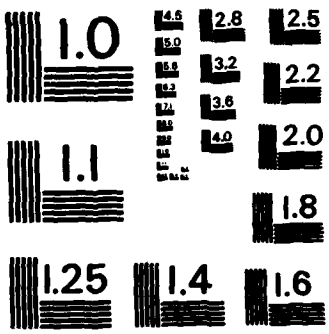
AD-A159 950 A METHODOLOGY FOR COMPUTATION REDUCTION FOR SPECIALLY STRUCTURED LARGE SC..(U) NORTH CAROLINA STATE UNIV AT RALEIGH DEPT OF INDUSTRIAL ENGIN.. F Y DING ET AL. 1/1

UNCLASSIFIED JAN 85 NCSU-IE-TR-85-3 N00014-84-K-0046 F/G 12/1 NL

END

FILMED

DTIC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

AD-A159 950

A Methodology for
Computation Reduction for Specially Structured
Large Scale Markov Decision
Problems

NCSU-IE Technical Report #85-3

DEPARTMENT OF

INDUSTRIAL ENGINEERING

DTIC FILE COPY

DTIC
ELECTE
OCT 2 1985
S
A

has been approved
sale; its
unlimited

North Carolina State University
Raleigh, N. C.

85 8 21 207

A Methodology for Computation Reduction for Specially Structured Large Scale Markov Decision Problems

NCSU-IE Technical Report #85-3

Fong-Yuen Ding*
Thom J. Hodgson*
Russell E. King**

DTIC
ELECTE
OCT 2 1985
S D
A

January, 1985

* Industrial Engineering Department
North Carolina State University
Raleigh, North Carolina 27695-7906

** Department of Industrial & Systems Engineering
University of Florida
Gainesville, Florida 32611

This research was supported in part by the Office of Naval Research, under contract number N00014-84-K-0046.

approved
sale; its
unlimited.

ABSTRACT

Markov Decision Processes deal with sequential decision making in stochastic systems. Existing solution techniques provide powerful tools for determining the optimal policy set in such systems, however, many practical problems have extremely large state and action spaces making them computationally intractable. Typically, the state variable definition is n-dimensional and the number of states expands at a rate proportional to the power of n. For such large problems, the need for large amounts of random access memory and computation time restricts the ability to obtain solutions. The purpose of this paper is to both present a methodology which facilitates the solution of large scale problems, and provide computational results indicating the value of the approach.

Additional Keywords: table (data); convergence.

DTIC COPY INSPECTED 3

INTRODUCTION

An undiscounted, infinite horizon, discrete time Markov Decision Process problem can be described as follows: A system has N states and for each state i the decision maker can select any alternative from set K_i. If the system is in state i and alternative k is selected, the system will make a transition to state j, j=1,...,N, according to the given transition probability vector $[p_{i1}^k, p_{i2}^k, \ldots, p_{iN}^k]$, and earn a reward (cost) r_{ij}^k.

A policy is defined as a collection of selected alternatives for all states. $v_i(n)$ denotes the total expected earnings over the next n transitions if the system is now at state i and the optimal policy is followed.

The Dynamic Programming formulation developed by Howard [5] for the finite horizon problem gives the following recursive equations :

$$v_i(n+1) = \max_{k \in K_i} \left\{ \sum_{j=1}^{N} p_{ij}^k (r_{ij}^k + v_j(n)) \right\}$$

$$= \max_{k \in K_i} \left\{ q_i^k + \sum_{j=1}^{N} p_{ij}^k v_j(n) \right\}, \quad (i=1,\ldots,N),$$

$$\text{where} \quad q_i^k = \sum_{j=1}^{N} p_{ij}^k r_{ij}^k.$$

Since the infinite horizon process is allowed to make infinitely many transitions, the total expected earnings is eventually infinite. The goal of infinite horizon Markov Decision Process optimization is thus to find the policy that maximizes (minimizes) the expected gain (g) per transition.

$$g = \sum_{i=1}^{N} \pi_i q_i,$$

where π_i is the limiting state probability for state i.

Howard's Policy Iteration method [5] can be applied to infinite horizon problems. In this procedure, relative values, v_i's, are determined for a given policy by solving a set of N simultaneous linear equtions. These relative values are used in a policy improvement procedure to find a policy with a higher (lower) gain. The new policy is used to determine a new set of relative values v_i. This process is repeated until no better policy can be found. The disadvantage of Howard's Policy Iteration for large scale problems is the computational effort required to solve the N simultaneous linear equations.

A successive approximation approach for solving the N equations can be shown empirically to be computationally more efficient. The successive approximation method can be stated as follows :

If all transition probability matrices (of all possible policies) are single chained, and all those associated with

maximal gain policies are aperiodic, the following recursive computation will eventually converge to the optimal policy [12] :

$$y_i(n+1) = \underset{k \in K_i}{\mathrm{MAX}}\{q_i^k + \sum_{j=1}^{N} p_{ij}^k w_j(n)\},$$

$$w_i(n+1) = y_i(n+1) - y_N(n+1),$$

starting with $w_i(0) = y_i(0) - y_N(0)$.

It also can be shown that $y_N(n)$ converges to the optimal gain and the $w_i(n)$'s converge to the relative values as in Howard's method.

Morton [9] showed that a fixed policy successive approximation guarantees the convergence of the relative values (i.e.,$w_i(n)$'s) in on the order of $1/(1-\beta)$ iterations, where β is the second largest eigenvalue of the transition matrix of this fixed policy. He suggested a method similar to Howard's Policy Iteration procedure, except that the relative values are computed by the fixed policy successive approximation. In this paper, we modify Morton's approach in order to gain computational efficiency.

OTHER RELEVANT LITERATURE

We will limit discussion to the undiscounted, discrete time Markov Decision Process problem. Bellman [1] first proposed the Markov Decision Process problem. Howard [5]

presented the Policy Iteration approach, which later was generalized to other classes of Markov Decision Processes. His work also gave a detailed discussion of the modeling concepts of Markov Decision Processes.

White [17] first proved the asymptotic behavior of successive approximations under the condition that there exists a path of u step transitions (for all possible policies) connected to some state from all other states, where u is a positive integer. Schweitzer [12] later showed that if all transition probability matrices are single chained and aperiodic, then the successive approximation for Howard's Dynamic Programming approach [5] results in the asymptotic convergence of the total expected earnings ($v_i(n)$) to a linear function.

In addition, alternate Linear Programming formulations were presented by Manne [7], Wolfe and Dantzig [18], and Wagner [16]. It was shown that the optimal solutions are always pure policies. It might also be noted that for a Markov Decision Process, Howard's Policy Iteration is equivalent to the block pivoting approach of Linear Programming.

Odoni [10] presented upper and lower bounds for the gain and relative state costs using the successive approximation approach, thus establishing rational stopping criteria for the procedure. A more general condition of the asymptotic behavior of successive approximation for the multichain case was given by Schweitzer and Federgruen [13].

The generalizations of Policy Iteration and successive

approximation to periodic Markov Decision Processes have been presented by Peterson [11] and Su and Deninger [15]. Zaldivar and Hodgson [19] proposed an extrapolation procedure for speeding convergence of the relative values. Hodgson and Koehler [4] investigated scaling techniques to speed convergence in Markov, semi-Markov and continuous parameter Decision Processes. Schweitzer and Seidmann [14] suggested a method of polynomial approximations for the relative values by noticing that the relative values could be fit accurately on the state space.

OBSERVATIONS

Three observations relative to real-world large scale Markov Decision Process problems can be made.

(1) We have examined many production-inventory problems formulated as Markov Decision Processes. Typically, a large portion of states are transient for an optimal policy set. Since recurrent states form a closed communicating class, then by the fixed policy successive approximation, the alternatives at transient states do not influence the calculation of the relative values of the recurrent states or the gain of the system.

(2) The recurrent states of the optimal policy tend to cluster in a small number (possibly 1) of compact groups.

(3) The state space is vector valued, i.e., $\mathbf{i}=(i_1,i_2,\ldots,i_s)$ and the alternatives can also be described as vector valued, i.e., $\mathbf{k}=(k_1,k_2,\ldots,k_m)$. Note that the

dimension (s) of the state variable could be different from the dimension (m) of the alternative vector. An interpretation of the state variable **i** might be the amount of inventory of product at each stage of production. An interpretation of the alternative vector **k** might be the amount of product ordered at each stage of production. It should be noted that in the following state designations **i** and **j** refer, when appropriate, to the vector state designation and **k** refers to the vector alternative designation.

BASIC CONCEPTS FOR THE PROCEDURE

(1) I-step transient states

For a given transition matrix, let R denote the set of recurrent states and T denote the set of transient states. We know that if $i \varepsilon R$, and $j \varepsilon T$, then $p_{ij} = 0$, i.e., there is no transition from a recurrent state to a transient state.

Note that the fixed policy successive approximation is as follows,

$$y_i(n+1) = q_i + \sum_{j=1}^{N} p_{ij} w_j(n), \qquad (1)$$

$$w_i(n+1) = y_i(n+1) - y_N(n+1). \quad (i=1,..,N)$$

For a state $i \varepsilon R$, $p_{ij} > 0$ only for $j \varepsilon R$. Thus it is possible to ignore the computation of $w_j(n)$ and $y_j(n)$ for transient

states and limit the range of the summation in (1) to the recurrent states without altering the result of the computation for the recurrent states.

In addition, many of the transient states can be classified as "one-step transient states", ..., "I-step transient states", etc. A state is said to be a "I-step transient state" if, upon starting in this state, the system will reside in a recurrent state after exactly I-transitions with probability = 1. Clearly, within one step, a two-step transient state can only reach recurrent states and one-step transient states.

In computing equation (1), if i is a one-step transient state, the relative value w_i can be calculated exactly with one iteration after convergence of the relative values of the recurrent states is achieved. As for a two-step transient state, it is necessary only to calculate its relative value for two iterations after the convergence of recurrent states is achieved.

(2) Neighboring states

The neighboring states to state i are defined to be those states that are within a radius r of the state i in the s-dimensional state space. For example, a two dimensional state (i_1,i_2) has the following neighboring states within a radius of 1 : (i_1+1,i_2), (i_1,i_2+1), (i_1-1,i_2), (i_1,i_2-1).

It should be noted that within the concept of Markov Processes, it is useful to think of the neighborhood of a set of states (e.g., the set of recurrent states). Since there

would normally be considerable overlap in the identification of neighboring states, the neighborhood of the recurrent set of a Markov Process might contain fewer states than are in the recurrent set itself.

The concept of neighboring states will help restrict computation to a limited set of states. Consequently, it will limit both the amount of computation and active memory required.

A PROCEDURE

We now state the procedure. The conditions under which the procedure is guaranteed to achieve the optimal policy set will be given in the next section.

1. For a given policy, compute the limiting state probabilities in order to determine the set of recurrent states, R.

2. Find neighboring states within some radius r for all recurrent states. Let the set of all the neighboring states not in R be A.

3. Find all the states that are reachable (in one or more transitions) from A but not including the states in A or R. Let the set of these states be C.

4. Implement the fixed policy successive approximation for the states in the set R+A+C for a fixed number of iterations.

5. Making use of the relative values w_i calculated in step 4 for states in R+A+C, implement the policy improvement:

$$\max_{k \in K_i'} \{q_i^k + \sum_j p_{ij}^k w_j(n)\}, \quad i,j \in R+A+C,$$

where K_i' denotes the set of alternatives for state i that can only make transition to states in R+A+C.

6. If there is no change in the policy set and the $w_i(n)$'s have "converged", stop. Otherwise, go to step 1.

The purpose of determining set C in step 3 is to find all the states that are in the paths from neighboring states to recurrent states. This is relatively easy computationally because of the structure of the state space in real problems, i.e., a "neighboring" state should be very "close" to the recurrent states in the state space. Computational experience has shown that C is typically a very small set.

In step 4, the fixed policy successive approximation is restricted to the set R+A+C. Since all the states in A+C have a path to R (note that many states in A+C are typically I-step transient with small I), fixed policy successive approximation guarantees the convergence of the relative values.

Step 5 restricts the policy improvement to the set of alternatives that communicate within R+A+C. If an alternative communicates with a state j outside R+A+C, it is not possible to compare this alternative with other alternatives because the relative value ($w_j(n)$) is unknown. Note that the new policy generated by this policy improvement procedure restricts all states in R+A+C to communicate only with the states in R+A+C. In other words, the new recurrent chain, R',

contains only states in R+A+C. With this new recurrent chain, we have a new set R'+A'+C' which could be used to find new relative values and, consequently, a new policy set.

Notice that step 1 is actually efficient in computation. Initially a single state is assigned a state probability of 1 and then the state probabilities are computed recursively by using $\pi(n+1)=\pi(n)P$, where $\pi(n)$ is the state probability vector after n transitions and P is the transition probability matrix. In real Markov Decision problems $\pi(n)$ is usually a sparse vector, the actual computations at each iteration would only include these states that have positive entries of $\pi(n)$ as well as reachable states from these states. Given that the underlining Markov process is single-chained, it will eventually converge to its limiting state probabilities and thus find the set of recurrent states. In the actual computations, we would also restrict the number of iterations to a predeterminated number (say, 10). Even though some states that are actually recurrent may not be included in the recurrent set within the given number of iterations, if there is at least one recurrent state in this "incomplete" recurrent set, step 3 of the procedure will find all the other recurrent states.

The savings in computation of the procedure directly relate to the scheme that only a part of states (R+A+C) are actually involved in the computations at each iteration.

CONDITIONS FOR CONVERGENCE OF THE PROCEDURE

First, the definition of a concave (convex) function in the discrete space is introduced.

Definition Let Ω be an m-dimensional discrete state space and $(k_1,k_2,\ldots,k_m) \in \Omega$, where the k_i are integers. A function $f(k_1,k_2,\ldots,k_m)$ is said to be concave (convex) if for any vectors $\mathbf{k_a},\mathbf{k_b},\mathbf{k_c} \in \Omega$, where $\mathbf{k_c} = \lambda\mathbf{k_a}+(1-\lambda)\mathbf{k_b}$ for some $\lambda \in (0,1)$,

we have $\lambda f(\mathbf{k_a})+(1-\lambda)f(\mathbf{k_b}) \leq f(\mathbf{k_c})$

$$(\lambda f(\mathbf{k_a})+(1-\lambda)f(\mathbf{k_b}) \geq f(\mathbf{k_c})).$$

In general, assume that there exits an s-dimensional state variable $(i_1,i_2,\ldots,i_s)$ and an m-dimensional alternative variable $(k_1,k_2,\ldots,k_m)$ at each state. Assume also the objective is to maximize the gain of the system. The procedure will find the optimal solution if the system satisfies the following two conditions :

1. Consider an alternative variable $(k_1,k_2,\ldots,k_m)$ that brings the system to a set of states, say S. The neighborhood chosen to implement the procedure must satisy the following: If we increase (or decrease) any entry of this alternative variable by 1 unit, for example, $(k_1,k_2,\ldots,k_i+1,\ldots,k_m)$, the new alternative will bring the system in one transition to a set of states that is included in the states in S plus the neighborhood of S.

2. For each state i, the test quantity

$q_i(k_1,k_2,\ldots,k_m)+\sum_j p_{ij}(k_1,k_2,\ldots,k_m)w_j$ is a concave function in $(k_1,k_2,\ldots,k_m)$ (The test quantity needs to be convex if the objective is to minimize the gain).

If the above two conditions are true, then whenever the current policy is not optimal for a given state in the current set R, there must exist some better alternative (i.e., larger (smaller) test quantity) that keeps the system within the current R+A+C set. This alternative would be found by the modified policy iteration. This implies that under conditions 1 and 2, if the policy set for states in R is not optimal, then a new (better) policy set can be found. This procedure can be repeated eventually terminating with the optimal policy set. The problem is that while many models satisfy condition 1, few models will fully satisfy condition 2. However, in the following section, it will be seen that, empirically at least, this approach to computation is both robust and extremely efficient.

A TEST PROBLEM

In this section a test problem is described. The problem does not fully satisfy condition 2. However, we will see in the following section (**COMPUTATIONAL RESULTS**) the robustness of the procedure in solving the problem. A multistage production-inventory system has been used as a test problem for our methodology. Clark and Scarf [3] formulated this problem and proposed a heuristic algorithm to find control rules for the system. Lambrecht, Luyten and Muckstadt [6]

formulated it as a Markov Decision Process and offered an interesting computation comparison of these two approaches.

Consider a two-stage production-inventory system which operates on a period to period basis. (See FIGURE 1 for a schematic of the system.) At the beginning of each period, production at each stage must be determined. There is an in-process and a finished product inventory. The units of inventory and production are integers. There is a one period delay for production at the second stage. The system incurs set up and variable costs of production, inventory costs and shortage costs. The maximum inventory level at each facility is restricted. The demand for a period is expressed as a probability mass function. This is an infinite horizon, undiscounted, discrete time Markov Decision Process problem. The objective is to minimize the total cost rate of the system.

The following terminology is useful in describing the model:

1. State variable (i_1,i_2) : i_1 is a nonnegative integer value representing the on-hand inventory level of stage 1, i_2 is a nonnegative integer representing the on-hand plus on-order inventory of stage 2. Let $\mathbf{i}=(i_1,i_2)$.

2. Alternative variable (k_1,k_2) : k_1 is a nonnegative integer representing the number of units to be produced at stage 1, k_2 is a nonnegative integer representing the number of units that to be produced at stage 2. Let $\mathbf{k}=(k_1,k_2)$.

3. Demand D : a random variable which can take on the values 0, 1, 2, ..., depicting the units of finished product

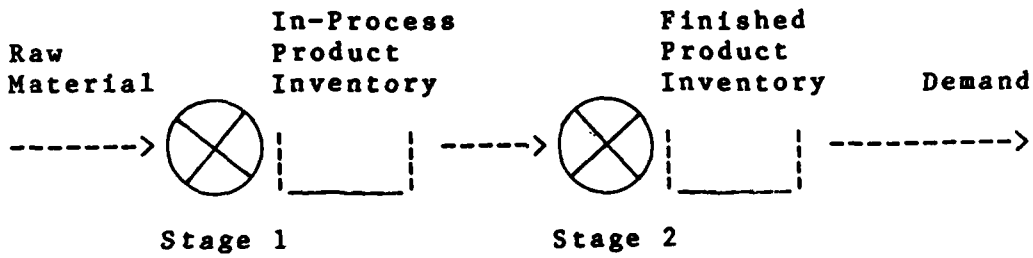

Raw
Material
In-Process
Product
Inventory
Finished
Product
Inventory
Demand
Stage 1
Stage 2

FIGURE 1

ordered according to probability mass function, p(d).

4. Maximum inventory L1 and L2 : L1 is the maximum inventory of stage 1 and L2 is the maximum inventory of stage 2.

The state variables are defined to be the integer pairs (i_1,i_2), with $i_1 \leq L1$ and $i_2 \leq L2$. If the the system is now at state (i_1,i_2) and alternative (k_1,k_2) is chosen, the transition is

$$(i_1,i_2) \text{-------}> (i_1+k_1-k_2,(i_2-D)^+ +k_2), \tag{2}$$

where $(x)^+ = \max(0,x)$

To assure that the inventory remains at or below the maximum inventory level, the sum of on hand and on order inventory is limited to be less than or equal to L2 (the maximum inventory level at stage 2), i.e., $k_2+i_2 \leq L2$.

It can be shown that if the current alternative at some state is (k_1,k_2) and the state communicates directly to a set of states, say R, then an increase (or decrease) of one unit of k_1 or k_2 will bring the system to a set of states that is in R+A, where A is the neighboring set of radius $r= \sqrt{2}$. Notice that this is equivalent to condition 1 of the procedure.

From our computational experience for this model, the relative costs w_i's normally form a convex function in (i_1,i_2) (A similar observation was given by Schweitzer and Seidmann [14]). Making use of this observation and the

state transitions described in (2), it can be shown that $\sum_j p_{ij}^k w_j$ is normally a convex function in (k_1,k_2).

The cost of a transition given a demand d is

$$r_{i,j(d)}^k = HC1*(i_1+i_2)+HC2*i_2+$$
$$VC1*(k_1)+VC2*(k_2)+$$
$$\min(1,k_1)*SC1+\min(1,k_2)*SC2+$$
$$SHC*(i_2-d)^-,$$

where HC1 and HC2 are the echelon inventory holding costs of stage 1 and 2, VC1 and VC2 are the variable costs of stage 1 and 2, SC1 and SC2 are the setup costs of stage 1 and stage 2, SHC is the shortage cost, and $(x)^-=-\min(0,x)$. Note that j is determined by d.

It is easy to see that $r_{i,j(d)}^k$ is a convex function in (k_1,k_2) for $k_1 \geq 1$ and $k_2 \geq 1$. Thus $q_i^k = \sum_d p(d) r_{i,j(d)}^k$ is also a convex function in (k_1,k_2) for $k_1 \geq 1$ and $k_2 \geq 1$. However, q_i^k is not convex for $k_1 \geq 0$ and $k_2 \geq 0$ due to the setup costs, SC1, SC2.

Notice that the above discussion of the convexity of q_i^k is also true for the test quantity $q_i^k + \sum_j p_{ij}^k w_j$.

We have shown that the test quantity of this test problem does not "completely" satisfy the convexity requirement. However, all the test problems were solved during our computational experiments provided a sufficiently large radius r was used. The following offers some insights into observed robustness of the algorithm for this set of

test problems. Since the recurrent states tend to cluster in a small number of groups (possibly one), the range of policy choices for many states tends not to be constrained by the limit of the neighboring states within the specified radius, r. This is because the collection of neighboring states of all recurrent states usually includes the neighboring states within a radius larger than r of a single recurrent state.

COMPUTATIONAL RESULTS

The data examined by Lambrecht, Luyten and Muckstadt [6] were used as test data. In addition, the formulation was extended to a three stage problem. All data are given in TABLE 1. In TABLES 2-7, the CPU time in VAX 11/750 virtual seconds and the number of policy iterations necessary to solve each of the problems are given. A total of 36 different problems were solved. Each was solved conventionally using all states, and solved using the neighborhood procedure with radii (r) of 1, $\sqrt{2}$, $\sqrt{3}$, 2, $\sqrt{5}$. Of the 180 solutions using the neighborhood approach, all but 1 were solved optimally. 10 cheap iterations were used per policy iteration.

The convergence criteria used were: 1. no policy changes for any state within the neighborhood (of radius r) and 2. the sum of the absolute values of the residuals [19] must be less than a predetermined value. The value was set nominally at an order of magnitude above the round-off error capability of the computer for each particular problem. For the 121 state problems (TABLES 2 and 3), the average number of

alternatives per state is 50 and relatively small amounts of virtual storage are required. For the 1331 state problems (TABLES 4 and 5) and 10648 state problems (TABLES 6 and 7), the average number of alternatives per state are 147 and 137 respectively, and the requirements for virtual storage are approximately 2 megabytes and 15 megabytes respectively (the number of alternatives per state of the 10648 state problems was limited artificially).

For 121 state problems (TABLES 2 and 3), the reduction in computational effort runs generally about 67.2%. An exception is problem 1, which has an unusually large number of recurrent states. For the 1331 state problems the reduction averages 93.6%. For the 10648 state problems, the average reduction is greater than 99.5% in each case. In almost all cases the optimal solution was achieved. The exception occurs in problem 5, TABLE 5. This turned out to be a difficult problem for the procedure. For a radius of 2 the final policy set was non-optimal. However, the gain for that policy set was within 0.056% of optimal. In addition, TABLE 5 problem 6, solutions of radius $\sqrt{2}$ and $\sqrt{3}$ were not solved optimally initially. They were solved optimally with the following simple extension to the procedure. When the procedure converges, check a larger radius ($\sqrt{5}$ in this case) on the last policy iteration. If the convergence criteria is still satisfied, stop.

It is important to note that implementation of the procedure is extremely easy. The code was written in FORTRAN IV with only the simpliest of list processing applications

(a few pointers is all that is necessary). By arranging storage so that variables associated with a particular state are grouped together, the virtual memory software of the VAX 11/750 (or any other virtual machine) automatically keeps those segments of memory associated with the recurrent set plus neighborhood in core. Non-active states are cycled to the disk. This is particularly important for large problems.

In summary, an approach to solving specially structured large scale Markov Decision Processes has been presented. Experimental testing indicates that the computational savings over conventional (all states) approaches are considerable particularly for larger scale problems. Finally, the procedure is easily programmed and can be readily configured to take advantage of the natural strategies of virtual memory computers.

problem*		1	2	3	4	5	6
setup costs	stage 1	1	440	5	40	10	5
	stage 2	50	4	10	4	5	10
	stage 3	100	1	5	1	20	5
echelon	stage 1	10	5	7	5	1	1.3
holding	stage 2	1	4	15	4	0.5	2.7
cost	stage 3	1	3	7	3	2	1.3
variable	stage 1	10	50	7	50	10	7
production	stage 2	1	40	15	40	5	15
cost	stage 3	1	30	7	30	20	7
shortage	(TABLES 2,4,6)	100	200	100	200	50	100
cost	(TABLES 3,5,7)	200	400	200	400	100	200
demand(d)	d=0	.25	.25	.15	.25	.15	.25
distribution	d=1	.50	.50	.20	.50	.20	.50
	d=2	.25	.25	.30	.25	.30	.25
	d=3			.20		.20	
	d=4			.15		.15	

* 11 levels of inventory (0-10) for each stage results in 121 states for each 2-stage problem (TABLES 2,3) and 1331 states for each 3-stage problem (TABLES 4,5). 22 levels of inventory for each stage results in 10648 states for each 3-stage problem (TABLES 6,7).

TABLE 1 Test problem input data

(VAX 11/750) CPU time in Sec.

	all 121 states	r=1	$r=\sqrt{2}$	r=2	$r=\sqrt{5}$	Average Computation Reduction
problem 1	9.93	6.53	6.91	7.38	7.74	28.1%
problem 2	9.96	2.45	2.49	3.56	3.66	69.5%
problem 3	5.95	0.74	0.83	1.10	1.24	83.6%
problem 4	5.53	1.10	1.16	1.28	1.46	77.4%
problem 5	5.68	1.10	1.26	2.40	2.26	58.2%
problem 6	5.55	0.97	1.16	1.27	1.39	78.4%

TABLE 2

(VAX 11/750) CPU time in Sec.

	all 121 states	r=1	$r=\sqrt{2}$	r=2	$r=\sqrt{5}$	Average Computation Reduction
problem 1	9.68	6.86	7.93	9.99	9.86	10.5%
problem 2	9.20	1.69	1.74	2.17	2.65	77.6%
problem 3	5.89	0.96	1.07	1.33	1.56	79.1%
problem 4	5.57	1.09	1.21	6.52	5.96	33.7%
problem 5	6.99	0.95	1.09	1.57	2.28	78.9%
problem 6	4.84	6.52	1.62	1.35	2.93	35.8%

Note: r=1: each state has 4 neighboring states
$r=\sqrt{2}$: each state has 8 neighboring states
r=2: each state has 12 neighboring states
$r=\sqrt{5}$: each state has 20 neighboring states

TABLE 3

(VAX 11/750) CPU time in Sec.

	all 1331 states	r=1	$r=\sqrt{2}$	$r=\sqrt{3}$	r=2	$r=\sqrt{5}$	Average Computation Reduction
problem 1	636.27	8.34	10.71	11.91	13.71	20.56	97.9%
problem 2	10176.13	49.78	72.70	111.51	120.90	224.32	98.9%
problem 3	878.26	5.64	9.41	10.53	12.13	15.42	87.9%
problem 4	2176.90	5.28	9.84	8.52	9.76	12.08	99.6%
problem 5	606.74	12.40	11.22	14.88	14.53	25.47	97.4%
problem 6	441.79	23.21	33.70	40.33	28.92	31.23	92.9%

TABLE 4

(VAX 11/750) CPU time in Sec.

	all 1331 states	r=1	$r=\sqrt{2}$	$r=\sqrt{3}$	r=2	$r=\sqrt{5}$	Average Computation Reduction
problem 1	1358.61	13.88	17.60	18.86	21.73	31.80	98.5%
problem 2	997.62	28.62	49.13	55.04	53.10	96.66	94.3%
problem 3	961.36	18.00	22.59	24.19	24.83	28.31	97.5%
problem 4	939.59	10.86	12.14	13.09	14.17	14.39	98.6%
problem 5	446.93	89.81	226.24	263.95	86.25	125.65	64.6%
problem 6	842.57	21.08	48.84	54.35	31.04	33.40	95.5%

Note: r=1: each state has 6 neighboring states
$r=\sqrt{2}$: each state has 18 neighboring states
$r=\sqrt{3}$: each state has 26 neighboring states
r=2: each state has 32 neighboring states
$r=\sqrt{5}$: each state has 56 neighboring states

TABLE 5

(VAX 11/750) CPU time in Sec.

	all 10648 states	r=1	$r=\sqrt{2}$	$r=\sqrt{3}$	r=2	$r=\sqrt{5}$	Average Computation Reduction
problem 1	9460.38	14.30	15.88	17.44	22.70	28.43	99.8%
problem 2	22671.96	14.40	26.76	19.50	39.90	56.73	99.9%
problem 3	10262.93	7.74	9.24	9.55	15.84	16.63	99.9%
problem 4	8469.15	4.14	5.90	5.74	8.95	10.97	99.9%
problem 5	8137.63	4.30	5.73	5.92	10.98	14.39	99.9%
problem 6	10777.60	34.42	25.38	51.23	40.19	75.90	99.6%

TABLE 6

(VAX 11/750) CPU time in Sec.

	all 10648 states	r=1	$r=\sqrt{2}$	$r=\sqrt{3}$	r=2	$r=\sqrt{5}$	Average Computation Reduction
problem 1	10212.46	16.08	21.15	20.87	29.00	42.26	99.7%
problem 2	7002.88	8.95	12.33	10.70	17.00	30.21	99.8%
problem 3	7276.78	33.04	30.56	24.13	36.61	44.68	99.5%
problem 4	8947.70	7.67	9.41	9.64	9.53	12.62	99.9%
problem 5	7757.61	22.45	25.03	28.79	30.26	40.19	99.6%
problem 6	9754.22	40.77	50.43	49.28	53.71	60.39	99.5%

Note:
- r=1: each state has 6 neighboring states
- $r=\sqrt{2}$: each state has 18 neighboring states
- $r=\sqrt{3}$: each state has 26 neighboring states
- r=2: each state has 32 neighboring states
- $r=\sqrt{5}$: each state has 56 neighboring states

TABLE 7

REFERENCES

1. Bellman, R., "A Markov Decision Process", J. Math. Mech. 6, 679-684(1957).

2. _____, Dynamic Programming, Princeton Univ. Press, Princeton, New Jersey, 1957.

3. Clark , A. and Scarf, H., "Optimal Policies for a Multi-Echelon Inventory Problem", Management Science, Vol. 6, No. 4, 475-490(1960).

4. Hodgson, T. J., and Koehler, G. J., "Computation Techniques for Large Scale Undiscounted Markov Decision Processes", Naval Research Logistics Quarterly, Vol. 26, No. 4, 587-594(1979).

5. Howard, R. A., Dynamic Programming and Markov Processes, M.I.T. Press, Cambridge, Massachusetts, 1960.

6. Lambrecht, M. R. , Leuten, R. and Muckstadt, J. A., "Safety Stock Policies for Multi-Echelon Production Systems", International Journal of Production Research, To appear.

7. Manne, A. S., "Linear Programming and Sequential Decisions", Management Science, 6, 259-267(1960).

8. Mine, H., and Osaki, S., Markovian Decision Processes, American Elsevier Publishing Company, Inc., New York, 1970.

9. Morton, T. E., "On the Asymptotic Convergence Rate of Cost Difference for Markovian Decision Processes", Operations Res., 19, 244-248(1971).

10. Odoni, A. R.,"On Finding the Maximal Gain for Markov Decision Processes", Operations Res., 17, 857-860(1969).

11. Peterson, E. R., "Periodic Markov Programming", Canadian Operations Research Society, March, 39-44(1969).

12. Schweitzer, P. J., "Perturbation Theory and Markov Decision Processes", Technical Report No. 15, M. I. T., 1965.

13. Schweitzer, P. J., and Federgruen, A., "The Asymptotic Behavior of Undiscounted Value Iteration in Markov Decision Problems", Math. of Operations Res., Vol. 2, No. 4, 360-382(1977).

14. Schweitzer, P. J., and Seidmann, A., "Generalized Polynomial Approximations in Markovian Decision Processes", Working Paper Series No. QM8326, The Graduate School of Management, The University of Rochester, 1983.

15. Su, S. Y., and Deninger, R. A., "Generalization of White's Method of Successive Approximations to Periodic Markovian Decision Processes", Operation Res., 20, No. 2, 318-326(1972).

16. Wagner, H., "On the Optimality of Pure Strategies", Management Science, 6, 268-269(1960).

17. White, D. J., "Dynamic programming, Markov Chains, and Method of Successive Approximations", J. Math. Anal. Appl. 6,373-376(1963).

18. Wolfe P., and Dantzig, G. B., "Linear Programming in a Markov Chain", Operations Res., 10, 702-710(1962).

19. Zaldivar, M., and Hodgson, T. J., "Rapid Convergence for Markov Decision Processes", Decision Science, 6, 14-24(1975).

END

FILMED

11-85

DTIC