FINDING OPTIMAL (s, S) POLICIES IS ABOUT AS SIMPLE AS EVALUATING A SINGLE POLICY

YU-SHENG ZHENG

University of Pennsylvania, Philadelphia, Pennsylvania

A. FEDERGRUEN

Columbia University, New York, New York

(Received March 1989; revisions received September, December 1989; accepted January 1990)

In this paper, a new algorithm for computing optimal (s, S) policies is derived based upon a number of new properties of the infinite horizon cost function $c(s, S)$ as well as a new upper bound for optimal order-up-to levels S^* and a new lower bound for optimal reorder levels s^*. The algorithm is simple and easy to understand. Its computational complexity is only 2.4 times that required to evaluate a (specific) single (s, S) policy. The algorithm applies to both periodic review and continuous review inventory systems.

In many models of single item inventory systems, it is well known that an optimal policy exists within the class of so-called (s, S) policies. Under an (s, S) policy, an order is placed to increase the item's inventory position (= inventory on-hand + orders outstanding − backlogs) to the level S as soon as this inventory position reaches or drops below the level s ($-\infty < s < S < \infty$; expressing demands as multiples of a standard unit, we assume without loss of generality, that both s and S are integers). In this paper, we present a new algorithm for computing an optimal policy that is simple to state, the correctness of which is simple to prove and which is more efficient than the best available alternatives. In fact, the computational complexity of the algorithm is only 2.4 times that required to evaluate a (specific) single (s, S) policy.

Until recently, algorithms for computing the best policy were rarely used because they were considered prohibitively expensive. Federgruen and Zipkin (1984) attempted to eradicate this myth by presenting an algorithm that could be implemented to require an average of no more than 0.4 CPU seconds on an IBM 4341. Their method, which is similar to that of Johnson (1968) and Kuenle and Kuenle (1977), is based on an adaptation of the general policy-iteration method for solving Markov decision problems, where the special structure of (s, S) policies is exploited in several ways.

The need for these algorithms arose from the general perception that the policy cost function is, in general, ill behaved, implying that almost all possible combinations $\{(s, S): s < S\}$ need to be evaluated in a direct search procedure. Indeed, the function in general fails to be quasiconvex and may have several local optima.

The methods of Veinott and Wagner (1965), Bell (1970), and Archibald and Silver (1978) therefore apply essentially full enumeration of the two-dimensional grid on the (s, Δ) plane ($\Delta \equiv S - s$). Such enumeration is facilitated by lower bounds $\underline{s}$, $\underline{S}$ and upper bounds $\bar{s}$, $\bar{S}$ for optimal reorder levels and order-up-to levels s^* and S^*, as established by Veinott and Wagner. (The effort of identifying bounds for s^* and S^* was started by Iglehart 1963.) Archibald and Silver's algorithm improves the efficiency by exploiting slightly better bounds for Δ, as well as the fact that the cost function $\bar{c}(s, \Delta)$ is convex in s, under some cost structures. Stidham (1977) and Sahin (1982) demonstrate additional properties of the cost function $\bar{c}(s, \Delta)$ under certain assumptions with respect to the demand distributions and/or lead times, in particular, the unimodality of $-\bar{C}(\Delta)$, where $\bar{C}(\Delta) = \min_s \bar{c}(s, \Delta)$. Stidham (1986) proves these properties under more general assumptions. See Sahin (1983, 1988) for a discussion of related properties.

In contrast with these methods that search in the (s, Δ) plane, our algorithm provides for an efficient search in the (s, S) plane itself and is based on a number of (to our knowledge hitherto undiscovered) properties of the cost function $c(s, S)$, as well as new tight lower and upper bounds for s^* and S^*, respectively, which are *iteratively* and *easily* updated, and converge *monotonically*. (These bounds are of independent interest.) Let $\underline{s}_0$ and $\bar{S}_0$ be the initial values of these bounds, i.e., $\underline{s} \leqq \underline{s}_0 \leqq s^*$ and $S^* \leqq \bar{S}_0 \leqq \bar{S}$ for any optimal (s^*, S^*)

Subject classifications: Inventory/production, policies: efficient algorithm for optimal (s, S) policy. Inventory/production, stochastic: efficient algorithm for optimal (s, S) policy.

0030-364X/91/3804-0654 $01.25

policy. Our algorithm requires fewer than $(\bar{S}_0 - \underline{S}) + (s^* - \underline{s}_0)$ evaluations of the $c(\cdot\,\cdot)$-function instead of $\Omega((\bar{S} - \underline{S})(\bar{s} - \underline{s}))$ in existing search methods. (An algorithm has complexity $C = \Omega(N)$ if C grows at least linearly with N, i.e., $\liminf_{N\to\infty} C(N)/N > 0$.) Moreover, we exploit a characterization of the cost function to allow for fast updates of $c(s, S)$ when only the value of s is altered.

Numerous *heuristic* methods have been proposed, in addition to the *exact* methods discussed; see e.g., Roberts (1962), Porteus (1979), Wagner et al. (1965), Wagner (1975), Ehrhardt (1979, 1984), Freeland and Porteus (1980 a, b), Naddor (1975), Sivazlian (1971), Schneider (1978), Tijms and Groenevelt (1984), and Sahin and Sinha (1987). This paper generalizers an even simpler algorithm that we obtained (Federgruen and Zheng 1988) for the special case where orders are placed when the recorder point is reached *exactly*. (This situation arises in continuous review systems in which demands occur on a unit-by-unit basis.) Such policies are usually referred to as (r, Q) policies, where r denotes the recorder level and Q is the (in this case) *fixed order size*.

The following is an outline of the remainder of this paper. In Section 1 we define the notation and give a description of the model. We initially confine ourselves to periodic review models and the long-run average cost criterion. Section 2 derives the properties of the cost function and the new bounds for s^* and S^*. Section 3 contains the algorithm and a detailed discussion of its complexity. In Section 4 we exhibit how the algorithm performs on a sample of test problems. Extensions to continuous review systems and the discounted cost criterion are discussed in Section 5. A comparison with alternative procedures, in particular the Federgruen and Zipkin, and the Archibald and Silver algorithms, may be found in the Appendix.

1. NOTATION AND PRELIMINARIES

We consider first a discrete time inventory system in which an order may be placed with an outside supplier at the beginning of each period. All stockouts are backordered. We assume that one period demands are i.i.d. and integer valued and that the cost structure and parameters are stationary as well. Our (initial) objective is to minimize long-run average costs over an infinite horizon. In Section 5, we discuss how our algorithm applies to models with the discounted cost criterion and/or continuous review systems. Let

D = the one-period demand (random variable);
$p_j = \Pr\{D = j\}, \quad j = 0, 1, 2, \ldots;$
K = the fixed cost to place an order;
$G(y)$ = the one-period expected costs, including e.g., holding and backlog penalty costs, when starting with an inventory position y; y integer.

(The long-run average order quantity equals ED under any policy that avoids infinitely large inventories or backlogs. Linear order costs may thus be ignored for the purpose of determining an optimal policy.) We assume only that $-G(\cdot)$ is unimodal and

$$\lim_{|y|\to\infty} G(y) > \min_y G(y) + K.$$

Without loss of generality, we assume that $K > 0$, as for $K = 0$ the policy $(y^* - 1, y^*)$ is optimal for any minimum y^* of the $G(\cdot)$ function. The assumed properties include the common case where the one-period holding (backlogging) costs increase linearly or convexly with the end-of-period inventory (backlog) size, and where $G(\cdot)$ is convex and

$$\lim_{|y|\to\infty} G(y) = +\infty,$$

as in Iglehart (1963) and Veinott and Wagner. A fixed delivery lag can be incorporated by a standard redefinition of $G(\cdot)$; cf. e.g., Veinott and Wagner (1965), Veinott (1966) or other standard treatments of single item inventory models, e.g. Denardo (1982), Heyman and Sobel (1984), and Tijms (1986). The same form of $G(\cdot)$ arises under *random* lead times generated by an exogenous supply process that is *independent* of the sequence of demands and with the property that orders are received in the same sequence as they are placed; see Zipkin (1986, 1988) for a precise description. Under these lead time assumptions, $G(\cdot) = E_L G_L(\cdot)$, where $G_L(\cdot)$ is the one-period cost function under a fixed lead time L, and the expectation is taken over the lead time distribution. Clearly, if $G_L(\cdot)$ is convex, so is $G(\cdot)$ as a convex combination of convex functions. (An *approximate* model of the same form is often used when the lead times fail to satisfy the above properties; see e.g., Ehrhardt 1984.)

If $-G(\cdot)$ is unimodal and

$$\lim_{|y|\to\infty} G(y) > \min_y G(y) + K$$

an (s, S) policy is optimal when minimizing long-run average costs, as shown by Veinott. The following expression for the long-run average cost $c(s, S)$ of a given (s, S) policy is well known and easily verified from renewal theory or by characterizing the steady-state inventory position distribution; see e.g., Veinott and Wagner (1965)

$$c(s, S) = M(S - s)K + \sum_{j=0}^{S-s-1} m(j)G(S - j) \qquad (1)$$

where

$$m(0) = (1 - p_0)^{-1}, \quad M(0) = 0 \tag{2a}$$

$$m(j) = \sum_{l=0}^{j} p_l m(j-l), \quad j = 1, 2, \ldots \tag{2b}$$

and

$$M(j) = M(j-1) + m(j-1), \quad j = 1, 2, \ldots. \tag{2c}$$

Equations 3–5 provide a concise verification based on renewal theory. These formulas are needed in the next section. Under an (s, S) policy the inventory position process regenerates at every replenishment order epoch when the inventory position increases to S. Let M(j) (k(s, y)) be the expected total time (costs) until the next order is placed, when starting with an inventory position of s + j (y) units, j ≥ 1 (y > s).

It follows from the theory of regenerative processes (see, e.g., Ross 1970, proposition 5.9) that

$$c(s, S) = k(s, S)/M(S - s). \tag{3}$$

Fix s. It is clear that the functions M(·) and k(s, ·) satisfy the discrete renewal equations

$$M(j) = 1 + \sum_{i=0}^{j-1} p_i M(j-i), \quad j = 1, 2, \ldots$$

and

$$k(s, y) = G(y) + K \sum_{j=y-s}^{\infty} p_j + \sum_{j=0}^{y-s-1} p_j k(s, y-j) \quad y > s \tag{4}$$

with the unique solution (2) and

$$k(s, y) = K + \sum_{j=0}^{y-s-1} m(j) G(y-j) \quad y > s \tag{5}$$

respectively.

2. PROPERTIES OF THE COST FUNCTION C(·, ·) AND BOUNDS FOR s* AND S*

This section contains all of the results required to derive the algorithms, including some known bounds for optimal s^* and S^* values, with new and simpler proofs. Let $y_1^* = \min\{y : G(y) = \min G(x)\}$ and $y_2^* = \max\{y : G(y) = \min G(x)\}$ be the smallest and largest minimizers of $G(\cdot)$, respectively, $(-\infty < y_1^* \leqq y_2^* < \infty$ since

$$\lim_{|y| \to \infty} G(y) \geqq \min_x G(x) + K\Big).$$

Also define

$$\alpha_n \equiv \frac{M(n)}{M(n+1)}, \quad n = 1, 2, \ldots, .$$

Note that $0 < \alpha_n \leqq 1$. It follows directly from cost function (1) that

$$c(s-1, S) = \alpha_n c(s, S) + (1 - \alpha_n) G(s), \quad n = S - s \tag{6}$$

i.e., $c(s-1, S)$ is simply a weighted average of $c(s, S)$ and $G(s)$. The following properties of the cost function are therefore immediate.

Lemma 0. *For any fixed S and $s < S$*

a. $G(s) < (=, >) c(s-1, S) \leqslant (=, \geqslant) c(s, S)$ if $G(s) < (=, >) c(s, S)$.

b. $c(s-1, S) \leqslant (=, \geqslant) c(s, S)$ if $G(s) < (=, >) c(s-1, S)$.

Proof. The proof is by simple algebraic manipulation of (6).

Lemma 1 shows that $-c(s, S)$ is unimodal in s for fixed S and provides a useful characterization of optimal values of s (for fixed S).

Lemma 1. a. *For any given order-up-to level S, a reorder level $s^0 < y_1^*$ is optimal, i.e.,*

$$c(s^0, S) = \min_{s<S} c(s, S) \text{ if}$$

$$G(s^0) \geqq c(s^0, S) \geqq G(s^0 + 1). \tag{7}$$

Multiple optima of s for fixed S occur when at least one of the inequalities in (7) holds as an equality; that is, $s^0 - 1$ or $s^0 + 1$ are also optimal reorder levels for S if $G(s^0) = c(s^0, S)$ or $G(s^0 + 1) = c(s^0, S)$ hold, respectively.

b. *For any given order-up-to level S, there exists an optimal reorder level s^0 such that $s^0 < y_1^*$ and (7) holds.*

c. *For any given order-up-to level S, let s_l^0 and s_u^0 be the smallest and largest optimal reorder levels below y_1^*, respectively. (See part b.) Then $G(s_l^0) > c(s_l^0, S) = c(s_u^0, S) > G(s_u^0 + 1)$.*

Proof. a. Assume that (7) holds for some s^0 with $s^0 < y_1^*$. To show that s^0 is optimal (for S), we prove: i) $c(s-1, S) \geqq c(s, S)$ for $s \leqq s^0$; ii) $c(s, S) \leqq c(s+1, S)$ for $s \geqq s^0$. To prove i, we show by induction that

$$G(s) \geqq c(s-1, S) \geqq c(s, S) \tag{8}$$

for all $s \leqq s^0$. By applying Lemma 0a with $s = s^0$, the first inequality of (7) implies that (8) holds for $s = s^0$.

Assume that (8) holds for $s = y + 1$, that is

$$G(y+1) \geqq c(y, S) \geqq c(y+1, S) \quad \text{for some } y(< s^0). \quad (9)$$

Note that since $y < y_1^*$, $G(y) > G(y+1)$, and in view of (9)

$$G(y) \geqq c(y, S). \quad (10)$$

Applying Lemma 0a with $s = y$, we conclude that

$$G(y) \geqq c(y-1, S) \geqq c(y, S).$$

This completes the proof of i.

Next, to show that ii holds for $s^0 \leqq s < y_1^*$, we prove by induction that

$$G(s+1) \leqq c(s, S) \leqq c(s+1, S) \quad (11)$$

for all $s^0 \leqq s < y_1^*$. By applying Lemma 0b with $s = s^0 + 1$, the second inequality of (7) implies that (11) holds for $s = s^0$; assume that it holds for some $s = y - 1$ with $s_0 \leqq y - 1 < y_1^* - 1$, that is

$$G(y) \leqq c(y-1, S) \leqq c(y, S). \quad (12)$$

We show that (11) holds for $s = y$. Note again that $y < y_1^*$ implies that $G(y+1) \leqq G(y)$, which in conjunction with (12) implies that $G(y+1) \leqq c(y, S)$. By applying Lemma 0b with $s = y + 1$, this in turn implies that $c(y, S) \leqq c(y+1, S)$, thus completing the induction step.

For $y_1^* \leqq s < S$, the proof of ii is similar to that of i; the induction starts with $s = S - 1$ and uses the fact that $c(S-1, S) = K(1 - p_0) + G(S) \geqq G(S)$.

The remainder of part a follows immediately from Lemma 0b and c. Let s_l^0 be the lowest optimal reorder level for S. If $s_l^0 \geqq y_1^*$, then $c(s_l^0, S) \geqq G(s_l^0)$ in view of (1). Thus, $c(s_l^0 - 1, S) \leqq c(s_l^0, S)$ by Lemma 0a, contradicting the definition of s_l^0. Thus, $s_l^0 \leqq y_1^* - 1$.

If $G(s_l^0) \leqq c(s_l^0, S)$, then $c(s_l^0 - 1, S) \leqq c(s_l^0, S)$ by Lemma 0a, contradicting the definition of s_l^0. This proves the first inequality of c; the proof of the second inequality of c is similar. Finally, it follows from c and the fact that $G(y)$ is decreasing for $y < y_1^*$ that (7) holds for some s^0 with $s_l^0 \leqq s^0 \leqq s_u^0$.

The following corollary provides an efficient way to find an optimal reorder level for any given order-up-to level S.

Corollary 1. *For any given order-up-to level S, let $s^0 = max\{y < y_1^* : c(y, S) \leqq G(y)\}$. Then, (7) holds and s^0 is an optimal reorder level (for S).*

Proof. Since $G(s^0 + 1) < c(s^0 + 1, S)$, it follows from Lemma 0a that $G(s^0 + 1) < c(s^0, S)$ while $c(s^0, S) \leqslant G(s^0)$ by the definition of s^0. Thus, (7) holds and the optimality of s^0 follows from Lemma 1a.

We now derive bounds that apply to any optimal reorder level s^*. These follow rather straightforwardly from Lemma 1.

Corollary 2. (Bounds for s^*):

a. *Let s_l^* denote the smallest optimal reorder level $s_l^* \leqq \bar{s} \overset{\text{def}}{=} y_1^* - 1$.*

b. *Let s_u^* denote the largest optimal reorder level $< y_1^*$; if s^0 satisfies (7) for some arbitrary order-up-to level S, then $s^0 \leqq s_u^*$.*

Proof. a. The proof is immediate from Lemma 1b; b. Note that $G(s_u^* + 1) < c(s_u^*, S^*) \leqq c(s^0, S) \leqq G(s^0)$: The first inequality follows from Lemma 1c. Then $s^0 \leqq s_u^*$ follows from the fact that $G(s)$ is nonincreasing for $s \leqq s^0 < y_1^*$.

The upper bound $s^* \leqq \bar{s} =_{\text{def}} y_1^* - 1$ was first discovered by Veinott and Wagner. The *lower* bound $s^0 \leqq s^*$ (with s^0 as a reorder level satisfying (7) and, hence, for *some* order-up-to-level) is, to our knowledge, new. This bound can continuously be improved, as new optimal reorder levels for new values of S are determined. Corollary 3 establishes alternative lower bounds for s^*. While this corollary is not needed for the derivation of the algorithm in Section 3, we state it for the sake of completeness. Let c^* denote the optimal average cost value.

Corollary 3. (*Alternative lower bounds for s^**): *There exists an optimal reorder level s^* such that the following properties hold*:

a. Let $\underline{s}^* =_{\text{def}} \max\{y < y_1^* : G(y) \geqq c^*\}$. Then $\underline{s}^* \leqq s^*$.

b. Let $c \geqq c^*$ (e.g., c is the cost of an arbitrary policy) and $\underline{s}_c =_{\text{def}} \max\{y < y_1^* : G(y) \geqslant c\}$. Then $\underline{s}_c \leqq s^*$.

Proof. Part b is immediate from part a and the unimodality of $-G(\cdot)$. Let S^* be an optimal order-up-to level. It follows from Lemma 1b that there is an optimal reorder level s^* ($< y_1^*$) (for S^*) such that $G(s^* + 1) \leqq c^* = c(s^*, S^*) \leqslant G(s^*)$, so that $s^* \leqq s^*$.

Let $\underline{s} =_{\text{def}} \max\{y < y_1^* : G(y) > [\min_x G(x) + K]\}$. Veinott and Wagner's lower bound $\underline{s} \leqq s^*$ follows as a special case of Corollary 3 with $c = c(y_1^* - 1, y_1^*) = [K + G(y_1^*)/(1 - p_0)](1 - p_0) \leqq K + \min_x G(x)$.

Lemma 2 derives bounds for S^*.

Lemma 2. a. (*Veinott and Wagner's lower bound for S^**): *There exists an optimal policy (s^*, S^*) with $S^* \geqq \underline{S} =_{\text{def}} y_2^*$.*

b. (*New upper bound for S^**): *For every optimal policy (s^*, S^*)*

$$S^* \leqq \bar{S}^* =_{\text{def}} \max\{y \geqq y_2^* : G(y) \leqq c^*\}.$$

c. *Any upper bound c of c^* (e.g., c is the average cost of an arbitrary policy) defines an upper bound* $\bar{S}_c =_{\text{def}} \max\{y \geqq y_2^* : G(y) \leqq c\}$. *Moreover,* $\bar{S}^* \leqq \bar{S}_{c_1} \leqq \bar{S}_{c_2}$ *if* $c^* \leqq c_1 \leqq c_2$.

Proof. a. Let (s_u^*, S_u^*) be an optimal (s, S) policy that maximizes the value of S^*. Assume that $S_u^* < y_2^*$. Since $G(y)$ is nonincreasing for $y \leqq y_2^*$, $G(S_u^* - j) \geqq G(S_u^* - j + 1)$ for all $j \geqq 0$ and, hence, $c(s_u^*, S_u^*) \geqq c(s_u^* + 1, S_u^* + 1)$ (see (1)), which contradicts the definition of S_u^*.

b. Assume to the contrary that an optimal (s^*, S^*) exists with $G(S^*) > c^*$. Note that $Pr(D < S^* - s^*) > 0$ (otherwise $c^* = K + G(S^*) \geqq G(S^*)$; see (1)). Let X be a one-period demand truncated on the interval $[0, S^* - s^* - 1]$, i.e., $X =_{\text{def}} (D \mid D < S^* - s^*)$ or $Pr(X = j) = P_j / Pr(D < S^* - s^*)$, $j = 0, \ldots, S^* - s^* - 1$. Consider the modification of the (s^*, S^*) policy in which $S^* - X(> s^*)$ is used as the (random) order-up-to level instead of S^* (i.e., an independent realization x of X is determined whenever the inventory position reaches or drops below s^*, and the inventory position increases to S^*-x). Under this policy, the system regenerates at each replenishment epoch as well, so that its long-run average cost c_* is given by $c_* = \underline{k}/\underline{M}$ where

$$\underline{k} = \sum_{j=0}^{S^*-s^*-1} p_j k(s^*, S^* - j) / Pr(D < S^* - s^*)$$

$$\underline{M} = \sum_{j=0}^{S^*-s^*-1} p_j M(S^* - j - s^*) / Pr(D < S^* - s^*).$$

Note, however, from (3) and (4) that

$$c^* = c(s^*, S^*) = \frac{k(s^*, S^*)}{M(S^* - s^*)}$$

$$= \frac{G(S^*) + KPr(D \geqq S^* - s^*) + \underline{k} Pr(D < S^* - s^*)}{1 + \underline{M} Pr(D < S^* - s^*)}$$

$$> \frac{c^* + c_* \underline{M} Pr(D < S^* - s^*)}{1 + \underline{M} Pr(D < S^* - s^*)}$$

and hence, $c_* < c^*$ contradicts the definition of c^*. Part c is immediate from part b.

We observe that Veinott and Wagner's upper bound $\bar{S} =_{\text{def}} \max\{y \geqq y_2^* : G(y) \leqq \min_x G(x) + K\}$ arises as a special case of Lemma 2c: note that $c(y_2^* - 1, y_2^*) \leqq \min G(x) + K$. Lemma 2c may be used to identify increasingly tighter upper bounds for S^* as increasingly better policies (s, S) are found.

Our last lemma allows for a major reduction in our algorithm's complexity. For any fixed order-up-to level S, let

$$c^*(S) \overset{\text{def}}{=} \min_{s<S} c(s, S).$$

S is said to be *improving* upon S^0, if $c^*(S) < c^*(S^0)$.

Lemma 3. *For a given order-up-to level $S^0 (\geqq y_1^*)$, let $s^0 (< y_1^*)$ be an optimal reorder level.*

a. $c^*(S) < c^*(S^0)$ if and only if $c(s^0, S) < c(s^0, S^0)$.

b. Assume that (7) holds with $S = S^0$. If $c(s^0, S') < c(s^0, S^0)$ for some $S' (\geqq y_1^*)$, then $s' =_{\text{def}} \min\{y \geqq s^0 : c(y, S') > G(y+1)\}$ is optimal for S'; moreover, $s' < y_1^*$ and $G(s') \geqq c(s', S') > G(s'+1)$.

Proof. The "if" part of a is trivial. To prove the "only-if" part, assume that $c(s^0, S) \geqq c(s^0, S^0)$. In view of Lemma 1b, it suffices to show that $c(s, S) \geqq c(s^0, S^0)$: i) for $y_1^* > s > s^0$, and ii) for $s < s^0$. We only prove i; the proof of ii is analogous.

Let $\beta = M(S - s)/M(S - s^0)$. Note that $0 < \beta \leqq 1$. In view of (1)

$$c(s^0, S)$$

$$= \frac{K + \sum_{j=0}^{S-s-1} m(j) G(S-j) + \sum_{j=S-s}^{S-s^0-1} m(j) G(S-j)}{M(S - s^0)}$$

$$= \frac{c(s, S) M(S-s) + \sum_{j=S-s}^{S-s^0-1} m(j) G(S-j)}{M(S - s^0)}$$

$$\leqq \frac{c(s, S) M(S-s) + \sum_{j=S-s}^{S-s^0-1} m(j) c(s^0, S)}{M(S - s^0)}$$

$$= \beta c(s, S) + (1 - \beta) c(s^0, S).$$

This inequality follows from $c(s^0, S) \geqq G(s^0 + 1) \geqq G(y)$ for $y \geqq s^0 + 1$ in view of Lemma 1 and the unimodality of $-G(\cdot)$. We conclude that $c(s^0, S) < (\leqq) c(s, S)$ if $c(s^0, S^0) < (=) c(s^0, S)$.

b. Note that s' is well defined in view of Corollary 2b. Note that $G(s') \geqq c(s'-1, S') \geqq c(s', S') > G(s'+1)$, where the first and the third inequality follow from the definition of s', and the second inequality follows from the first and Lemma 0a. Lemma 1a thus establishes the optimality of s' (for S'). Finally, $s' < y_1^*$ in view of $G(s') > G(s'+1)$.

3. THE ALGORITHM AND ITS COMPLEXITY

The lemmas and corollaries of the previous section suggest the following algorithm. Let y^* be a minimum point of the $G(\cdot)$ function.

Algorithm

Step 0. $s := y^*$;
$S_0 := y^*$;
Repeat $s := s - 1$ *until* $c(s, S_0) \leqq G(s)$;
$s_0 := s$; $c^0 := c(s_0, S_0)$; $S^0 := S_0$; $S := S^0 + 1$;

Step 1. While $G(S) \leqq c^0$ *do*
begin If $c(s, S) < c^0$
then begin $S^0 := S$.
While $c(s, S^0) \leqq G(s+1)$ *do* $s := s + 1$;
$c^0 = c(s, S^0)$;
end;
$S := S + 1$;
end.

The algorithm is easy to understand. In Step 0, we enter with an initial order-up-to level $S^0 = y^*$, with y^* as an arbitrary minimum of the $G(\cdot)$ function. We then find an optimal corresponding reorder level s^0 by decreasing s from y^* with step size 1 until $c(s, S^0) \leqq G(s)$. Optimality of s^0 (for S^0) follows from Corollary 1.

In Step 1, we search for the smallest value of S that is larger than S^0, which improves on S^0. S is increased with increments of one. A single comparison of $c(s^0, S)$ and $c(s^0, S^0)$ suffices to verify whether a given value for S improves on S^0 (see Lemma 3a). If case S *is* improving, S^0 is updated to S and the new corresponding optimal reorder level s^0 is determined by incrementing the old value of s^0 by units of one, until $c(s, S^0) > G(s+1)$. The existence of such a reorder level s^0, its optimality (for the new value S^0) and $s^0 < y^*$ are all guaranteed by Lemma 3b.

Finally, note that whenever Step 1 is initiated, c^0 represents an upper bound for c^* (and in fact is the best available such bound). In view of Lemma 2c the search for an improving value of S may be terminated as soon an $G(S) > c^0$. Indeed, at the last iteration of the algorithm, when $S^0 = S^*$ and $s^0 = s^*$ for some optimal policy (s^*, S^*), we have $c^0 = c^*$ and $S^0 \leqq \bar{S}^*$ in view of Lemma 2b. It follows that the test in the outer *while-do* loop of Step 1 fails when $S := \bar{S}^* + 1$, in view of the definition of $\bar{S}^*$; see Lemma 2b.

Complexity of the Algorithm

Consider the two-dimensional integer grid with s on the horizontal axis and S on the vertical one: Each point (s, S) with $s < S$ represents a policy. We assume that y^* is given. (Note that y^* represents a solution of a one-period newsboy problem. The most efficient procedure for its determination depends on the specific form of the $G(\cdot)$ function; e.g., with linear holding and penalty costs, y^* is obtained as a fractile of the lead time demand distribution.) In the following discussion, let s_0 denote the value of s obtained at the end of Step 0, which is optimal for $S = y^*$.

Starting from the point (s_0, y^*) Step 1 leads the search through a vertically-up and horizontally-right path ending at $(s^*, \bar{S}^* + 1)$; see Figure 1 and the previous observations. Regardless of the exact path followed, it consists of exactly $(\bar{S}^* - y^* + 1)$ *vertical* moves, in which the S-value increases by one with s fixed, and $(s^* - s_0)$ *horizontal* moves, in which the s-value increases by one with S fixed.

We count the number of operations needed for the algorithm. Note that each vertical move (except for the last one which involves a single comparison only) corresponds with one execution of the outer *while-do* loop in Step 1. This loop consists of one evaluation of the $c(\cdot, \cdot)$ function, one addition and at most three comparisons (two if S^0 is not updated). Each horizontal move requires an additional execution of the inner *while-do* loop which requires one *update* of the $c(\cdot, \cdot)$ function, one comparison and one addition.

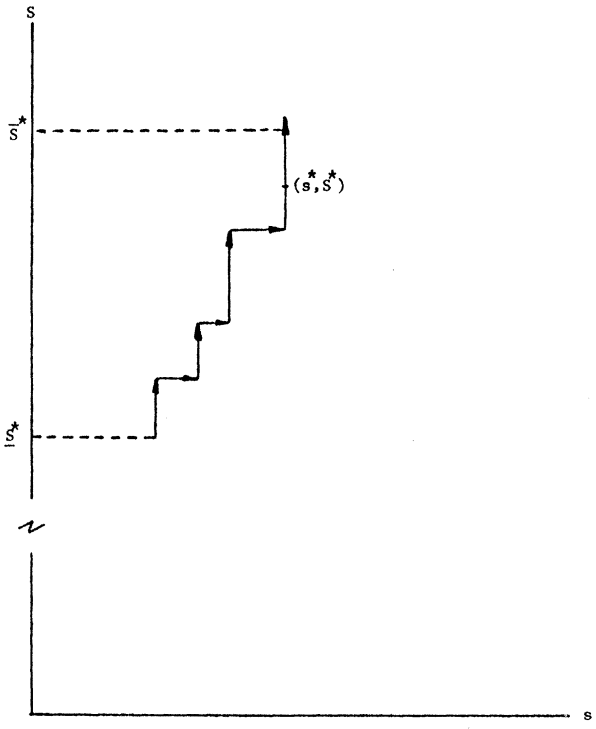

Figure 1. A typical execution of the algorithm.

Straightforward evaluation of $c(s, S)$ via (1) requires $2(S-s)+1$ elementary operations ($(S-s)$ additions, $(S-s)$ multiplications and one division) assuming all required $G(\cdot)$, $m(\cdot)$ and $M(\cdot)$ values are available.

Indeed, evaluations of the $c(\cdot, \cdot)$ function in the *if* test of the outer *while-do* loop, i.e., when performing vertical moves, need to be done via (1). When computing the numerator in (1), we obtain the numbers

$$A_i(S) = K + \sum_{j=0}^{i} m(j)G(S-j) \qquad \text{for } i = 0, 1, \cdots, S-s-1$$

as intermediate results. Storing these numbers until the next update of S simplifies the evaluations of the $c(\cdot, \cdot)$ function in the subsequent series of horizontal moves, i.e., in the inner *while-do* loop. For, let s^0 denote the value of s at the start of such an inner loop. Observe that in the course of this inner loop S is fixed and for this value all numbers $\{A_i(S): i = 0, 1, \ldots, S-s^0-1\}$ are available when stored in the process of the evaluation of $c(s^0, S)$ in the last preceding *if* test. Since only values of $s > s^0$ are needed in the inner loop, all required $c(\cdot, \cdot)$ values can be obtained by a *single* division of one of these A-numbers and an $M(\cdot)$-value.

The total number of elementary operations needed in Step 1 is thus bounded by

$$\sum_{S=y^*+1}^{\bar{S}^*} [2(S-s_0)+1] + 4(\bar{S}^* - y^*) + 1$$

for the vertical moves (including the final comparison verifying that $S > \bar{S}^*$)

$$+3(s^* - s_0) \qquad \text{for the horizontal moves}$$

$$= (\bar{S}^* + y^* - 2s_0 + 6)(\bar{S}^* - y^*) + 3(s^* - s_0) + 1.$$

The remaining computational effort of the algorithm consists of the following components:

i. Determination of the $(\bar{S}^* - s_0 + 2)$ values $\{G(s_0), G(s_0+1), \ldots, G(\bar{S}^*+1)\}$;

ii. Determination of s_0 and $c(s_0, S_0)$; this requires successive evaluation of $A_i(S_0)$ for $i = 0, 1 \ldots S_0 - s_0 - 1$, which can be done in $2(S_0 - s_0)$ operations, $(y^* - s_0)$ additional divisions to calculate $c(s, S_0)$ for $s = y^* - 1, \ldots, s_0$ and as many comparisons between $c(s, S_0)$ and $G(s)$. Since $S_0 = y^*$, the total number of elementary operations required for ii is thus given by

$$2(S_0 - s_0) + 2(y^* - s_0) = 4(y^* - s_0).$$

iii. Determination of all required $m(\cdot)$ and $M(\cdot)$ values. The exact number depends on the specific path followed by the algorithm but can obviously be bounded by $(\bar{S}^* - s_0)$. Unless a closed form expression of the renewal function is available, the $m(\cdot)$ and $M(\cdot)$ values are recursively obtained from (2). Observe from (2b) that

$$m(j) = (1-p_0)^{-1} \sum_{l=1}^{j} p_l m(l-j)$$

$$= m(0) \sum_{l=1}^{j} p_l m(l-j).$$

Computation of $m(0)$ thus requires two operations, of $m(j)$ $2j$ operations ($j = 1, 2, \ldots$) and a single addition to obtain $M(j+1)$ from $M(j)$ and $m(j)$; see (2c) ($j \geqq 0$). The total number of operations required for iii is thus bounded by

$$2 + \sum_{j=1}^{\bar{S}^* - s_0 - 1} 2j + (\bar{S}^* - s_0 - 1)$$

$$= 2 + (\bar{S}^* - s_0)(\bar{S}^* - s_0 - 1) + (\bar{S}^* - s_0 - 1)$$

$$= (\bar{S}^* - s_0)^2 + 1.$$

The total number of elementary operations needed for ii and iii is thus bounded by

$$B \equiv (\bar{S}^* + y^* - 2s_0 + 6)(\bar{S}^* - y^*) + 3(s^* - s_0) + 2 + 4(y^* - 2s_0) + (\bar{S}^* - s_0)^2$$

$$= 2\Delta_1^2 - \Delta_2^2 + 6\Delta_1 - 2\Delta_2 + 3\Delta_3 + 2$$

$$\leqq 2(\Delta_1+1)^2 + 2(\Delta_1+1) - (\Delta_2+1)^2 + 3\Delta_3$$

where $\Delta_1 = \bar{S}^* - s_0$, $\Delta_2 = y^* - s_0$ and $\Delta_3 = s^* - s_0$. Note that $(\Delta_2+1)^2 \geqq (\Delta_3+2)^2 \geqq 3\Delta_3$ so that the above bound is $\leqq 2(\Delta_1+1)^2 + 2(\Delta_1+1)$.

We thus conclude the discussion with the following theorem.

Theorem 1. *a. The algorithm terminates with (s, S^0) being an optimal policy and $c^0 = c^*$.*

b. Assuming that y^* is known, the algorithm requires no more than $B = 2(\Delta_1+1)^2 + 2(\Delta_1+1) - (\Delta_2+1)^2 + 3\Delta_3$ elementary operations and (Δ_1+2) evaluations of the G-function.

c. Assuming that y^* and $G(s_0+1), \ldots, G(\bar{S}^*+1)$ are known, let $R(\Delta_1) =_{\text{def}}$ [the number of elementary operations needed for the algorithm] [the number of

elementary operations needed to calculate to single value $c(s_0, \bar{S}^*)$] $R(\Delta_1) \leqslant 2.4$ *and* $\lim_{\Delta_1 \to \infty} R(\Delta_1) = 2$.

Proof. Parts a and b have been verified above. To prove c note that the one-time evaluation of $c(s_0, \bar{S}^*)$ may be achieved via (1) or (3). It is easy to verify that the former approach is cheaper. Calculating $m(0), \ldots, m(\bar{S}^* - s_0 - 1)$ and $M(1), \ldots, M(\bar{S}^* - s_0)$ requires $(\bar{S}^* - s_0)^2 + 1 = \Delta_1^2 + 1$ operations; see above. The remaining calculation of $c(s^0, \bar{S}^*)$ (via 1) takes $2\Delta_1 + 1$ operations resulting in a total of $\Delta_1^2 + 2\Delta_1 + 2 = (\Delta_1 + 1)^2 + 1$ operations. The complexity ratio is thus bounded by

$$R(\Delta_1) = \frac{B}{(\Delta_1 + 1)^2 + 1} \leqq \frac{2(\Delta_1 + 1)^2 + 2(\Delta_1 + 1)}{(\Delta_1 + 1)^2 + 1}.$$

One easily verifies that $R(\Delta_1)$ is nonincreasing, so that $R(\Delta_1) \leqslant R(1) \leqslant 12/S = 2.4$. Clearly,

$$\lim_{\Delta_1 \to \infty} R(\Delta_1) = 2.$$

In evaluating the complexity ratio R in part c, we exclude the computational effort to determine all necessary $G(\cdot)$ values because the latter heavily depends on the specific choice of the $G(\cdot)$ function. Note, however, that the algorithm requires $\{G(s_0), \ldots, G(\bar{S}^* + 1)\}$ (see part b, while evaluation of $c(s_0, \bar{S}^*)$ requires the same values except for $G(s_0)$ and $G(\bar{S}^* + 1)$ i.e., Δ_1 instead of $(\Delta_1 + 2)$ values. The computational effort for determining the $G(\cdot)$ values tends to be minor compared to the remainder, see Section 4; moreover, its inclusion tends to *reduce* the complexity ratio in part c.

Optimal order-up-to levels S^* may be far from y^* and reasonable initial estimates for an optimal value of S^* may be available, e.g., from simple heuristics (such as Ehrhardt's power approximation, see the Introduction) or known optimal values for problems with similar parameter values. It is therefore useful to observe that (a slight variation of) the algorithm allows for an arbitrary starting value S_0 of S. Indeed, the enumeration of all S in $[y^*, \bar{S}^* + 1]$ in Step 1 may be carried out in any desired sequence. The advantage of choosing a good heuristic value $S_0 > y^*$ is that by doing so the initial s_0 value is likely to be closer to an optimal reorder level s^*, thus reducing the magnitude of Δ_1. This complexity count is easily adapted for the more general version. (We merely need to replace Δ_1 by $\max(\bar{S}^*, S_0) - s_0$.)

Algorithm for General Initial Value $S_0 \geqq y^*$

Step 0. $s := y^*$;
  Repeat $s := s - 1$ *until* $c(s, S_0) \leqq G(s)$;
  $s_0 := s$; $c^0 := c(s_0, S_0)$; $S^0 := S_0$; $S := S^0$;

Step 1. While $S > y^*$ *do*
  begin S := S − 1;
    If c(s, S) < c^0.
    then begin
      S^0 := S;
      while c(s, S^0) > G(s + 1) do s := s + 1;
      c^0 = c(s, S^0);
      end;
  end.
S := S_0 + 1;
While G(S) ≦ c^0 *do*
  begin If c(s, S) < c^0
    then begin S^0 := S;
      while c(s, S^0) ≦ G(s + 1) *do* s := s + 1;
      c^0 = c(s, S^0);
      end;
  S := S_0 + 1;
  end.

For any given precision ε, the algorithm may be terminated prior to convergence with an ε-optimal policy, by invoking the bounds in Federgruen and Zipkin (1984, Theorem 3), themselves applications of Odoni (1969) and Hastings (1971). However, the use of the bounds does not appear to be necessary or recommendable given the simplicity with which an *optimal* policy can be determined.

4. NUMERICAL EXAMPLES

Table I below exhibits how our algorithm performs on 24 test problems. To this end, we have chosen the example model in Veinott and Wagner with linear holding and backlogging costs, zero lead time and Poisson distributed one-period demands. The following parameters are common to all 24 problems and identical to those used in Veinott and Wagner:

fixed setup cost $K = 24$;

holding cost rate $h = 1$;

penalty cost rate $p = 9$.

The problems differ with respect to the mean one-period demand μ. In the first part of the table, μ varies from 10 to 75 in increments of 5; in the second part we evaluate all remaining cases in Veinott and Wagner. We observe that in all 24 cases the bound B for the computational effort, exclusive of the evaluation of the required $G(\cdot)$ values comes within 10% of predicting the actual number of computations required. The ratio $\hat{R}$ defined by (the number of elementary operations required to find and evaluate an *optimal* policy)/(the number of operations to evaluate the single policy $(s_0, \bar{S}^*)$) varies in the

Table I
Performance of Algorithm on 24 Test Problems

μ	s^*	S^*	c^*	y^*	$\underline{s}$	s_0	$\bar{S}$	$\bar{S}^*$	$\bar{S}-\underline{s}$	u	IS	B	#+	#*	#/	#COM	#TOT	#SNG	$\hat{R}$
10	6	40	35.022	14	3	3	53	45	50	39	26	3,650	1,559	1,564	45	129	3,297	1,892	1.74
15	10	49	42.698	20	7	7	83	57	76	47	29	5,118	2,240	2,247	53	149	4,689	2,652	1.77
20	14	62	49.173	26	12	12	92	69	80	55	29	6,626	3,031	3,041	59	161	6,292	3,422	1.84
25	19	56	54.262	32	16	16	98	79	82	60	19	8,041	3,621	3,631	66	152	7,470	4.160	1.80
30	23	66	57.819	37	21	21	103	87	82	64	16	8,830	4,098	4,110	68	151	8,427	4,556	1.85
35	28	77	61.215	43	26	26	109	96	83	68	15	9,907	4,626	4,639	72	156	9,493	5,112	1.86
40	33	87	64.512	48	31	31	115	104	84	71	13	10,783	5,061	5,074	75	158	10,368	5,550	1.87
45	37	97	67.776	54	36	36	121	112	85	75	12	11,655	5,584	5,600	77	160	11,421	6,006	1.90
50	42	108	70.975	59	41	41	126	120	85	78	10	12,603	6,053	6,069	80	162	12,364	6,480	1.91
55	47	118	74.149	65	46	46	132	129	86	82	8	13,884	6,686	6,703	84	165	13,638	7,140	1.91
60	52	129	77.306	70	51	51	137	137	86	85	4	14,916	7,200	7,217	87	163	14,667	7,656	1.92
65	56	75	78.518	75	56	56	143	143	87	87	0	15,265	7,467	7,486	87	156	15,196	8,732	1.94
70	62	81	79.037	81	62	62	149	149	87	87	0	15,265	8,467	7,486	87	156	15,196	7,832	1.94
75	67	86	79.554	86	67	67	154	154	87	87	0	15,265	7,467	7,486	87	156	15,196	7,832	1.94
21	15	65	50.406	27	13	13	93	71	80	56	27	6,862	3,145	3,155	60	159	6,519	3,540	1.84
22	16	68	51.632	28	14	14	94	73	80	57	26	7,102	3,262	3,272	61	159	6,754	3,660	1.85
23	17	52	52.757	29	15	15	95	75	80	58	23	7,346	3,380	3,390	62	155	6,987	3,782	1.85
24	18	54	53.518	30	15	15	96	77	81	59	21	7,818	3,513	3,522	65	155	7,255	4,032	1.80
51	43	110	71.611	60	42	42	127	122	85	79	10	12,927	6,213	6,229	81	164	12,687	6,642	1.91
52	44	112	72.246	61	43	43	129	124	86	80	9	13,255	6,374	6,390	82	164	13,010	6,806	1.91
59	51	126	76.679	69	50	50	136	135	86	84	5	14,568	7,027	7,044	86	163	14,320	7,482	1.91
61	52	131	77.929	71	52	52	139	138	87	86	2	14,913	8,293	7,312	86	158	14,849	7,656	1.94
63	54	73	78.287	73	54	54	141	141	87	87	0	15,265	7,467	7,486	87	156	15,196	7,832	1.94
64	55	74	78.402	74	55	55	142	142	87	87	0	15,265	7,467	7,486	87	156	15,196	7,832	1.94

u = the maximum index j for which $m(j)$ is computed;
IS = the number of improving S values;
B = the bound for the number of operations needed as defined in the text;
+ (*, /) = the actual number of additions (multiplications, divisions);
#COM = the actual number of comparisons;
#TOT = the actual total number of elementary operations (= (# +) + (#*) + (#/) + (#COM));
#SNG = the number of operations needed for computing a single cost value $c(s_0, \bar{S}^*)$;
$\hat{R}$ = #TOT/#SNG.

range [1.74, 1.94] and is therefore close to the theoretical bound of 2.4; see Theorem 1. Evaluation of the necessary $G(\cdot)$ values requires only a few hundred operations both for the computation of $c(s_0, \bar{S}^*)$ and for the optimization algorithm. (The latter requires only 4 more operations than the former.) Note also that in all 24 problems $\bar{S} - \underline{s} \leqq 100$ and even the "largest" problems require no more than about 15,000 elementary operations. On an IBM 4381 for example, an elementary operation requires 69 nanoseconds, so that all of the above problems can be solved in less than one millisecond.

5. CONTINUOUS REVIEW SYSTEMS; THE DISCOUNTED COST CRITERION

(s, S) policies continue to be optimal in many continuous review systems with fixed order lead times. This applies, for example, to models with compound Poisson demands (see Beckmann 1961, Veinott 1966, or Stidham 1986). Moreover, the long-run average cost is of the form (1) with $G(y)$ the expected (total) costs in an interdemand period and K replaced by K/(mean interdemand time).

The same is true in the more general case where the demand process is compound renewal provided that orders can only be placed at demand epochs, as argued in Federgruen and Zipkin (1985, p. 426). We repeat this argument in a slightly more detailed form: construct a related discrete-time model as follows. The demand in each period is the demand size at renewal epochs in the original model. The one-step cost function is the expected cost until the next renewal (after demand, but before the order, if any) in the original model. Under standard cost assumptions in the continuous model, this function is quasiconvex (i.e., its negative is unimodal). This may be verified from the fact that the steady-state inventory *level* $IL(\infty)$ may be characterized as

$$IL(\infty) = IP(\infty) - D(\infty \mid L) \tag{13}$$

with $IP(\infty)$ the steady-state inventory position, $D(\infty \mid L)$ the steady-state demand in a lead time, and with $IP(\infty)$ and $D(\infty \mid L)$ independent of each other; see Sahin (1979) and Zipkin (1986). We refer to Zipkin (1988) for a discussion of continuous-time systems with other types of demand processes in which (13) continues to hold and the one-step expected cost function continues to be quasiconvex.

As for discrete-time systems, the same form of the one-step expected cost function arises under *random* lead times provided the lead time process satisfies the properties mentioned in Section 1; see also Zipkin (1986).

As shown in Veinott and Wagner, the total discounted cost over an infinite horizon under an (s, S) policy is also of the form (1) with the renewal density replaced by a so-called discount renewal density. This term was introduced by Veinott and Wagner (see *ibid.* for a detailed specification). (This analogy applies both to the discrete- and continuous-time models.) There exists an optimal (s, S) policy under this criterion as well (see Iglehart 1963).

Our analysis in this paper is solely based on the cost function $c(\cdot, \cdot)$ being of the form (1). This implies that all of the results we present for minimizing long-run average costs in discrete-time models apply to the alternative models as well.

APPENDIX

Comparison With Existing Algorithms

Veinott and Wagner (1965), Bell (1970) and Archibald and Silver's (1978) methods are all based on (partial) enumeration of the policy pairs (s, S) in the rectangle $\{s \leqq s < y_1^*;\ y_2^* \leqq S \leqq \bar{S}\}$. We confine our discussion to Archibald and Silver, the most recent of these three that builds on Veinott and Wagner. Here, the long-run average cost of an (s, S) policy is written as a function $\bar{c}(s, \Delta)$ of s and $\Delta =_{\text{def}} S - s$. A limited number of properties of the $\bar{c}(\cdot, \cdot)$ function are exploited, most notably the fact that $\bar{c}$ is convex (and, hence, $-\bar{c}$ is unimodal) in s for fixed Δ, provided the $G(\cdot)$ function is convex itself. (Recall that $G(\cdot)$ fails to be convex in the presence of *stockout* penalty costs.) No complexity bound is provided. In fact, it appears impossible to bound the number of (s, Δ) pairs that need to be evaluated to anything significantly less than the total number of $(\bar{S} - \underline{S})\ (\bar{s} - \underline{s})$. Archibald and Silver derive bounds for the optimal value of Δ that are tighter than the original bounds (0 and $\bar{S} - \underline{s}$) derived by Veinott and Wagner. They also show how these bounds may be iteratively improved. However, the computation of these bounds is relatively involved, and they appear to achieve a modest reduction of the feasible span as compared to the simple Veinott and Wagner bounds.

As substantiated in the Introduction, Federgruen and Zipkin's method appears relatively efficient on the basis of the 768 problems on which it was tested. No complexity bound is provided, however, and it appears in fact impossible to bound even the *number* of "policy improvements iterations" which this procedure consists of. Even though this method is derived by tailoring a general policy iteration method for Markov decision problems, it is nevertheless possible to make a number of interesting comparisons with the algorithm above.

An iteration of the policy iteration method starts with some policy $(s^{\text{old}}, S^{\text{old}})$ and terminates with the conclusion that $(s^{\text{old}}, S^{\text{old}})$ is optimal or with an *improving* policy $(s^{\text{new}}, S^{\text{new}})$, i.e., either $c(s^{\text{new}}, S^{\text{new}}) < c(s^{\text{old}}, S^{\text{old}})$ or $c(s^{\text{new}}, S^{\text{new}}) = c(s^{\text{old}}, S^{\text{old}})$ and the so-called relative cost function of $(s^{\text{new}}, S^{\text{new}})$ dominates that of $c(s^{\text{old}}, S^{\text{old}})$ (pointwise). (See below for a definition of this function.) Such an iteration consists of two parts: i) the Value Determination, and ii) the Policy Improvement.

The Policy Improvement part starts with identifying S^{new} such that $c(s^{\text{old}}, S^{\text{new}}) < c(s^{\text{old}}, S^{\text{old}})$; if no such value can be found, one sets $S^{\text{new}} = S^{\text{old}}$. The actual search is based on finding a value S^{new} with a lower relative cost; see (10) in Federgruen and Zipkin. It is, however, easy to verify that this occurs if and only if $c(s^{\text{old}}, S^{\text{new}}) < c(s^{\text{old}}, S^{\text{old}})$ and the work we require is about the same as that to find S^{new} on the basis of the $c(\cdot, \cdot)$ function. (This is observed on page 1282 of Federgruen and Zipkin; the earlier method "1" of Johnson (1968) identifies S^{new} on the latter basis, indeed.) With S^{new} determined, a corresponding reorder level s^{new} is found. Note that s^{new} may fail to be optimal for S^{new}.

Recall that the algorithm consists of sequences of vertical moves alternating with sequences of horizontal moves (see the figure). It may thus be viewed as consisting of a number of iterations as well, where an iteration is defined as *one* sequence of vertical moves, followed by one sequence of horizontal moves. Note that an iteration starts with a policy (s, S) and terminates with a policy (s', S'), where s and s' are *optimal* with respect to S and S', respectively. The work performed in such an iteration is of the same order as that of the Policy Improvement step in a single iteration of the policy iteration method. (In fact, the work associated with the vertical moves is comparable to that of the search for S^{new} and the work required to perform the horizontal moves is about the same as that involved in determining s^{new}. Note, however, that in our case only moves to the right are needed, i.e., the equivalent of the search in (12)

of Federgruen and Zipkin is avoided due to Corollary 1 and the fact that the iteration's starting value of s is optimal for the starting value of S.)

The algorithm avoids, however, the relatively expensive Value Determination parts altogether. Another essential difference with the policy iteration method is the fact that an iteration in the algorithm succeeds in eliminating an entire interval for S and an entire interval for s, while no such eliminations appear possible in the policy iteration method. As a consequence, we can verify that the work required to do the very last Policy Improvement Step (i.e., when optimality is reached) is about the same as that of Step 1 in the algorithm, i.e., that of the entire algorithm with the exception of the initialization Step 0.

The policy iteration method has, on the other hand, the advantage of achieving somewhat more than an average cost optimal policy only. It finds among all such policies, one with a pointwise minimal relative cost function. The relative cost function of a policy is defined with respect to an arbitrary reference state, e.g., that corresponding with a zero inventory position; it specifies for each starting inventory position, the difference in *total* expected costs over an infinitely long period of time by starting with that inventory position as opposed to starting with an inventory position at the reference (say 0) level. See, e.g., Tijms (1986, Chapter 3) for a general discussion of "relative value functions" and in particular, Hordijk and Tijms (1975) for a verification of the interpretation in our specific inventory model.

ACKNOWLEDGMENT

We would like to thank Evan Porteus, the Associate Editor, for his thorough reading and helpful comments on a previous version of this paper.

REFERENCES

ARCHIBALD, B., AND E. SILVER. 1978. (s, S) Policies under Continuous Review and Discrete Compound Poisson Demands. *Mgmt. Sci.* **24**, 899–908.

BECKMANN, M. 1970. An Inventory Model for Arbitrary Interval and Quantity Distributions of Demands. *Mgmt. Sci.* **8**, 35–37.

BELL, C. 1970. Improved Algorithms for Inventory and Replacement Stock Problems. *SIAM J. Appl. Math.* **18**, 558–566.

DENARDO, E. 1982. *Dynamic Programming*. Prentice-Hall, Englewood Cliffs, N.J.

EHRHARDT, R. 1979. The Power Approximation for Computing (s, S) Inventory Policies. *Mgmt. Sci.* **25**, 777–786.

EHRHARDT, R. 1984. (s, S) Policies for a Dynamic Inventory Model With Stochastic Lead Times. *Opns. Res.* **32**, 121–132.

FEDERGRUEN, A., AND Y. S. ZHENG. 1988. A Simple and Efficient Algorithm for Computing Optimal (r, Q) Policies in Continuous-Review Stochastic Inventory Systems. Working Paper, Graduate School of Business, Columbia University, New York.

FEDERGRUEN, A., AND P. ZIPKIN. 1984. An Efficient Algorithm for Computing Optimal (s, S) Policies. *Opns. Res.* **34**, 1268–1285.

FEDERGRUEN, A., AND P. ZIPKIN. 1985. Computing Optimal (s, S) Policies in Inventory Models With Continuous Demands. *Adv. Appl. Prob.* **17**, 424–442.

FREELAND, J., AND E. PORTEUS. 1980a. Evaluating the Effectiveness of a New Method for Computing Approximately Optimal (s, S) Inventory Policies. *Opns. Res.* **28**, 353–364.

FREELAND, J., AND E. PORTEUS. 1980b. Easily Computed Inventory Policies for Periodic Review Systems: Shortage Cost and Service Level Models. Working Paper, Graduate School of Business, Stanford University, Stanford, Calif.

HASTINGS, N. 1971. Bounds on the Gain of a Markov Decision Process. *Opns. Res.* **19**, 240–244.

HEYMAN, D., AND M. SOBEL. 1984. *Stochastic Models in Operations Research*, Vol. I. McGraw-Hill, New York.

HORDIJK, A., AND H. TIJMS. 1974. Convergence Results and Approximations for Optimal (s, S) Policies. *Mgmt. Sci.* **20**, 1432–1438.

IGLEHART, D. 1963. Dynamic Programming and Stationary Analysis in Inventory Problems. In *Multi-Stage Inventory Models and Techniques*, chap. 1, H. Scarf, D. Guilford and M. Shelly (eds.). Stanford University Press, Stanford, Calif.

JOHNSON, E. 1968. On (s, S) Policies. *Mgmt. Sci.* **15**, 80–101.

KAKUMANU, P. 1977. Relation Between Continuous and Discrete Time Markovian Decision Problems, *Naval Res. Logist. Quart.* **24**, 431–439.

KUENLE, C., AND H. KUENLE. 1977. *Durchschnittsoptimale Strategien in Markovschen Entscheidungsmodellen bei Unbeschraenkten Kosten*, *Math. Operations forsch . Statis. Ser. Optimization.* **8**, 549–564.

NADDOR, E. 1975. Optimal and Heuristic Decisions in Single- and Multi-Item Inventory Systems. *Mgmt. Sci.* **21**, 1234–1249.

ODONI, A. 1969. On Finding the Maximal Gain for Markov Decision Processes. *Opns. Res.* **17**, 857–860.

PORTEUS, E. 1979. An Adjustment to the Norman-White Approach to Approximating Dynamic Programs. *Opns. Res.* **27**, 1203–1208.

ROBERTS, D. 1962. Approximations to Optimal Policies in a Dynamic Inventory Model. In *Studies in Applied Probability and Management Science*, Chap. 13, K. Arrow, S. Karlin and H. Scarf (eds.). Stanford University Press, Stanford, Calif.

ROSS, S. 1970. *Applied Probability Models With Optimization Applications*. Holden-Day, San Francisco.

SCHNEIDER, H. 1978. Methods for Determining the Re-order Point of an (*s*, *S*) Ordering Policy When a Service Level is Specified. *J. Opns. Res. Soc.* **29**, 1181–1194.

SAHIN, I. 1979. On the Stationary Analysis of Continuous Review (*s*, *S*) Inventory Systems With Constant Leadtimes. *Opns. Res.* **27**, 717–729.

SAHIN, I. 1982. On the Objective Function Behavior in (*s*, *S*) Inventory Models. *Opns. Res.* **30**, 709–725.

SAHIN, I. 1983. On Sufficient Conditions for Cost Rate Function Unimodality in Periodic Review (*s*, *S*) Inventory Models. *Opns. Res. Letts.* **2**, 77–79.

SAHIN, I. 1988. Optimality Conditions for Regenerative Inventory Systems Under Batch Demands, *Appl. Stochastic Models and Data Anal.* **4**, 173–183.

SAHIN, I., AND D. SINHA. 1987. On Asymptotic Approximations for (*s*, *S*) Policies. *Stochastic Anal. and Applic.* **5**, 189–212.

SIVAZLIAN, B. 1971. Dimensional and Computational Analysis in (*s*, *S*) Inventory Problems With Gamma Distributed Demand. *Mgmt. Sci.* **17**, B307–B311.

STIDHAM, S. 1977. Cost Models for Stochastic Clearing Systems. *Opns. Res.* **25**, 100–127.

STIDHAM, S. 1986. Clearing Systems and (*s*, *S*) Inventory Systems With Nonlinear Costs and Positive Lead Times. *Opns. Res.* **34**, 276–280.

TIJMS, H. 1986. *Stochastic Modelling and Analysis*. John Wiley & Sons, Chichester, England.

TIJMS, J., AND H. GROENEVELT. 1984. Approximations for (*s*, *S*) Inventory Systems with Stochastic Leadtimes and a Service Level Constraint. *Eur. J. Opnl. Res.* **17**, 175–190.

VEINOTT, A. 1966. On the Optimality of (*s*, *S*) Inventory Policies: New Condition and a New Proof. *J. SIAM Appl. Math.* **14**, 1067–1083.

VEINOTT, A., AND H. WAGNER. 1965. Computing Optimal (*s*, *S*) Inventory Policies. *Mgmt. Sci.* **11**, 525–552.

WAGNER, H. 1975. *Principles of Operations Research*, 2nd ed. Prentice-Hall, Englewood Cliffs, N.J.

WAGNER, H., M. O'HAGAN AND B. LUNDH. 1965. An Empirical Study of Exact and Approximately Optimal Inventory Policies. *Mgmt. Sci.*. **11**, 690–723.

ZIPKIN, P. 1986. Stochastic Leadtimes in Continuous-Time Inventory Models. *Naval Res. Logist. Quart.* **33**, 763–774.

ZIPKIN, P. 1988. Lecture Notes in Inventory Theory. Graduate School of Business, Columbia University, New York.