2) = \cdots = s(m_q) = s^* \geq 1. \quad (14)$$

(The proof of these necessary and sufficient conditions is analogous to that given in Federgruen and Tzur (1993).) We refer to the pair of algorithms GENERAL for (11) and (12), combined with the pair of tests (13) and (14) at the end of their jth iteration, as the algorithm GENBACK. We conclude:

THEOREM 4. (*a*) *Algorithm* GENBACK *solves and finds minimal forecast horizons in the dynamic lot sizing model with arbitrary concave holding cost functions, piecewise linear order cost functions, and fixed-plus-linear backlogging cost functions in* $(P_{\text{tot}}^2) = O(n^2 P_{\max}^2)$ *time.*

(*b*) *Algorithm* GENBACK *has complexity* $O(P_{\text{tot}}) = O(nP_{\max})$ *if an integer* $M \geq 1$, *and constants* h_*, $\hat{h}_*$, K^*, c_* *and* c^* *exist such that* $(d_i + \cdots + d_{i+M}) \geq 1$, $\underline{\lim}_{x\to\infty} \nabla^+ h_i(x) \geq h_*$, $\hat{h}_i \geq \hat{h}_*$, $K_i \leq K^*$, *and* $c_* \leq c_i \leq c^*$, *for all* $i = 1, \ldots, n$.

5. Additional Examples

In Federgruen and Tzur (1992) we describe a number of additional planning models which can be formulated as generalized shortest path problems of type (1) with difference functions of type (2), thus allowing for algorithm GENFOR to be used as a solution method and a means toward detecting minimal forecast horizons. These include a scheduling problem discussed below, as well as machine replacement, bond refunding problems and modified edit distance problems; the latter arise in many applications, e.g. in the context of sequence comparison in molecular biology (to identify similarities in the acid sequences of proteins or to identify RNA secondary structure; see e.g., Waterman (1984) and Waterman and Smith (1986) in geology (e.g., Smith and Waterman 1980) and in speech recognition (Kruskal and Sankoff 1983). Here we confine ourselves to the scheduling and modified edit distance problems.

5.1. A Scheduling Problem

A given number of items (say n) is to be processed on a single machine during a given period of time (a shift, day, week, etc.). The items are to be processed in a *given* sequence (perhaps a First Come First Serve sequence). The processing of the items starts after a machine setup; completed items, however, are not made available until the next setup. In other words, to make the items available to downstream activities (such as shipping), a setup delay of S time units is incurred. A batch includes the items completed between consecutive setups. The flow time of an item (= time spent in the system) coincides with the completion time of the batch in which the item is processed; thus, all items in a batch have the same flow time.

Assuming that n, the total number of jobs to be processed, is known in advance, the problem is to find the collection of batches that partition the set of items in a *dynamic* environment, such that the sum of the jobs' flow times is minimized. In a dynamic environment, we may at any stage of the process have knowledge of the processing times of only few (say m) subsequent jobs. Coffman et al. (1990) studied the problem in a *static* environment (i.e., one where all items' processing times are known in advance), and where the sequence to process the items is to be determined as well.

It is clearly optimal to complete (start) a batch when the last (first) job in the batch is completed (started), i.e., an optimal schedule has no idle times. The problem can thus be represented as a shortest path problem on an acyclic network with a set of nodes $\mathbb{N} = \{0, \ldots, n\}$ and arcs (i, j) for all $0 \le i < j \le n$. A path which uses arc (i, j) corresponds with a schedule which employs the batch $\{i + 1, \ldots, j\}$.

Define $c(i, j)$ as the contribution of jobs $i + 1, \ldots, j$ to the flow times of all jobs. Let p_i = the processing time of job i and $X(i) = \sum_{l=1}^{i} p_l$ = cumulative processing time of the first i jobs.

Assume $\mathbb{I}(0) = \{n\}$ and $\mathbb{I}(l) = \mathbb{I}(l-1) \cup \{p_l\}, l = 1, \ldots, n$.

Note that a batch $\{i + 1, \ldots, j\}$ contributes $(S + X(j) - X(i))$ to jobs $i + 1, \ldots, n$ and nothing to jobs $1, \ldots, i$. Thus, $c(i, j) = (n - i)(S + X(j) - X(i))$. For any $k < l$ the difference function $\Delta_{k,l}(t)$ thus takes the form:

$$\Delta_{k,l}(t) = F(k) - F(l) + k \cdot X(k) - l \cdot X(l) + n \cdot (X(l) - X(k)) + (l - k)S + (l - k)X(t). \tag{15}$$

In other words, $\Delta_{k,l}(t) = \delta_{k,l}(X(t))$ where $X(t)$ is $\mathbb{I}(t)$-measurable, and where $\delta_{k,l}(\cdot)$ is $\mathbb{I}(l)$-measurable, and linear and nondecreasing in the single indicator $X(t)$. It follows from Proposition 1(b) that algorithm GEN-FOR (modified by the procedure SRUP) can be used to solve the problem and to detect minimal forecast horizons in $O(n)$ time only. (Note that the root of the linear function $\Delta_{k,l}(\cdot)$ can be computed in constant time; see (15).) The ability to detect minimal forecast horizons allows for the determination of optimal initial batches, even when only some processing times of future jobs are known.

5.2. The Modified Edit Distance Problem

The modified edit distance problem is defined as follows: given two strings of letters in an alphabet Σ, $x = (x_1, \ldots, x_m)$ and $y = (y_1, \ldots, y_n)$, the *modified edit distance* of x and y is the minimal cost of an edit sequence that changes x into y. The edit sequence consists of operations of the form **delete, insert,** and **substitute.** A *deletion* consists of deleting a substring from the string x; an *insertion* consists of adding a substring of y to an adjacent substring; a *substitution* consists of substituting x_i by y_j at a cost $s(x_i, y_j)$. In the above mentioned applications the deletion cost of a string $x_{l+1}, \ldots, x_k$ is of the form:

$$w^{\text{del}}(l, k) = f^1(x_l, x_{l+1}) + f^2(x_k, x_{k+1}) + g(k - l). \tag{16}$$

This cost consists of charges for breaking the string before x_{l+1} and after x_k plus an additional cost which is a function of the length of the gap. The cost of inserting a substring $(y_{l+1}, \ldots, y_i)$ to an existing string is given by $w^{\text{ins}}(l, i)$ which is of the same structure as (16).

To compute the modified edit distance, the following dynamic programming equations are used, see, e.g., Galil and Giancarlo (1989) or Park (1992). Let $D(i, j)$ = the minimum cost of changing the string $x_1, \ldots, x_j$ into the string $y_1, \ldots, y_i$.

$$D(i, j) = \min\{D(i-1, j-1) + s(x_j, y_i), E(i, j), F(i, j)\} \quad \text{where} \tag{17}$$

$$E(i, j) = \min_{0 \le k \le j-1} \{D(i, k) + w^{\text{del}}(k, j)\}, \tag{18}$$

$$F(i, j) = \min_{0 \le l \le i-1} \{D(l, j) + w^{\text{ins}}(l, i)\}, \tag{19}$$

with initial conditions:

$$D(i, 0) = w^{\text{ins}}(0, i), \quad 1 \le i \le m, \quad \text{and}$$
$$D(0, j) = w^{\text{del}}(0, j), \quad 1 \le j \le n.$$

The first alternative in (17) consists of transferring $(x_1, \ldots, x_{j-1})$ into $(y_1, \ldots, y_{i-1})$ and then substituting x_j by y_i; the second alternative consists of deleting string $(x_{k+1}, \ldots, x_j)$ from the end of x and transforming $(x_1, \ldots, x_k)$ to $(y_1, \ldots, y_i)$ (see (18)); the third alternative consists of transforming $(x_1, \ldots, x_j)$ into $(y_1, \ldots, y_l)$ and inserting the string $(y_{l+1}, \ldots, y_i)$ at the end, see (19). Notice that the computation of $D(i, j)$ reduces to the computation of $E(i, j)$ and $F(i, j)$. The computation of a row of E and of a column of F are each equivalent to the following problem:

$$E(j) = \min_{0 \le k \le j-1} \{D(k) + w(k, j)\}, \; j = 1, \ldots, n \tag{20}$$

where $w(\cdot, \cdot) = w^{\text{del}}(\cdot, \cdot)$ or $w^{\text{ins}}(\cdot, \cdot)$, and $D(k)$ is easily computable from $E(k)$. This is a generalized shortest path problem of type (1). Its difference functions are given by:

$$\Delta_{k,l}(t) = D(k) - D(l) + w(k, t) - w(l, t). \quad (21)$$

Since $w(\cdot, \cdot)$ is of the form (16), the difference function $\Delta_{k,l}(t)$ is of the form $\delta_{k,l}(X(t))$ with $X(t) = t$, i.e. the difference function depends on a single indicator $X(t) = t$. Algorithm GENFOR may thus be used; if the function $g(\cdot)$ in (16) is convex (concave), it follows that the difference functions are monotonically nondecreasing (nonincreasing); the complexity of the algorithm thus reduces to $O(n)$ computations of roots of difference functions and $O(n)$ additional operations. The actual time to compute a root depends on the function $g(\cdot)$; since the roots needs to be determined up to the nearest integer only, the computation can always be performed by a binary search on the integers $\{0, \ldots, n\}$ in log n time, resulting in an overall complexity of $O(n \log n)$.

Eppstein (1990) considers the generalization where breakpoints $c_1, \ldots, c_s$ are given such that the function $g(\cdot)$ is concave on $(0, c_1)$, convex on (c_1, c_2), etc. Under this structure, condition (C) is satisfied with $R = \lfloor c_s \rfloor + 1$ since there is at most one root t with $t - l > c_s$ and since there are at most $\lfloor c_s \rfloor$ integer points in $[l, c_s + l]$ where the function changes signs.

Determination of minimal forecast horizons is essential in some of the above applications (e.g., speech recognition) where the elements of the string become known progressively; the detection of minimal forecast horizons allows us to implement optimal initial edits prior to observing the entire string (message, speech).[1]

[1] We would like to thank Dan Bienstock for introducing us to the literature on Davenport-Schinzel sequences as well as the referees of this paper for many insightful comments.

Appendix. An $O(n \log n)$ Algorithm for the Case of a Single Root

To achieve the complexity of $O(n \log n)$ for our method when $R = 1$, it is necessary to do all updates within a *single* pair of arrays Ω and G, as opposed to the two pairs $\{\Omega^{\text{new}}, G^{\text{new}}\}$ and $\{\Omega^{\text{old}}, G^{\text{old}}\}$, employed by GENERAL. (Recall that the mere transfer of the updated values in Ω^{new} and G^{new} to Ω^{old} and G^{old} in Step 2, involves $O(n^2)$ operations by itself.) We number the elements of $\Omega(\cdot)$ and $G(\cdot)$ as FIRST, FIRST + 1, ..., LAST for appropriate values of FIRST and LAST.

The update of the lists in the jth iteration starts with the determination of an index p (if any) with

$$[g(p), g(p + 1)] \cap [\underline{g}, \bar{g}] \neq \emptyset.$$

This is achieved by a procedure POSITION (see Step 1 in the algorithm below) which applies a binary search on the range of indices FIRST, FIRST + 1, ..., LAST. This procedure terminates in $O(\log |\Omega(j-1)|) = O(\log j)$ iterations either with an appropriate index p or the conclusion that $\Omega(j) = \Omega(j-1)$. In the former case, the remaining update consists of eliminating from the list nodes currently in positions $\underline{p} + 1, \ldots, p$ and those in positions $p + 1, \ldots, \bar{p} - 1$ for appropriate values of $\underline{p}$ and $\bar{p}$. Recall that if a node in the lth position is to be eliminated with $l < p$ or $l > p$, then so do all intermediate nodes; otherwise j would dominate in (part of) $[g(p), g(p + 1)]$ as well as $[g(l), g(l + 1)]$ but *not* in between, implying that node j would need to appear more than once in the list. The elimination of nodes in positions $l \leq p$ $(l \geq p)$ is terminated as soon as one is found for which $G(i_l, j)$, the root of the difference equation $\delta_{i_l,j}$ is $> g(l)$ $(<g(l + 1))$. (If this difference equation fails to have a root, node j dominates i_l, i.e. $\delta_{i_l,j}(x) > 0$ throughout and i_l can thus be eliminated; the alternative where $\delta_{i_l,j}(x) < 0$ throughout violates the fact that node j dominates in part or all of the interval $[g(p), g(p + 1)]$.)

To delete nodes and insert node j into the list we employ three procedures:

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

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

INSERT (j, p): this procedure reduces the record indices $\{$FIRST, ..., $p\}$ 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 $O(n)$ 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 elementary arithmetic operations. The problem may be overcome by maintaining the list Ω as a *balanced binary tree* see e.g. Tarjan (1983). The work required to maintain and access such a tree is $O(n \log n)$ only.

In line with the earlier convention, we set $G(i, j) = -\infty$ $(+\infty)$ if $\delta_{i,j}(\cdot)$ has no root and is positive (negative).

The algorithm for updating the list in a given iteration j can thus be described as follows:

Single Root Update Procedure (SRUP)

```
a:= FIRST; b:=LAST;
Step 1: Procedure POSITION(j)
If (b ≥ a) then k:= ⌊(a+b)/2⌋; otherwise stop.
Compute $G(i_k, j)$, the single root of $\delta_{i_k,j}$ (if any).
If (no root exists and $\delta_{i_k,j}(g(k)) < 0$) then stop.
  ($j$ is always dominated by $i_k$ and $\Omega(j) = \Omega(j-1)$.)
If (no root exists and $\delta_{i_k,j}(g(k)) > 0$) then p:= k; ($G(i_p, j) = -\infty$); go to Step 2.
  ($j$ dominates over $0, \ldots, j-1$ in $[g(k), g(k+1)]$.)
```

```
If ($G(i_k, j) \leq g(k)$ and $\delta_{i_k,j}(g(k+1)) < 0$) then $b := k-1$; repeat Step 1.
    (Node $j$ does not dominate for $x \in [G(i_k, j), \infty)$; the search for $p$
    can be confined to the indices $a, a+1, \ldots, k-1$.)
If ($G(i_k, j) \geq g(k+1)$ and $\delta_{i_k,j}(g(k)) < 0$) then $a := k+1$; repeat Step 1.
    (Node $j$ does not dominate for $x \in (-\infty, G(i_k, j)]$; the search for
    $p$ can be confined to the indices $k+1, \ldots, b$.)
$p := k$;
    (In all other cases either (i) $G(i_k, j) < g(k+1)$ and $\delta_{i_k,j}(g(k+1))$
    $> 0$, or (ii) $G(i_k, j) > g(k)$ and $\delta_{i_k,j}(g(k)) > 0$, so that $j$ dominates
    in all or part of $[g(k), g(k+1)]$.)
If ($\delta_{i_k,j}(g(k+1)) < 0$) then begin $\bar{p} := k$; $g(k) := G(i_k, j)$ end
If ($\delta_{i_k,j}(g(k)) < 0$) then begin INSERT($j, p$); $g(p) := G(i_k, j)$; go to Step 3 end
$p := p-1$;

Step 2:
Unless already determined, compute $G(i_p, j)$, the single root of $\delta_{i_p,j}$;
If ($G(i_p, j) \leq g(p)$) then begin DELUP($p$); if ($p >$ FIRST) then repeat Step 2;
    otherwise INSERT($j, p$); $g(p) := -\infty$; end
otherwise begin $x := G(i_p, j)$; INSERT($j, p$); $g(p) := x$; end
if ($\bar{p} = k$) then stop.

Step 3:
Compute $G(i_{p+1}, j)$, the single root of $\delta_{i_{p+1},j}$.
If ($G(i_{p+1}, j) > g(p+2)$) then begin DELDOWN($p$); if ($p <$ LAST) then repeat Step 3;
    otherwise, stop. end
$g(p+1) := G(i_{p+1}, j)$.
```

PROOF OF PROPOSITION 1. (a) Step 1 of SRUP (the procedure POSITION) is repeated at most $\log|\Omega(j-1)| \leq \log j$ times. In each execution of this procedure, a single root is determined and a bounded number of additional operations is required. Note that a given node is deleted at most once in the course of the algorithm. Thus the total number of executions of Steps 2 and 3 is at most n, and the amount of work required in each execution is $O(\log n)$ (given that the list is maintained as a balanced binary tree).

(b) Assume all difference functions are nondecreasing. (The proof for the case where they are all non-increasing is identical.) Assume node j enters the list (in the jth iteration). Let $\Omega(j-1) = \{i_1, \ldots, i_m\}$. We prove that j enters the list at its bottom. This implies that the POSITION procedure (Step 1 in SRUP) can be eliminated and that a simple list suffices to do all deletions and insertions throughout the algorithm with $O(n)$ determinations of roots of difference functions and $O(n)$ additional operations.

Let $G(i_m, j)$ be the single root of $\delta_{i_m,j}(\cdot)$. (If $\delta_{i_m,j}(x) \geq 0$ for all x, set $G(i_m, j) = -\infty$; if $\delta_{i_m,j}(x) \leq 0$ for all x, $j \notin \Omega(j)$.) Let $x^0 > \max\{G(i_m, j), g(m)\}$. By the definitions of $\Omega(j-1)$ and x^0, i_m dominates every $i_k \in \Omega(j-1)\backslash\{i_m\}$ on $[x^0, \infty)$. Also, since the difference functions are nondecreasing, if j enters the list it dominates i_m on $[x^0, \infty)$. Therefore j dominates every element of $\Omega(j-1)$ on $[x^0, \infty)$, implying that it enters the list at its bottom. □

References

Aggarwal, A., M. M. Klaw, 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, *Oper. Res.*, 41 (1993), 549–571.

→ Bean, J. and R. Smith, Condition for the Existence of Planning Horizons, *Math. Oper. Res.*, 9 (1984), 391–401.

—— and ——, Conditions for the Discovery of Solution Horizons, *Math. Programming*, 59 (1993), 215–229.

Bensoussan, A., J. M. Proth, and M. Queyranne, A Planning Horizon Algorithm for Deterministic Inventory Management with Piecewise Linear Concave Costs, *Naval Res. Logistics*, 38 (1991), 729–742.

Coffman, E. G., M. Yannakakis, M. J. Magazine, and C. Santos, Batch Sizing and Job Sequencing on a Single Machine, *Annals of Oper. Res.*, 26 (1990), 135–147.

Conte, S. and C. de Boor, *Elementary Numerical Analysis: An Algorithmic Approach*, McGraw Hill, NY, 1972.

Davenport, H. and A. Schinzel, A Combinatorial Problem Connected with Differential Equations, *Amer. J. Math.*, 86 (1965), 684–694.

Denardo, E. V., *Dynamic Programming Models and Applications*, Prentice-Hall, NJ, 1982.

Eppstein, D., Sequence Comparison with Mixed Convex and Concave Costs, *J. Algorithms*, 11 (1990), 85–101.

→ Federgruen, A. and M. Tzur, Minimal Forecast Horizons and a New Planning Procedure for the General Dynamic Lot Sizing Model: Nervousness Revisited, *Oper. Res.*, 42 (1994), 456–469.

-→ and ——, A Simple Forward Algorithm to Solve General Dynamic Lot-sizing Models with n Periods in $O(n \log n)$ or $O(n)$ Time, *Management Sci.*, 37 (1991), 909–925.

—— and ——, Fast Solution and Detection of Minimal Forecast Horizons in Dynamic Programs with a Single Indicator of the Future: Applications to Dynamic Lot-sizing Models, unabridged version, University of Pennsylvania, 1992.

—— and ——, The Dynamic Lot-sizing Model with Backlogging: A Simple $O(n \log n)$ Algorithm and Minimal Forecast Horizon Procedure, *Naval Res. Logistics*, 40 (1993), 459–478.

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

Hart, S. and M. Sharir, Nonlinearity of Davenport-Schinzel Sequences and Generalized Path Compression Schemes, *Combinatorica*, 6 (1986), 151–177.

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

IMSL. *Mathlibrary*, vol. 2, IMSL, Inc., Houston, TX, 1991.

Karlin, S. and J. W. Studden, *Tchebycheff Systems: With Application in Analysis and Statistics*, Wiley & Sons, 1966.

Karp, R. M., On-line Algorithms Versus Off-line Algorithms: How Much Is It Worth to Know the Future?, University of California at Berkeley, 1992.

Kruskal, J. B. and D. Sankoff, Eds., *Time Warps, String Edits, and*

Macromolecules: The Theory and Practice of Sequence Comparison, Addison-Wesley, NY, 1983.

Leavenworth, D., Algorithm 25: Real Zeros of an Arbitrary Function, *Communications of the ACM*, 3 (1960), 602.

McKenzie, L., Turnpike Theory, *Econometrica*, 44 (1976), 841–864.

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

Muller, D., A Method for Solving Algebraic Equations Using an Automatic Computer, *Mathematical Tables and Aids to Computation*, 10 (1966), 208–215.

Park, K., Fast Sequential and Parallel Dynamic Programming, Ph.D. Dissertation, Columbia University, New York, NY, 1992.

Rockafellar, R., *Convex Analysis*, Princeton University, Princeton, NJ, 1970.

Sharir, M., Almost Linear Upper Bounds on the Length of General Davenport-Schinzel Sequences, *Combinatorica*, 7 (1987), 131–193.

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

Szemerédi, E., On a Problem of Davenport and Schinzel, *Acta Arithmetica*, 25 (1974), 213–224.

Tarjan, R., *Data Structures and Network Algorithms*, CBMS-NSF Regional Conference Series in Applied Mathematics, Volume 44, 1983.

Tzur, M., A New Planning Approach for Single and Multi-item Dynamic Lot-sizing Models, Ph.D. Dissertation, Columbia University, New York, NY, 1992.

Wagelmans, A., S. Van Hoesel and A. Kolen, Economic Lot-sizing: An $O(n \log n)$-Algorithm That Runs in Linear Time in the Wagner-Whitin Case, *Oper. Res.*, 40 (1992), S145–S156.

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, *Bull. Math. Biology*, 46 (1984), 473–501.

—— and T. F. Smith, Rapid Dynamic Programming Algorithms for RNA Secondary Structure, *Advances in Applied Math.*, 7 (1986), 455–464.

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

Zangwill, W. I., Minimum Concave Cost Flows in Certain Networks. *Management Sci.*, 14 (1968), 429–450.

——, A Backlogging Model and a Multi-echelon Model of a Dynamic Economic Lot Size Production System—A Network Approach. *Management Sci.*, 15 (1969), 506–527.

Accepted by Thomas M. Liebling; received July 1993. This paper has been with the authors 7 months for 2 revisions.