An Efficient Algorithm for Computing Optimal (s, S) Policies

AWI FEDERGRUEN and PAUL ZIPKIN
Columbia University, New York, New York
(Received February 1982; accepted July 1983)

This paper presents an algorithm to compute an optimal (s, S) policy under standard assumptions (stationary data, well-behaved one-period costs, discrete demand, full backlogging, and the average-cost criterion). The method is iterative, starting with an arbitrary, given (s, S) policy and converging to an optimal policy in a finite number of iterations. Any of the available approximations can thus be used as an initial solution. Each iteration requires only modest computations. Also, a lower bound on the true optimal cost can be computed and used in a termination test. Empirical testing suggests very fast convergence.

MANY PRACTICAL inventory replenishment problems satisfy reasonably closely the mathematical conditions under which (s, S) policies are optimal. Rules of this type, moreover, are easy to implement and require no more data than other standard techniques. Scientific methods for computing the best (or even a "good") policy, however, are rarely used, because—according to folklore—they are prohibitively expensive. We hope the current paper will contribute to the retirement of this myth.

We present here an algorithm to compute an optimal (s, S) policy under standard assumptions (stationary data, well-behaved one-period costs, discrete demand, full backlogging and the long-run-average cost criterion). The overall strategy of the algorithm is policy-iteration, modified to exploit an embedding technique (related to the renewal-theoretic approach) which streamlines many of the computations. The only linear systems that need be solved are all triangular, and hence can be solved by simple substitution. This technique also removes the need for truncation of the state space (as required by standard implementations of policy-iteration and value-iteration, cf. Howard [1960]), so the algorithm is truly exact.

Starting with a given (s, S) policy, the algorithm evaluates a sequence of policies, all of this form (unlike standard policy-iteration), and converges to an optimal one in a finite number of iterations. The policies

Subject classification: 117 dynamic programming, Markov, infinite state, 362 inventory/production stochastic models.

Operations Research
Vol. 32, No. 6, November–December 1984
0030-364X/84/3206-1268 $01.25

generated are strictly improving, but not in the usual sense: Average costs decrease, but not necessarily strictly; cycling is precluded by strict improvement in a certain (natural) lexicographic criterion. In addition, a lower bound on the optimal cost can be computed in every iteration; the algorithm can thus be terminated with a suboptimal policy whose cost achieves any desired level of precision.

A major advantage of such an iterative algorithm is the ability to use any of the available approximations to an optimal policy as an initial solution. These include the method of Roberts [1962] and modifications of it developed by Wagner et al. [1965], Wagner [1975] and Ehrhardt [1979]; and the techniques of Porteus [1979] and Freeland and Porteus [1980a, 1980b], which are based on the general approach of Norman and White [1968]. On the other hand, each of these approximations is accurate only over a limited range of the data of the problem, and the respective ranges are imperfectly known; for any given problem, therefore, one cannot predict how far from optimality the computed policy is. Our algorithm and our bounds thus provide a much-needed corrective to these methods.

It is often asserted (e.g. Ehrhardt [1979]) that methods requiring the full probability distribution of demand are impractical, since they require more information than is usually available. Many of the approximations mentioned above use only the mean and variance of demand. The empirical results of Archibald and Silver [1978], however, indicate that the optimal policy can be sensitive to more than the first two moments of the distribution. Exact methods are the only ones currently known which can use additional information. Fortunately, more information (perhaps qualitative rather than quantitative) often is available; for example, one often knows whether or not the distribution is symmetric. Such information can be used to guide the choice of an appropriate distributional form.

The overall logic of the algorithm is close to that of a method proposed by Johnson [1968], though there are differences in the specifics of the computations. There are problems in Johnson's proof, however. These have been rectified by Küenle and Küenle [1977], under somewhat different assumptions. The Appendix below discusses in detail the models, algorithms and proofs in these two papers in relation to ours. Other methods for computing optimal (s, S) policies include those of Veinott and Wagner [1965], Archibald and Silver, and Bell [1970]; these are essentially enumerative rather than iterative. (Sahin [1982] has recently demonstrated some convexity properties of the average cost as a function of s and S under certain assumptions on the demand distribution. These results permit some of the enumerative algorithms to exploit an advanced starting solution, like our iterative algorithm.)

The embedding technique mentioned above can be seen as a special case of the approach developed by De Leve et al. [1970, 1977a]. The approach has been applied to a variety of systems by Tijms [1976, 1977], De Leve et al. [1977b], Tijms and Van Der Duyn Schouten [1978] and Federgruen et al. [1984]; cf. Tijms [1980] for a survey. Our application, however, is the first for which finite convergence to an optimal policy is proven.

Other work on related inventory systems includes that of Sivazlian [1971], Tijms [1972], Gross and Ince [1975], Naddor [1975], Schneider [1978], and Tijms and Groenevelt [1984].

Computational testing suggests the algorithm converges very quickly. We solved 768 test problems, requiring an average of 0.39 CPU seconds on an IBM 4341. Excluding some unlikely cases (with penalty cost = (1/10) × holding cost), the average drops to 0.09 second. This is certainly fast enough for most applications.

In Section 1 we define notation and discuss certain computations required in the algorithm. The algorithm itself is stated in Section 2, while Section 3 contains the convergence proof and the derivation of the bounds on the minimal cost. Our computational experience is reported in Section 4. An Appendix discusses related algorithms.

1. NOTATION AND PRELIMINARIES

First we define notation; since the problem is stationary we shall suppress time subscripts throughout.

x = inventory at the beginning of a period.
y = inventory position after ordering, but before demand, $y \geq x$.

Both x and y are always integer valued. We assume all stockouts are backordered, so x and y may be negative.

$p_j = \Pr\{\text{one-period demand} = j\}, j = 0, 1, 2, \cdots$.
K = fixed cost to place an order.
$G(y)$ = one-period expected costs, including, e.g., holding and penalty costs, $y = \cdots -1, 0, 1, \cdots$.

We assume only that $p_0 < 1$, $-G$ is unimodal, and $\lim_{|y|\to\infty} G(y) > [\min_y G(y)] + K$. This includes the common case where G 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, cf., e.g., Veinott and Wagner. (An approximate model of the same form can be constructed to handle the case of stochastic delivery lags, cf. Ehrhardt [1984].)

Under these assumptions an (s, S) policy is optimal, as shown by Veinott [1966]. Let $R = (s, S)$ denote a particular policy of this class. We

interpret R to specify orders as follows: If $x \leq s$, set $y = S$ (order $S - x$); otherwise, set $y = x$ (do not order).

In policy-iteration the usual approach to the evaluation of a policy R is based on the resulting stochastic process described by the state variable x, as follows: Define

$\gamma(x, y)$ = one-step total expected cost when x is the initial state, and an order $y - x$ is placed.

$$= \begin{cases} G(y), & y = x \\ G(y) + K, & y > x. \end{cases}$$

$y_R(x)$ = value of y specified by policy R in state x.

(We shall also denote $\gamma_R(x) = \gamma[x, y_R(x)]$.)

g^* = minimal average cost over all policies.

g_R = average cost of policy R.

$\pi_R(x)$ = unique steady state distribution of the inventory x under policy R (cf. Iglehart).

One standard characterization of g_R uses the (infinite) system of equations in the unknowns g and $v(\cdot)$:

$$v(x) = \gamma_R(x) - g + \sum_{j=0}^{\infty} p_j v[y_R(x) - j], \quad \text{for all } x, \tag{1a}$$

$$v(S) = 0 \tag{1b}$$

((1b) serves as a normalization condition.) Now, suppose $\{g, v(\cdot)\}$ solves (1) with $v(x)$ bounded, $x \leq S$. Then, it is not hard to show that $g = g_R$. (Multiply (1a) by $\pi_R(x)$, sum over x and rearrange terms.) In practice, of course, the system (1) must be truncated, and approximate solutions used.

Our approach is based instead on the inventory process observed at epochs *following* replenishment opportunities, that is, on the sequence of y's: Define

$t(v)$ = expected time until the next order is placed when starting v units above the order point, i.e., when $v = y - s$, $v > 0$.

$k_s(y)$ = total expected costs until the next order is placed, when starting with inventory y, $y > s$.

The functions t and k_s, respectively, satisfy the following equations:

$$t(v) = 1 + \sum_{j=0}^{v-1} p_j t(v - j), \qquad v > 0, \tag{2}$$

$$k_s(y) = G(y) + \sum_{j=0}^{y-s-1} p_j k_s(y - j), \qquad y > s. \tag{3}$$

Observe that t is independent of the policy R, and that k_s depends on R only through s. Moreover, systems (2) and (3) are triangular, so t can be computed starting with $v = 1$ by simple substitution, and k_s can be

computed similarly starting with $y = s + 1$. (Note, t is the renewal function of $\{p_j\}$ plus one.)

The key point is that g_R and $v_R(x)$ can be computed from these functions as follows:

$$g_R = [k_s(S) + K]/t(S - s), \tag{4}$$

$$v_R(x) = \begin{cases} [k_s(x) + K] - g_R t(x - s), & x > s \\ K, & x \leq s. \end{cases} \tag{5}$$

((4) is a standard result in the theory of regenerative processes, cf., e.g., Ross [1970], Proposition 5.9. Direct substitution shows that the solution specified by (4), (5) satisfies (1a) and (1b); cf. also Theorem 2.1(a) in Tijms [1980] and Lemma 1 in Federgruen et al.) Thus, no truncation or approximation is necessary.

A policy-iteration algorithm also requires test quantities, in order to search for improved policies; these quantities are defined as follows:

$$I_R(x, y) = \gamma(x, y) - g_R + \textstyle\sum_{j=0}^{\infty} p_j v_R(y - j), \qquad y \geq x. \tag{6}$$

The algorithm uses $I_R(x, y)$ only for certain values of (x, y), and in these cases I_R simplifies greatly: for $y > x$, we have

$$I_R(x, y) = K + G(y) - g_R + \textstyle\sum_{j=0}^{\infty} p_j v_R(y - j). \tag{7}$$

If $y > s$ also, we have $\gamma_R(y) = G(y)$, so using (1), (7) becomes

$$I_R(x, y) = K + v_R(y), \qquad y > x, y > s. \tag{8}$$

Similarly, inserting (5) into (6), we obtain

$$I_R(x, x) = K + G(x) - g_R, \qquad x \leq s. \tag{9}$$

Also, for any two policies R and R', denote $I_R(x, R') = I_R[x, y_{R'}(x)]$.

The algorithm and the convergence proof utilize constants L, M, U, which serve as bounds on the optimal policy, specifically, $L \leq s < M \leq S \leq U$. These may be computed as follows (cf. Veinott):

M = smallest integer minimizing $G(y)$,
U = smallest integer greater than M with $G(U + 1) \geq G(M) + K$,
L = smallest integer such that $G(L + 1) \leq G(M) + K$.

2. STATEMENT OF THE ALGORITHM

Step 0 (Initialize)

Compute the bounds L, M and U as in the previous section. Choose an initial policy $R = (s, S)$ with $L \leq s < M \leq S \leq U$. Compute the function $t(v)$, $v = 1, \cdots, U - L$, using (2).

Step 1 (Value Determination)

If s has changed from the prior iteration, compute $k_s(y)$, $y = s + 1, \cdots, U$, using (3). Compute g_R using (4), and $v_R(x)$, $x = L, \cdots, U$, using (5).

Step 2 (Policy Improvement)

(a) Find S' satisfying $M \leq S' \leq U$ and

$$v_R(S') = \min_{M \leq y \leq U}\{v_R(y)\}. \tag{10}$$

(If S attains the minimum, set $S' = S$.)

(b) Search for an integer y satisfying $s < y < M$ and

$$K + v_R(S') < v_R(x), \qquad s < x \leq y. \tag{11}$$

If such a y exists, set s' to be the *largest* such y, and go to Step 3. Otherwise, search for x satisfying $L \leq x < s$ and

$$G(y) < g_R, \qquad x < y \leq s. \tag{12}$$

If such an x exists, set s' to be the *smallest* such x, and go to Step 3. Otherwise, set $s' = s$.

Step 3 (Test for Termination)

If $R' \equiv (s', S') = R$, stop: R is optimal. Otherwise, set $R = R'$, and return to Step 1.

Remarks. (1) Convergence to the optimum is still guaranteed if Step 2(a) is modified to set S' to be *any* y, $M \leq y \leq U$, such that $v_R(y) < v_R(S) = 0$; if no such value exists, set $S' = S$. For example, we can choose S' as follows: If $\min\{v_R(y)\colon M \leq y \leq S\} < 0$, set S' to achieve the minimum; otherwise set S' to be the smallest *local* minimum of $v_R(y)$, $S \leq y \leq U$, with $v_R(y) < 0$; if none exists, set $S' = S$. This approach reduces the computation required in Step 1: In Step 1 compute $k_s(y)$ and $v_R(y)$ for y up to S, not U. (This much is needed to compute g_R.) Compute these functions for $y > S$ successively, but only when required, in Step 2. Similarly, the new s' can be chosen to be *any* y, satisfying (11) and $s < y < M$, not just the largest, or *any* x satisfying (12) and $L \leq x < s$, not just the smallest.

(2) The policy improvement step does not necessarily select a policy achieving the best one-step improvement (as defined in standard policy-iteration, that is, by choosing y to minimize $I_R(x, y)$ for each x), even when S' is chosen as the global minimum in (10). (Such a policy may fail to have the (s, S) structure.) However, the algorithm can be viewed as performing *restricted* minimizations of I_R: By (8) Step 2(a) chooses y

to minimize $I_R(x, y)$ for $x \leq s$, given $y > x$. Then, Step 2(b) chooses y to minimize $I_R(x, y)$, where y is restricted to $y = x$ or $y > s$, for an interval of values of x containing s; the interval is increased as long as the minimization produces improvements over $I_R(x, R)$.

3. CONVERGENCE PROOF AND BOUNDS

The proof requires several lemmas and two theorems. The bounds are derived afterward.

LEMMA 1. *Let $R' = (s', S')$ be obtained from $R = (s, S)$ by Step 2 of the algorithm. Then,*

$$I_R(x, R') \leq v_R(x), \quad \textit{for all } x. \tag{13}$$

$$g_{R'} \leq g_R. \tag{14}$$

Moreover, (14) *holds strictly if* (13) *is strict for some state x which is positive recurrent under R'.*

Proof. For (13) we consider two cases, (a) and (b):

(a) Suppose $s' < s$. First, for $x \leq s'$, we have

$$I_R(x, R') = K + v_R(S') \leq K + v_R(S) = K = v_R(x),$$

by (4), (5), (8) and (10). Second, for $s' < x \leq s$,

$$I_R(x, R') = G(x) - g_R + K < K = v_R(x),$$

by (5), (9) and (12). Finally, for $x > s$,

$$I_R(x, R') = I_R(x, R) = v_R(x).$$

(b) Suppose $s' \geq s$. For $x \leq s$,

$$I_R(x, R') \leq K = v_R(x),$$

as in case (a). For $s < x \leq s'$,

$$I_R(x, R') = K + v_R(S') \leq v_R(x),$$

by (8) and (11). For $x > s'$, $I_R(x, R') = v_R(x)$ as in case (a), completing the proof of (13).

Now, multiply (13) by $\pi_{R'}$, sum over x, and use (6) to yield $\Sigma_x \pi_{R'}(x) v_R(x) \geq \Sigma_x \pi_{R'}(x) \gamma_{R'}(x) - g_R + \Sigma_x \pi_{R'}(x) \Sigma_j p_j v_R[y_{R'}(x) - j]$. Letting $z = y_{R'}(x) - j$, the double sum (the last term) can be written $\Sigma_x \pi_{R'}(x) \Sigma_z \Pr\{\text{next state} = z \mid x, R'\} v_R(z) = \Sigma_z \pi_{R'}(z) v_R(z)$, using the fact that $\pi_{R'}$ is the equilibrium distribution. But $\Sigma_x \pi_{R'}(x) v_R(x) < \infty$, since $v_R(x)$ is bounded, $x \leq S$, so this term cancels from both sides. Also, $\Sigma \pi_{R'}(x) \gamma_{R'}(x) = g_{R'}$, yielding (14). The assertion concerning strict inequality follows similarly.

(*Remark.* The proof that (13) implies (14) closely follows that of Theorem 2.1(b) in Tijms [1980].)

LEMMA 2. *Let R' be obtained from R by Step 2 of the algorithm. If $R' \neq R$, then either*

(i) $g_{R'} < g_R$;

or (ii) $g_{R'} = g_R$, *but* $v_{R'}(x) \leq v_R(x)$, *for all* x, *and the inequality is strict for some* x.

Proof. Again, there are two cases:

(a) $S' \neq S$: We have $v_R(S') < v_R(S) = 0$ by (10). By (8), therefore, for all $x < M$, $I_R(x, S') < I_R(x, S) = K$, so

$$I_R(x, S') < v_R(x), \qquad x \leq s. \tag{15}$$

If $s' \leq s$, (15) holds in particular for $x \leq s'$; if $s < s'$, (11) ensures that

$$I_R(x, S') = K + v_R(S') < v_R(x), \qquad s < x \leq s',$$

while (15) still holds for $x \leq s$. In either case, we have

$$I_R(x, R') = I_R(x, S') < v_R(x), \qquad x \leq s'. \tag{16}$$

Now, the set $\{x: x \leq s'\}$ clearly includes states that are positive recurrent under R'. Thus, Lemma 1 yields $g_{R'} < g_R$.

(b) $S' = S$ (hence $s' \neq s$): Assume (i) does not hold, so that $g_{R'} = g_R$. We now show by induction that

$$v_{R'}(x) \leq I_R(x, R') \quad \text{for all } x. \tag{17}$$

First, $v_{R'}(x) = K = I_R(x, R')$, $x \leq s'$, by (8). Now suppose $v_{R'}(x) \leq I_R(x, R')$ for all $x \leq \bar{x}$, for some $\bar{x} \geq s'$. From (13) and the definition of I_R,

$$\begin{aligned} v_R(\bar{x} + 1) &\geq I_R(\bar{x} + 1, R') \\ &= G(\bar{x} + 1) - g_R + \textstyle\sum_{j=1}^{\infty} p_j v_R(\bar{x} + 1 - j) + p_0 v_R(\bar{x} + 1). \end{aligned} \tag{18}$$

By repeated substitutions in (18) for the term involving $v_R(\bar{x} + 1)$ we obtain

$$\begin{aligned} v_R(\bar{x} + 1) &\geq I_R(\bar{x} + 1, R') \\ &\geq (1 - p_0)^{-1}[G(\bar{x} + 1) - g_R + \textstyle\sum_{j=1}^{\infty} p_j v_R(\bar{x} + 1 - j)]. \end{aligned}$$

By the induction hypothesis and (13), $v_R(x) \geq v_{R'}(x)$, $x \leq \bar{x}$. Hence, since $g_R = g_{R'}$,

$$\begin{aligned} I_R(\bar{x} + 1, R') &\geq (1 - p_0)^{-1}[G(\bar{x} + 1) - g_{R'} \\ &\qquad + \textstyle\sum_{j=1}^{\infty} p_j v_{R'}(\bar{x} + 1 - j)] = v_{R'}(\bar{x} + 1). \end{aligned}$$

(The equality follows from (1).) This completes the proof of (17).

Now we show that (13) holds strictly for some x (not necessarily a positive recurrent one). If $s' > s$, choose $x = s'$. Then, using (8) and (11),

$$I_R(s', R') = K < v_R(s').$$

If $s' < s$, choose $x = s$. By (9) and (12),

$$I_R(s, R') = I_R(s, s) = K + G(s) - g_R < K = v_R(s),$$

proving the assertion.

This fact, (13) and (17) yield (ii) immediately.

COROLLARY 1. *If $p_1 > 0$ and $R' \neq R$, then $g_{R'} < g_R$.*

Proof. When $p_1 > 0$, x is positive recurrent under R', $s' \leq x \leq S'$. In the proof of Lemma 2 case (b) we showed that (13) holds strictly for $x = s'$ (when $s' > s$) or $x = s$ (when $s' < s$). The result follows from Lemma 1.

THEOREM 1. *The algorithm terminates after a finite number of iterations.*

Proof. Only finitely many policies (s, S) satisfy the bounds

$$L \leq s < M \leq S \leq U,$$

and Lemma 2 ensures that no policy is ever repeated.

Consider now the standard dynamic-programming formulation of the problem; the action set for state x is $\{y: y \geq x\}$. Suppose we use the bounds L, M and U to define a new program with action sets restricted as follows:

$$Y(x) = \begin{cases} \{x\}, & x \geq M, \\ \{y: & y = x \quad or \quad M \leq y \leq U\}, & L < x < M, \\ \{y: & M \leq y \leq U\}, & x \leq L. \end{cases}$$

Also, we may restrict the state space to $X = \{x: x \leq U\}$. Clearly, any (s, S) policy satisfying the bounds, and hence the optimal policy, is feasible for this problem, so the new problem is equivalent to the original one. This construction is crucial in the proof below.

For later use, define Υ, the value-iteration operator in this model, by:

$$\Upsilon v(x) = \min_{y \in Y(x)} \{\gamma(x, y) + \textstyle\sum_{j=0}^{\infty} p_j v(y - j)\}, \qquad x \leq U. \quad (19)$$

Consider now the standard optimality conditions for policy iteration for this equivalent problem:

$$v_R(x) = \min\{I_R(x, y): y \in Y(x)\}, \qquad x \in X. \quad (20)$$

LEMMA 3. *If R satisfies* (20), *then R is optimal.*

Proof. By the remarks above, *some* optimal policy R^* is feasible for the equivalent problem. If R satisfies (20), in particular $v_R(x) \leq I_R(x, R^*)$,

$x \in X$. For $x \notin X$ note $\pi_{R^*}(x) = 0$; as in the proof of Lemma 1, therefore, $g_R \leq g_{R^*}$.

THEOREM 2. *The algorithm terminates with an optimal policy.*

Proof. Let R be the policy to which the algorithm converges. We shall verify that R satisfies (20).

From step 2 of the algorithm we obtain the following relations:

$$v_R(y) \geq v_R(S) = 0, \qquad M \leq y \leq U, \tag{21}$$

$$v_R(s+1) \leq v_R(S) + K = K, \tag{22}$$

$$G(s) \geq g_R. \tag{23}$$

Note first, for any $x < M$, if some $y > x$ achieves the minimum in (20), then $y = S$ does, by (8) and (21). Using the above restriction to the sets X and $Y(x)$, only the following cases need be considered:

(a) For $x \geq M$, $I_R(x, x) = v_R(x)$, by (1).

(b) For $x \leq s$, G is decreasing, so from (23)

$$G(x) \geq G(s) \geq g_R.$$

This together with (8) and (9) yields

$$I_R(x, x) \geq K = I_R(x, S) = v_R(x).$$

(c) For $s < x < M$ we wish to show that

$$I_R(x, x) = v_R(x) \leq K = I_R(x, S). \tag{24}$$

The proof is by induction on x:

First, (22) gives the result for $x = s + 1$. Suppose it is true for $x = s + 1, \cdots, \bar{x} < M - 1$. Then, by (1),

$$v_R(\bar{x}+1) \leq G(\bar{x}+1) - g_R + p_0 v_R(\bar{x}+1) + (1-p_0)K,$$

or

$$v_R(\bar{x}+1) \leq (1/(1-p_0))[G(\bar{x}+1) - g_R] + K. \tag{25}$$

But, from (2)–(4) and (6) we have

$$v_R(s+1) = (1/(1-p_0))[G(s+1) - g_R] + K,$$

so $v_R(s+1) \leq K$ implies $G(s+1) \leq g_R$, hence $G(\bar{x}+1) \leq g_R$. This, with (25), yields the result.

(This proof is related to that of Theorem 3.1 in Van Nunen and Puterman [1983].)

The following theorem shows that bounds on the minimal cost g^* may be computed easily at each iteration. For each policy R, let

$$v_R^+ = \max_{s \leq x < M} v_R(x).$$

THEOREM 3. *Let $R' = (s', S')$ be obtained from $R = (s, S)$ by Step 2 of the algorithm. Then,*

$$g_R + \min\{v_R(S') + K - v_R^+, G(s) - g_R\} \leq g^* \leq g_{R'} \leq g_R. \quad (26)$$

Proof. Only the first inequality needs to be proven. An immediate extension of the bounds in Odoni [1969] and Hastings [1971] (see also Hordijk and Tijms [1974]) results in:

$$\inf_x\{\Upsilon v_R(x) - v_R(x): x \in X\} \leq g^*.$$

To evaluate the infimum, we need only consider $x < M$, since

$$\Upsilon v_R(x) - v_R(x) = g_R, \qquad x \geq M.$$

If $s' < s$,

$$\Upsilon v_R(x) - v_R(x) = \begin{cases} v_R(S') + g_R, & x \leq s' \\ \min\{G(x), v_R(S') + g_R\}, & s' < x \leq s \\ \min\{g_R + v_R(S') + K - v_R(x), g_R\}, & s < x < M, \end{cases}$$

while, if $s' \geq s$,

$$\Upsilon v_R(x) - v_R(x) = \begin{cases} \min\{G(x), v_R(S') + g_R\}, & x \leq s \\ g_R + v_R(S') + K - v_R(x), & s < x \leq s' \\ \min\{g_R + v_R(S') + K - v_R(x), g_R\}, & s' < x < M. \end{cases}$$

The lower bound in (26) follows immediately from the definition of v_R^+ and the fact that $G(y)$ is decreasing, $L \leq y \leq M$.

Remarks. (1) The results of Odoni and of Hastings lead to a similar upper bound $\sup_x \{\Upsilon v_R(x) - v_R(x)\}$. The proof above shows, however, that this bound is never tighter than g_R.

(2) Upon termination of the algorithm, the gap between the lower and upper bounds in (26) vanishes; this follows from (23) and (24).

(3) The bounds in (26) may be used to terminate the algorithm prior to optimality when a desired tolerance level is met. For this purpose let δ_R denote the difference between the upper and lower bounds in (26), that is,

$$\delta_R = \max\{g_R - G(s), v_R^+ - K - v_R(S')\}.$$

Suppose we want an ϵ-optimal policy, i.e., a policy R with $g_R \leq g^* + \epsilon$ for some $\epsilon > 0$. Then, replace Step 3 in the algorithm by the following:

Step 3′ (Test for Termination)

Compute δ_R. If $\delta_R \leq \epsilon$, stop: R is ϵ-optimal.

TABLE I

PARAMETER SETTINGS

Parameter	No. Values	Values
Mean	4	2, 6, 20, 60
Variance/mean	4	0.33, 0.75, 1.5, 10
Lead time + 1	3	1, 5, 25,
Fixed cost	4	0.1, 1, 10, 100
Penalty cost	4	0.1, 1, 10, 100

Otherwise, set $R = R'$ and return to Step 1.

Alternatively, suppose we want a policy within a certain fraction of the optimal cost, i.e., such that $(g_R - g^*)/g^* \leq \alpha$ for some $\alpha > 0$. Then replace the test in Step 3′ above with the condition $\delta_R/(g_R - \delta_R) \leq \alpha$. (In either case we may wish to use R' instead of R as our final policy; if so, we may also wish to compute $g_{R'}$ using (3) and (4) before exiting.)

(4) Note that v_R^+ can be determined easily in Step 1 while computing $v_R(\cdot)$.

4. COMPUTATIONAL EXPERIENCE

To test the algorithm we solved 768 representative problems, each having discretized normal demands. The one-period cost functions G were based on linear holding and penalty costs and, in some cases, a fixed leadtime. Every combination of the parameters shown in Table I was used with holding cost = 1. (These parameter settings were suggested to us by Evan Porteus.) The initial policy in each case was determined by the approximation in Ehrhardt [1979].

The results are shown in Tables II–VI. Each table examines the effect of a single parameter. Table II, for example, shows averages over the other parameters for each value of mean demand; Tables III–VI are organized similarly. Each change in policy is counted as an iteration; when the initial policy is optimal, zero iterations are reported. The times reported are virtual seconds on an IBM 4341 operating under VM/CMS,

TABLE II

EFFECT OF MEAN ON PERFORMANCE

Mean	Cases	Iterations	CPU (sec)	Approximation % Over Optimal
2	192	1.97	0.27	14.81
6	192	1.84	0.30	14.54
20	192	1.84	0.39	17.63
60	192	1.68	0.59	18.82
Total	768	1.83	0.39	16.45

TABLE III

EFFECT OF VARIANCE/MEAN RATIO ON PERFORMANCE

Var/Mean	Cases	Iterations	CPU (sec)	Approximation % Over Optimal
0.33	192	1.75	0.37	23.65
0.75	192	1.69	0.37	18.25
1.50	192	1.75	0.38	14.39
10.00	192	2.15	0.43	9.51
Total	768	1.83	0.39	16.45

TABLE IV

EFFECT OF LEAD TIME ON PERFORMANCE

Lead + 1	Cases	Iterations	CPU (sec)	Approximation % Over Optimal
1	256	1.65	0.36	21.51
5	256	1.80	0.38	16.25
25	256	2.05	0.43	11.59
Total	768	1.83	0.39	16.45

TABLE V

EFFECT OF FIXED COST ON PERFORMANCE

Fixed Cost	Cases	Iterations	CPU (sec)	Approximation % Over Optimal
0.1	192	0.88	0.02	1.15
1	192	1.46	0.03	10.83
10	192	2.21	0.08	27.05
100	192	2.79	1.42	26.77
Total	768	1.83	0.39	16.45

TABLE VI

EFFECT OF PENALTY COST ON PERFORMANCE

Penalty Cost	Cases	Iterations	CPU (sec)	Approximation % Over Optimal
0.1	192	3.32	1.28	50.71
1	192	1.71	0.14	7.52
10	192	1.07	0.07	3.55
100	192	1.24	0.06	4.03
Total	768	1.83	0.39	16.45

and include computation of probabilities, the bounds L, M and U, the initial policies and the lower bounds (26), as well as the iterations themselves. The enhancement suggested in Remark (1) at the end of Section 2 was not used. The right-most column of each table compares the costs of the initial policy and the optimal policy. For each problem the figure 100 × (initial cost − optimal cost)/(optimal cost) was computed; the tables give averages of these figures.

Of all the parameters the penalty cost and the fixed cost evidently have the greatest effects on the performance of the algorithm: Smaller penalty costs and larger fixed costs require more computational effort, in part because the initial solution is less accurate in these cases, but also because each iteration is more expensive (the range U–L is larger). The other parameters have substantially smaller effects.

Overall, the computational demands of the algorithm seem to us quite reasonable. Note that a penalty cost of only 10% of the holding cost is rather unlikely in practice, and the other three cases average 0.09 CPU second per problem. Using this figure (as a quick calculation shows) an optimal policy for each of 10,000 items can be computed in 15 minutes of computer time. This, we believe, is a quite modest requirement.

APPENDIX: RELATED ALGORITHMS

Models

Call ours the "standard" model. Johnson permits a more general cost structure and dynamics, and we shall refer to his as the "generalized" model. Küenle and Küenle treat the standard model only, though with a slightly more general function $G(\cdot)$, and assuming $p_1 > 0$.

We describe the algorithms and results of these papers only as applied to the standard model, using our notation. In particular, the policy parameter s has a different meaning in these papers (order when inventory is strictly less than s), so we use $s + 1$ below when they write s.

Algorithms

Johnson considers several algorithms. Here, we shall discuss the two which seem most closely related to ours, which he calls "(*H*)" and "1." Küenle and Küenle consider only (*H*).

Algorithm (*H*) differs from ours in the following ways:

1. Only one coordinate of R is changed at a time. That is, in our Step 2(a), if $S' \neq S$, return to Step 1.
2. In Step 2(b), s is restricted to change by at most one; that is, either $s' = s$ or $s' = s \pm 1$. The test for increasing s can be shown equivalent to our (11), when $S' = S$ and $x = s + 1$. In this case (11) reduces to

$G(s + 1) > g_R$, the test used by Küenle and Küenle. The test for increasing s (p. 86 of Johnson) is $G(s) < g_R$, as in our (12).

Algorithm (H) is intended not as a computational tool itself, but rather as a means to other results. Algorithm 1 differs from ours in different ways:

1. Instead of specifying a particular value of s in each iteration, the algorithm determines an interval $[s_1, s^1]$ within which some value of s must be chosen.
2. Step 2(a) chooses S' to minimize $g_{(s,y)}$ over y, instead of $v_R(y)$.
3. The revision of $[s_1, s^1]$ is accomplished as follows:
   If $G(s + 1) > g_{(s,S')}$,

   set s_1 to be the smallest integer y such that $G(y + 1) < g_{(s,S')}$.

   If $G(s) < g_{(s,S')}$,

   set $s^1 = s - 1$, and s_1 to be the smallest x with $G(x + 1) < g(s, S')$.

We now discuss the similarities between our algorithm and Algorithm 1. In view of Remark (1) at the end of our Section 2, point (2) above is not essential. The work required to calculate $g_{(s,y)}$ is about the same as that for $v_R(y)$, and $g_{(s,y)} < g_R$ is equivalent to $v_R(y) < 0$. (This does not mean that S' is chosen identically in the two algorithms, of course.) Also, when $S' = S$ and $G(s) < g_R$ (so (12) is satisfied for $y = s$) our algorithm sets $s' = s_1$; when $S' \neq S$, however, $s' \leq s_1$, since $g_R \geq g(s, S')$. Finally, suppose (11) holds for some $x > s$. Using the definitions of v_R, k_s and t, there are positive constant $\alpha_0, \cdots, \alpha_{x-s-1}$ such that (11) is equivalent to

$$\sum_{i=0}^{x-s-1} \alpha_i[G(x - i) - g_R] > v_R(S').$$

In the case $S' = S$, therefore, if $s_1 \geq x$ (so each $G(x - i) - g_R \geq 0$ with the inequality strict for $x - i = s + 1$) then also $s' \geq x$, and thus $s' \geq s_1$. When $S' \neq S$, there appears to be no obvious relation between s' and s_1.

The differences between the algorithms can be summarized as follows: Ours provides a specific choice of s which works extremely well. The test quantities and policy improvements in our algorithm are chosen to remain as close as possible to the spirit of policy iteration, while restricting attention to (s, S) policies. Although there are apparent similarities to algorithm 1 in mechanics, the differences are significant, particularly when $s' > s$.

Proofs

The proof in Johnson that Algorithm (H) converges to an optimal policy presents several difficulties:

1. Results for finite-state dynamic programs are invoked, while the standard model has a countably infinite state space. To use these

results the state space must be truncated *a priori* to a finite set; also, to ensure that the state never leaves this set, the actions must be suitably restricted, and the demand distribution must be truncated in a manner depending on the current inventory level. (The discussion on p. 82 of Johnson could be interpreted as suggesting that such a truncation may be derived from subsequent results; this is not the case. This point is mentioned by Küenle and Küenle. Problems with infinite state spaces can be quite ill-behaved, cf. Derman [1966], Ross, and Federgruen and Tijms [1978].)

2. The core of the argument is the assertion that the policy strictly improves at each iteration. This is never substantiated.

The proof of Küenle and Küenle requires only that the state space be bounded above (so, in particular, the demand distribution need not be truncated). Assuming $p_1 > 0$, they show that g_R improves strictly in each iteration.

The situation can thus be summarized as follows: Assuming $p_1 > 0$ and the standard model with inventory levels bounded above, Algorithm (H) is correct. Indeed, it does improve at every iteration. With this fact one can construct a valid justification for Algorithm 1, along the lines suggested by Johnson. Even for the standard model, ours is the only proven algorithm of this kind when $p_1 = 0$ and for an unrestricted state space.

ACKNOWLEDGMENTS

This research was supported in part by the Faculty Research Fund, Graduate School of Business, Columbia University. The authors wish to thank L. Pohlman for help with the computation. The referees and the associate editor provided many valuable suggestions.

REFERENCES

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

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

DE LEVE, G., H. TIJMS, AND P. WEEDA. 1970. Generalized Markovian Decision Processes, Applications, Tract No. 5. Mathematisch Centrum, Amsterdam.

DE LEVE, G., A. FEDERGRUEN, AND H. TIJMS. 1977a. A General Markov Decision Method; I. Model and Techniques. *Adv. Appl. Prob.* **9,** 297–315.

DE LEVE, G., A. FEDERGRUEN, AND H. TIJMS. 1977b. A General Markov Decision Method; II. Applications. *Adv. Appl. Prob.* **9,** 316–335.

DERMAN, C. 1966. Denumerable State Markovian Decision Processes—Average Cost Criterion. *Ann. Math. Statist.* **37,** 1545–1553.

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 H. TIJMS. 1978. The Optimality Equation in Average Cost Denumerable State Semi-Markov Decision Processes, Recurring Conditions and Algorithms. *J. Appl. Prob.* **15,** 356–372.

FEDERGRUEN, A., H. GROENEVELT, AND H. TIJMS. 1984. Coordinated Replenishments in a Multi-Item Inventory System with Compound Poisson Demands and Constant Lead Times. *Mgmt. Sci.* **30,** 344–357.

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.

GROSS, D., AND R. INCE. 1975. A Comparison and Evaluation of Approximate Continuous Review Inventory Models. *Int. J. Product. Res.* **13,** 9–23.

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

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

HOWARD, R. 1960. *Dynamic Programming and Markov Processes.* Wiley, New York.

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.

KÜENLE, C., AND H. KÜENLE. 1977. Durchschnittsoptimale Strategien in Markovschen Entscheidungsmodellen bei Unbeschränkten Kosten, *Math. Operationsforsch. Statist., Ser. Optimization* **8,** 549–564.

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

NORMAN, J., AND D. WHITE. 1968. A Method for Approximate Solutions to Stochastic Dynamic Programming Problems Using Expectations. *Opns. Res.* **16,** 296–306.

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. Opnl. Res. Soc.* **29,** 1181–1194.

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

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

TIJMS, H. 1972. Analysis of (s, S) Inventory Models, Tract No. 40. Mathematisch Centrum, Amsterdam.

TIJMS, H. 1976. Optimal Control of the Workload in an $M/G/1$ Queuing System with Removable Server. *Math. Operationsforsch. Statist.* **7,** 933–943.

TIJMS, H. 1977. On a Switch-over Policy for Controlling the Workload in a Queuing System with Two Constant Service Rates and Fixed Switch-over Costs. *Z. Opns. Res.* **21,** 19–32.

TIJMS, H. 1980. An Algorithm for Average-Cost, Denumerable-State Semi-Markov Decision Problems with Applications to Controlled-Production and Queuing Systems. In *Recent Developments in Markov Decisions Processes*, pp. 143–179, R. Hartley, L. Thomas and D. White (eds.). Academic Press, London.

TIJMS, H., AND H. GROENEVELT. 1984. Approximations for (s, S) Inventory Systems with Stochastic Lead Times and a Service Level Constraint. *Eur. J. Opns. Res.* (to appear).

TIJMS, H., AND F. VAN DER DUYN SCHOUTEN. 1978. Inventory Control with Two Switch-over Levels for a Class of $M/G/1$ Queuing Systems with Variable Arrival and Service Rates. *Stoch. Proc. Appl.* **6,** 213–222.

VAN NUNEN, J., AND M. PUTERMAN. 1983. Computing Optimal Control Limits for $GI/M/s$ Queuing Systems with Controlled Arrivals. *Mgmt. Sci.* **29,** 725–734.

VEINOTT, A. 1966. On the Optimality of (s, S) Inventory Policies: New Conditions 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*, Ed. 2. Prentice-Hall, Englewood Cliffs, N.J.

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