EINDHOVEN UNIVERSITY OF TECHNOLOGY
and
GRADUATE SCHOOL OF MANAGEMENT, DELFT.

COSOR Memorandum 80-20

Some notes on Iterative optimization of structured Markov decision processes with discounted rewards.

by

Marcel Hendrikx, Jo van Nunen and
Jaap Wessels

November 1980
The Netherlands

Abstract

The paper contains a comparison of solution techniques for Markov decision processes with respect to the total reward criterion. It is illustrated by examples that the effect of a number of improvements of the standard iterative method, which are advocated in the literature, is limited in some realistic situations.
Numerical evidence is provided to show that exploiting the structure of the problem under consideration often yields a more substantial reduction of the required computational effort than some of the existing acceleration procedures.
We advocate that this structure should be analyzed and used in choosing the appropriate solution procedure. This procedure might be composed by blending several of the acceleration concepts that are described in literature. Four test problems are sketched and solved with several successive approximation methods. These methods were composed after analyzing the structure of the problem. The required computational efforts are compared.

1. Introduction

In recent years a number of papers appeared that described improvements of iterative methods for computing the total expected reward of a (semi-) Markov decision process. The proposed improvements all aim to reduce the required computational effort. However, they try to reach that goal in a different and sometimes even conflicting way.
Of each of the proposed improvements or variants of the standard successive approximation scheme there is numerical evidence that it works more efficiently in specific situations than the standard method. See, e.g. MacQueen's iteration method which incorporates the concept of bounds for the optimal solution [11].
Moreover, we refer e.g. to Van Nunen [17] who claims that value oriented methods are preferable, Hastings and Van Nunen [8] who advocate the advantage of action elimination, Porteus [22] who shows the efficiency of extrapolation methods and finally Bartmann [1] who gives numerical evidence that the Bisection method is very efficient.

However, although each of the proposed variants will have its specific value, we will illustrate by some examples, that the effect of each of them might be limited if one has to solve a real problem.
Numerical evidence is provided to show that exploiting the structure of the problem under consideration might yield a more substantial reduction of the required computational effort than some of the existing acceleration procedures. We advocate that this structure should be analyzed and used in choosing the appropriate solution procedure. The choice of a solution procedure will depend on that structure, with other words the structure of the problem will determine the way in which the respective acceleration concepts, are blended when solving a real problem.
One might argue that it is preferable to have one solution procedure available for all kind of problems. However, for practical applications, this is an unrealistic argument, as will be shown by the numerical results that are given and discussed in the final section of this paper.
The fact that one has to construct the solution procedure depending on the structure of the problem might be disappointing at first sight.
However, the construction of algorithms that exploit the structure of the problem is not extremely difficult in practice.
In fact, the reverse is true, (almost) all practical problems possess a certain structure and the use of that structure might enable you to find a solution in a reasonable time which might otherwise be impossible.
Especially in practical situations one has to solve the problem again and again with different values of the parameters as well as with e.g. aggregated and decomposed state and action spaces. This has to be done e.g. to evaluate several alternatives. So the reduction of the computation time might be very valuable. The numerical examples that we will give, will show how large the computational gain can be. We draw two examples from the existing literature but the other two stem in fact from two real life applications that were analyzed.
We don't claim that the comparison of methods that we will give will be exhaustive. We even are not in a position to give the best solution procedure for certain classes of problems. However,

the numerical experiences show how important it is to exploit the structure of the problem under consideration for the choice of an adequate solution procedure. Moreover, they show some directions in which this structure can be exploited and how the several acceleration concepts work in some specific situations.

In general successive approximation methods are preferable over the classical methods like policy iteration [9] and linear programming [5][13]. However, some problems may have a structure for which a policy iteration type of procedure is efficient. This holds e.g. for some G/M/s quening control systems, as was shown in Van Nunen and Puterman [19].

We will first introduce the model and some notation in order to be able to describe the relevant notions less verbal.
We consider a system which at discrete points in time ($t = 0, 1, 2, \ldots..$) can be identified as being in one of a finite number of states. The state space is $S := (1, 2, \ldots, N)$.
If the system is observed to be in state i at time t, an action a may be chosen from a finite set of actions $A = (1, \ldots.., k)$.
As a result of this action a the system moves to state j at time $t + 1$ with probability $p_{ij}^{a} \geq 0$ and an (expected) one stage reward $r(i, a)$ is earned. We assume $\sum_{j \in S} p_{ij}^{a} = 1$ for all $i \in S$ and $a \in A$. The objective is to maximize the total expected discounted rewards over an infinite time horizon and to determine a decision rule for which this maximal return is achieved.
The discount factor is $\beta < 1$. The restriction to discounted problems with $\sum_{j \in S} p_{ij}^{a} = 1$ is only chosen for the simplicity of the exposition, see e.g. [16][21].
A policy f is a function from $S \to A$ and a strategy π is a sequence of policies $\pi = (f_0, f_1 \ldots\ldots.)$.
So, if we use strategy π, then action $f_t(i)$ is chosen at time t if the system is observed to be in state i at that time.
A strategy is called stationary of all component functions f_t are equal i.e. $\pi = (f, f, f, \ldots.)$.
By r^f we denote the vector on S with components $r(i,f(i))$.
By P^f we denote the $N * N$ matrix with (i, j)-th component equal to $p_{ij}^{f(i)}$.

Let strategy π be given and let the starting state be state i. By the random variables X_t and A_t we denote the state and the action of the system at time t respectively.
Now $v^{\pi}(i)$ is defined by

$$v^{\pi}(i) = \mathbb{E}_{i,\pi} \sum_{t=0}^{\infty} \beta^t \, r(X_t, A_t), \tag{1}$$

The total expected discounted reward over an infinite time horizon given that the starting state is $i \in S$ and that strategy π is used.
$\mathbb{E}$ denotes the expectation with respect to the probability structure generated by π and i. By v^{π} we denote the vector with components $v^{\pi}(i)$.
For a stationary strategy $\pi = (f, f, f, ...)$ we have

$$v^f := v^{\pi} = \sum_{t=0}^{\infty} \beta^t (P^f)^t \, r^f \tag{2}$$

The goal is to determine v such that

$$v^* = \sup_{\pi} v^{\pi} \tag{3}$$

and to determine a strategy π^* for which v^* is attained or approximated.
It is well known, that under the simple conditions that we have here, there exists a policy f^* such that $v^{f^*} = v^*$.

The standard successive approximation method (SSA) introduced by Bellman [2] in 1957 can be used to determine f^* and v^{f^*}
In fact this SSA forms the basis for the variants that we will discuss. We define the mappings L^f and U from $V \to V$ for the set V of real valued functions v on S.

$$L^f v = r^f + \beta P^f v \tag{4}$$

$$Uv = \max_f L^f v = \max_f \{r^f + \beta P^f v\} \tag{5}$$

These mappings are used to formulate the following classical result.

Lemma 1 (Blackwell ([3])

L^f and U are monotone contraction mappings on V with fixed points v^f and v^t respectively. The contraction factor is β.
Moreover, for $v_o \in V$ and v_n defined by

$$v_n = Uv_{n-1} =: L^{fn}v_{n-1} \qquad (6)$$

We have

$$v_n \to v^*$$

with a rate of convergence that is equal to
Moreover,

$$v^{fn} \to v^*$$

with f_n the policy for which $U v_{n-1}$ is maximal.
Note that (6) can be expressed component-wize by

$$v_n(i) = \max_{a \in A} \{r(i,a) + \sum_{j \in S} \beta p^a_{ij} v_{n-1}(j)\} \quad i \in S \qquad (7)$$

The convergence of this standard successive approximation (SSA) method expressed in (6) or (7) is in general rather slow. Therefore several variants of the SSA-method have been introduced. The goal of these variants are different and can be divided into three groups.

The first group tries to use the information collected during the iteration process to get better estimates of v^*. This group contains in fact two basic principles of which several subvariants are available in literature.
These basic principles are

a) successive approximation methods which incorporate the computation of upper and lower bounds for the optimal value vector v^* in each iteration step of the actual algorithm. MacQueen [11], Porteus [20].

b) successive approximation methods which use extrapolations to v^*. Porteus [22].

In the second group of variants one tries to reduce the contraction factor. This should lead to a gain in the required number of iterations. Again there are 2 basic variants

c) variants in the policy improvement procedure (the maximization) step of the successive approximation method. Hastings [6], Reetz [24], Wessels [25], Van Nunen [16], Porteus [21], Van Nunen and Stidham [18]

d) the Bisection method in which in some iterations a contraction factor of .5 instead of β is achieved. Bartmann [1].

The third group tries to reduce the computational effort that is required to compute for each $i \in S$ the maximum over all actions $a \in A$ of the sum as given in the righthand side of [7] There are again 2 basic concepts.

e) S.A.methods that incorporate a test for the elimination of actions that can be identified as being non-optimal for a number of iteration steps. So, for this actions the computation of the mentioned sum can be avoided.
MacQueen [12], Hastings [7], Hastings and Van Nunen [8].

f) Value oriented successive approximation methods which provide better values for v^{f_n} by executing a number of times the mapping L^{f_n} instead of U, so that for these steps the maximization can be avoided. Morton [15], Van Nunen [16], [17], Puterman [23].

Of course one will use a combination of the above basic principles if one constructs an algorithm for solving a particular problem. However, the effects of above variants might be conflicting and depend heavily on the structure of the problem.
For example in a problem with a large number of states but with only a small number of decisions in each state, like it is the case in machine replacement problems where the only options could be to repair or to replace the machine, the computational effort required for the incorporation of an action elimination procedure might be more than the gain that can be achieved.
If, however for each state a lot of actions are possible the variants (e) and (f) might work quite well.

For example if the transition probabilities have a particular structure e.g. each matrix P^f is almost upper triangular, then this structure can be exploited by using a Gauss-Seidel variant. These variants belong to the class described under c).
As an example of conflicting effects we can use the effects that are achieved if one composes a procedure by using e.g. a Gauss-Seidel variant as well as the concept of bounds.
The Gauss-Seidel variant might cause an improvement in the contraction-rate but it might cause worser bounds. Which of these effects will be the most important can not be said in general, as we will see later.
Exploiting the structure of the problem might also lead to enormous gains in required computation time as is illustrated next. Many practical problems like the inventory and replacement problems we will discuss in this paper possess the property that p^a_{ij} is in fact independent of i. This is illustrated in the following simple example.
Suppose we have a single item inventory system where the states 0, 1, 2, ... N represent the available inventory at the beginning of each period e.g. each week. Orders are placed at the beginning of each week and delivery is instanteneously. The demand in each period equals k with probability q_k. If we define the decision a as the inventory level just after delivery, than $p^a_{ij} = q_{a-j}$ independent of i.
So if one computes in each iteration step in advance for each $a \in A$

$$d(a) = \sum_j \beta\, p^a_{\cdot j}\; v_{n-1}(j) \qquad (8)$$

one finds that (7) can be written as

$$\max_a\{r(i,a) + \sum_j \beta p^a_{ij}\; v_{n-1}\;(j)\} = \max_a\;\{r(i,a) + d(a)\} \qquad (9)$$

This is just one of the examples of how the specific structure can be used. Similar ideas can be used in computing e.g. the expected one-stage reward, if the underlying process is separable, see [4].

Moreover, the structure of optimal policies can be exploited. The combination of certain variants in relation with using the structure of the problem might also lead to conflicting effects. For example the idea expressed in (9) can not be exploited if e.g. a Gauss-Seidel variant is used.
The above discussion explains also why we did not use the same solution procedures for all four test problems.

In section 2 we discuss, in short, the available accelaration procedures. Section 3 is used to sketch the four test problems. Numerical results are given in section four.

2. Variants of successive approximation methods

In this section we will give a brief description of the underlying ideas of each of the acceleration procedures.

2.a Bounds for the optimal value vector v^*

The concept of Bounds for v^* was introduced by MacQueen[11]. Consider the SA method as described in (6) or (7). Then

$$\begin{aligned} Uv_n - v_n &= L^{f_{n+1}} v_n - L^{f_n} v_n \le L^{f_{n+1}} v_n - L^{f_{n+1}} v_n \\ &= (r^{f_{n+1}} + \beta P^{f_{n+1}} v_n) - (r^{f_{n+1}} + \beta P^{f_{n+1}} v_{n-1}) \\ &= \beta P^{f_{n+1}} (v_n - v_{n-1}) \\ &\leqslant \beta \max\{v_n(i) - v_{n-1}(i)\}\, e \end{aligned}$$

where e is the vector on S with all components equal to 1.
The difference between $v_{n+2} = U^2 v_n = U(Uv_n)$ and v_n is bounded from above by

$$U^2 v_n - v_n = U^2 v_n - Uv_n + Uv_n - v_n \le (\beta+\beta^2) \max_i\{v_n(i) - v_{n-1}(i)\}e$$

In general $U^k v_n - v_n$ can be estimated by

$$v_{n+k} - v_n = U^k v_n - v_n < (\beta+\beta^2+..\beta^k)\max_i\{v_n(i)-v_{n-1}(i)\}e \qquad (10)$$

Since $U^k v_n \to v^*$ it follows that an upperbound for v^* is given by

$$v^{f_n} \leq v^* \leq v_n + \frac{\beta}{1-\beta} \max\{v_n(i) - v_{n-1}(i)\}. \tag{11}$$

Note that (10) can also be used to obtain an upperbound for v_{n+k}.
Similarly a lowerbound l_n can be determined.

$$l_n := v_n + \frac{\beta}{1-\beta} \min_i \{v_n(i) - v_{n-1}(i)\} \cdot e \leqslant v^{f_n} \leqslant v^*. \tag{12}$$

The above bounds are referred so as the MacQueen bounds (MQB), see (11), and (17). So, a S.A.-algorithm could be

$$\left\{\begin{array}{ll} \text{choose} & v_o \in V \\ \text{compute} & v_n = Uv_{n-1} \\ \text{stop if} & (u_n - l_n) < \varepsilon \text{ or } u_n - l_n \leq \varepsilon \|v_n\| \end{array}\right. \tag{13}$$

The ε-optimal policy with which the above procedure ends is f_n and a good estimate for v^{f_n} and v^* is

$$\tfrac{1}{2}(u_n + l_n)$$

The above algorithm converges at least with a rate $\beta\gamma$ where γ is the subdominant eigen value of the matrix P^{f^*}. See [14]. This is based on the following result (see [10], [14]).

$$\begin{aligned} \text{span } (v_n - v_{n-1}) &= \max_i \{v_n(i) - v_{n-1}(i)\} - \min_i \{v_n(i) - v_{n-1}(i)\} \\ &\leqslant \beta\gamma \text{ span } (v_{n-1} - v_{n-2}). \end{aligned}$$

If one uses more information of the actual transition matrices, improved bounds can be obtained, see [26]
However, in general this will cost additional computational effort. In the derivation of the MacQueen bounds (MQB) u_n and l_n as given in (11) and (12) we used that for all $i \in s$ the sum $\sum_j p_{ij}^a = 1$.
If, however, this equal-row-new property does not hold, more complicated bounds have to be constructed. In this case we have $\beta \sum_j p_{ij}^a \neq \beta \sum_j p_{kj}^a$
Now, a straightforward extension of the MQB will lead to

$$\left\{\begin{array}{l} \tilde{u}_n = v_n + \frac{\alpha}{1-\alpha} \max_i (v_n(i) - v_{n-1}(i)) \cdot e \\ \tilde{l}_n = v_n + \frac{\alpha_n}{1-\alpha_n} \min_i (v_n(i) - v_{n-1}(i)) \cdot e \end{array}\right. \tag{14}$$

with $\alpha = \max_{i,a} \Sigma p_{ij}^{a}$ and $\alpha_n = \min_{i} \sum_{j} p_{ij}^{f_n(i)}$

In this more general situation $\alpha \geq \alpha_n$.

If $Uv_o \leq v_o$, these bounds need to be adapted slightly (see [16], [20]). Note, that in this case the difference between $\tilde{u}_n$ and $\tilde{l}_n$ cannot be expressed by means of the span $(v_n - v_{n-1})$ unless $\alpha = \alpha_n$

The use of a variant of the policy improvement procedure e.g. a stopping-time as described in section 2.c, transforms the problem into an equivalent problem for which the equal row sum property does not hold. This occurs e.g. if a Gauss-Seidel variant is used. So in that case one could use the more complicated bounds (14). In order to restore in such cases the equal row sum property one needs an additional transformation, see [18], [21].

2. b <u>Extrapolations</u>

It would seem a good idea to use the following S.A.-algorithm in which v_n is replaced by $\bar{v}_n$ which is the best current estimate of v^* based on the MacQueen bounds.

$$\begin{cases} \bar{v}_o \in V \\ v_n = U\bar{v}_{n-1} \\ \bar{u}_n = v_n + \frac{\beta}{1-\beta} \max_i\{v_n(i) - \bar{v}_{n-1}(i)\}\, e \\ \bar{l}_n = v_n + \frac{\beta}{1-\beta} \min_i\{v_n(i) - \bar{v}_{n-1}(i)\}\, e \\ \bar{v}_n = \frac{1}{2}(\bar{u}_n + \bar{l}_n) \end{cases} \qquad (15)$$

In the case of equal row sums, the difference $\bar{u}_n - \bar{l}_n$ equals $u_n - l_n$ as defined in (11) and (12).

So, the convergence is not improved by using the above algorithm (15) in the case of equal row sums.

However, in the case of unequal row sums, as might occur after using a variant of the policy improvement procedure, as described in section 2.c, a considerable gain in required computational effort might be obtained. The Extrapolation algorithms use the following idea.

$$\begin{cases} v_o \in V \\ \tilde{v}_n = Uv_{n-1} \\ v_n = \tilde{v}_n + c_n \end{cases} \qquad (16)$$

with c_n chosen appropriatly. For example in the case of unequal row sums (14) can be used to derive

$$c_n = \tfrac{1}{2}\left[\frac{\alpha}{1-\alpha}\max_i\{\tilde{v}_n(i) - v_{n-1}(i)\} + \frac{\alpha_n}{1-\alpha_n}\min\{v_n(i)-v_{n-1}(i)\}\right]$$

For a number of extrapolation methods and numerical evidence see Porteus [22].

2.c Variants of the policy improvement procedure

The S.S.A.-method described in (6) and (7) is often referred to as the pre-Jacobi method. Alternatives for the policy improvement step can be obtained by constructing mappings $\tilde{L}^f$ and $\tilde{U}$ instead of L^f and U such that the sequence $\tilde{v}_n$ defined by

$$\begin{cases} \tilde{v}_o \in V \\ \tilde{v}_n = \tilde{U}v_{n-1} := \max_f\{\tilde{r}^f + \tilde{P}^f\,\tilde{v}_{n-1}\} =: \tilde{L}^{f_n}v_{n-1} \end{cases} \qquad (17)$$

still converges to v^* at a geometric rate, i.e.

$$\tilde{v}_n \to v^* \qquad (18)$$

Often the goal is to define $\tilde{U}$ in such a way that the resulting convergence rate is smaller then β. Some of the policy improvement variants like Gauss-Seidel procedures, overrelaxation methods etc. are well known [6] [24]. A unified approach can be given by using the concept of stopping times. For details see e.g. Wessels [25], Van Nunen [16] and Van Nunen and Stidham [18]. These variants can be generated, also by using a (pre-inverse) transformation of the data as introduced by Porteus [21]. From a numerical point of view the advantage of having a smaller spectral radius $\rho(\tilde{P}^t) \epsilon \rho(\beta P^t) = \beta$ might be diminished by the fact that the transformed problem does not necessarily possess the equal-row sum property. As an example of a variant of the policy improvement procedure we describe the Gauss-Seidel variant.

$v_o \in V$ compute for $i = 1, 2, \ldots\ldots, N$

$$v_n(i) = \max_a \left\{ \frac{r(i,a) + \sum_{j<i} p_{ij}^a\, v_n(j) + \sum_{j>i} p_{ij}^a\, v_{n-1}(j)}{1 - p_{ii}^a} \right\} \quad (19)$$

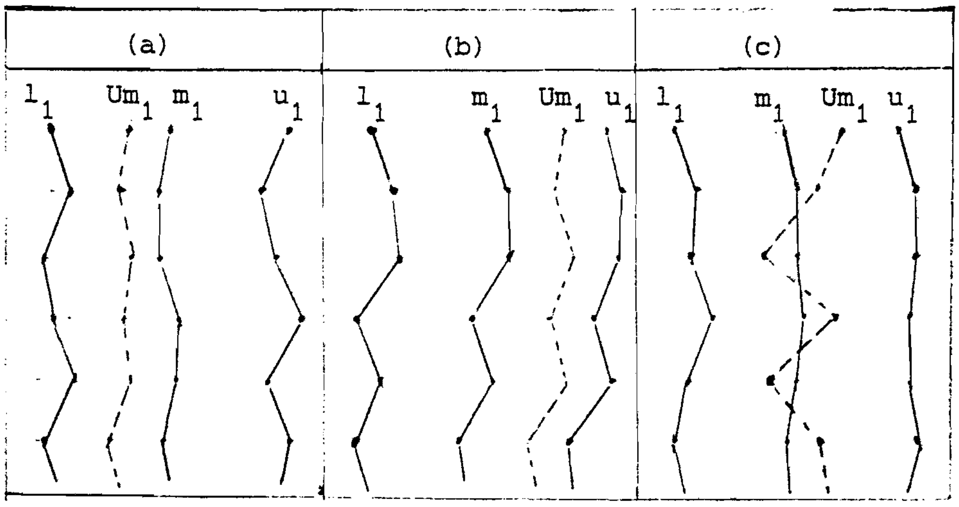
In this case the corresponding $\tilde{r}^f$ and $\tilde{\rho}^f$ have a particular form (see [16],[21]).
If the transition matrices are almost lower triangular a procedure based on (19) might yield good results.
We will refer to (19) as G.S.1.
For (almost) upper triangular problems, the procedure that starts with computing for state N, N-1, ... respectively

$$v_n(i) = \max_a \left\{ \frac{r(i,a) + \sum_{j>i} p_{ij}^a\, v_n(j) + \sum_{j<i} p_{ij}^a\, v_{n-1}(j)}{1 - p_{ij}^a} \right\}$$

will be preferable. We will refer to this variant as G.S.2.

2.d) The Bisection method

The Bisection method was introduced by Bartmann (1). By using the monotonicity property of the mapping U it is tried to make the contraction factor equal to ½ for as many steps as possible.
Let $v_0 \in V$ such that $Uv_0 \geq v_0$ and let $v_n := Uv_0$. Let l_1 and u_1 be as defined in (11) and (12). Note that $l_1 \leq v^* \leq u_1$.
Let $m_1 := \frac{1}{2}(l_1+u_1)$ and in general $m_n := \frac{1}{2}(l_n+u_n)$. Compute Um_1; now there are three possibilities which are described in the following picture



(a) If $Um_1 \leq m_1$ for all components, then $v^* \leq m_1$, which implies that $u_2 := Um_1$, $l_2 := l_1$ are also upper and lower bound for v^*.

(b) If $Um_1 \geq m_1$, then $l_2 := Um_1$, $u_2 := u_1$ are upper and lower bound for v^*.

Note that in the cases (a) and (b) we have

$$\| u_2 - l_2 \| \leq \tfrac{1}{2} \| u_1 - l_1 \|$$

(c) If (a) and (b) donot hold, we have to adjust the bounds according to (11) and (12). In this case

$$\| u_2 - l_2 \| < \beta \| u_1 - l_1 \|$$

Repeating the above procedure with $m_n = \tfrac{1}{2}(l_n + u_n)$ until l_n and u_n are close enough, results in an algorithm which might converge in a very-fast way. The speed of the convergence will depend on the number of bisection steps that is made, as is nicely shown in the examples.

2.e) The elimination of suboptimal actions

In the n-th step of the algorithm (6) one has to compute for all $i \in S$ the following term

$$\max_{a \in A} \{ r(i,a) + \beta \sum_{j \in S} p_{ij}^a v_{n-1}(j) \}$$

The goal of a sub-optimality test is to eliminate a number of irrelevant actions such that for these actions the summation $\sum_{j \in S} p_{ij}^a v_{n-1}(j)$ can be avoided.
The idea of using upper and lower-bounds in a procedure for eliminating actions, was given by MacQueen [12]. In [12] MacQueen showed how actions can be identified as being non-optima for the rest of the iteration process. In [7], [8] Hastings and Hastings and Van Nunen showed how similar ideas can be used to eliminate actions only temporarily. Suppose that we are in the situation of equal row sums. Then action $a \in A$ cannot be optimal in the next iteration step if

$$r(i,a) + \sum_j p_{ij}^a v_n(j) < v_n(i) + \beta \min_j (v_n(j) - v_{n-1}(j))$$

since $v_{n+1}(i) \geq v_n(i) + \beta \min_j (v_n(j) - v_{n-1}(j))$

This can also be done for subsequent iteration steps by using for v_n in the lefthand side of (10) upper bounds for v_{n+k} like defined in (10) while in the righthand side a similar expression with lower bounds is used. For detailed information on the action elimination (AE) see [8]. There some numerical evidence is also given.

2.f Value oriented methods

An other way to reduce the amount of computational effort is by decreasing the number of maximization steps as was proposed in [17].

Let $v_n = L^{fn} v_{n-1} = U v_{n-1}$.

Instead of determining v_{n+1} by performing a maximization step, one could proceed first with a number of iterations that use L^{fn} instead of U. This idea is expressed in the following S.A.-algorithm.

$$\begin{cases} v_0^{(\lambda)} \in V \qquad \lambda \in N \\ v_n^{(\lambda)} = (L^{fn})^{\lambda} v_{n-1} = \underbrace{(L^{fn} L^{fn} \ldots L^{fn}}_{\lambda\text{-times}} v_{n-1}))\ldots) \\ \qquad =: U^{(\lambda)} v_{n-1} \\ \text{with } L^{fn} v_{n-1} = U v_{n-1}. \end{cases} \tag{21}$$

However, the mapping $U^{(\lambda)}$ is neither necessarily monotone nor contracting. Nevertheless convergence to v* is preserved, see [16]. For numerical evidence of this method see (17), and the examples in section 4.

3. The test problems

We combined several of the variants discussed in section 2 to determine the optimal policy and the corresponding total expected discounted reward for four problems. These problems are briefly described in this section. Typical for these problems is that they have a lot of structure.

3.1 Howards auto replacement problem

This problem is described extensively in [9]. A car owner considers his situation every three months. The state of the system is determined by the age of his car, expressed in periods of 3 months.
It is supposed that a car of age 40 (10 years) is worn out. State 40 is also used to identify a car that is total loss. So, the number of states is 41. In each state he can sell his car and buy another (second hand or new one) of an age between 0 and 39, keeping the car is denoted by -1. So, the number of possible decisions in each state is 41. A car of age i has a probability of p_i to reach state 40 within the three months period, so for each i the number of probabilities $p^a_{ij} \neq 0$ is 2.
Costs are composed of purchasing costs, selling costs and expected repair and maintenance costs, which depend of course on the state (age) of the car.
The goal is to determine the policy for which the total expected discounted rewards are minimal. For the numerical exercise we took $\beta = 0,97$.

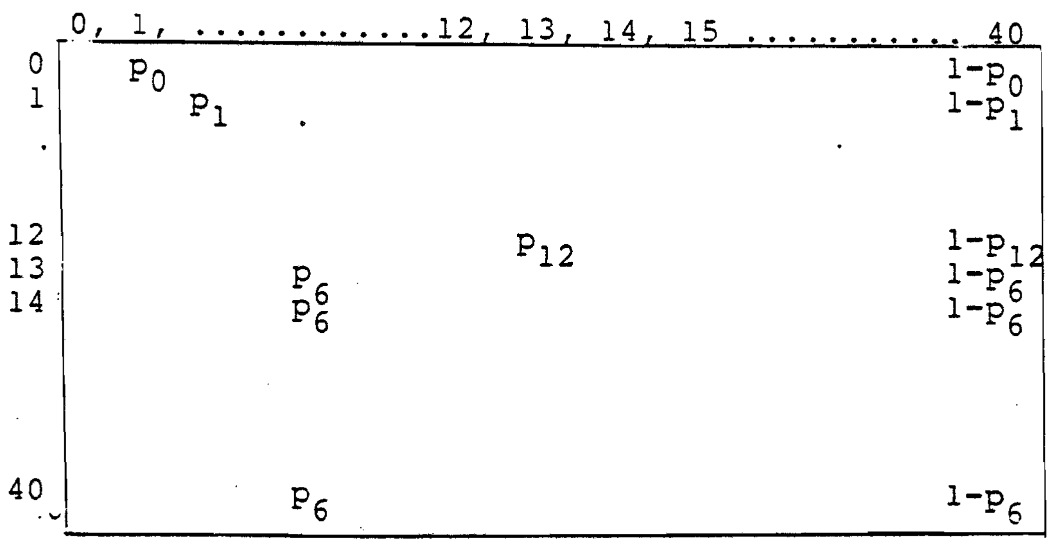


Figure 3.1.

The figure shows the structure of the transition matrix

Note that the problem is almost periodic, since the probabilities P_6 to P_{12} are close to 1.

3.2 The replacementproblem of Hastings

For details we refer to [6].

A machine is considered at discrete, equidistant points in time. The state of the machine is determined by its age and the level of required repair and maintenance costs for the next period. The time interval (period) is chosen such that the possibility of two break downs in a period can be neglected. We consider the situation that the age of a machine is maximally 100 periods and for each age there are 10 repair cost levels. So the number of states is 1000. Denoted by $\{(1,1)(1,2), \ldots.(1,10), (2,1) \ldots.. (100,10)\}$.

In each state there are two possible actions e.g. reparation of the machine (0) or replacement by a new machine (1). Repair costs depend on the level as well as on the age of the machine.

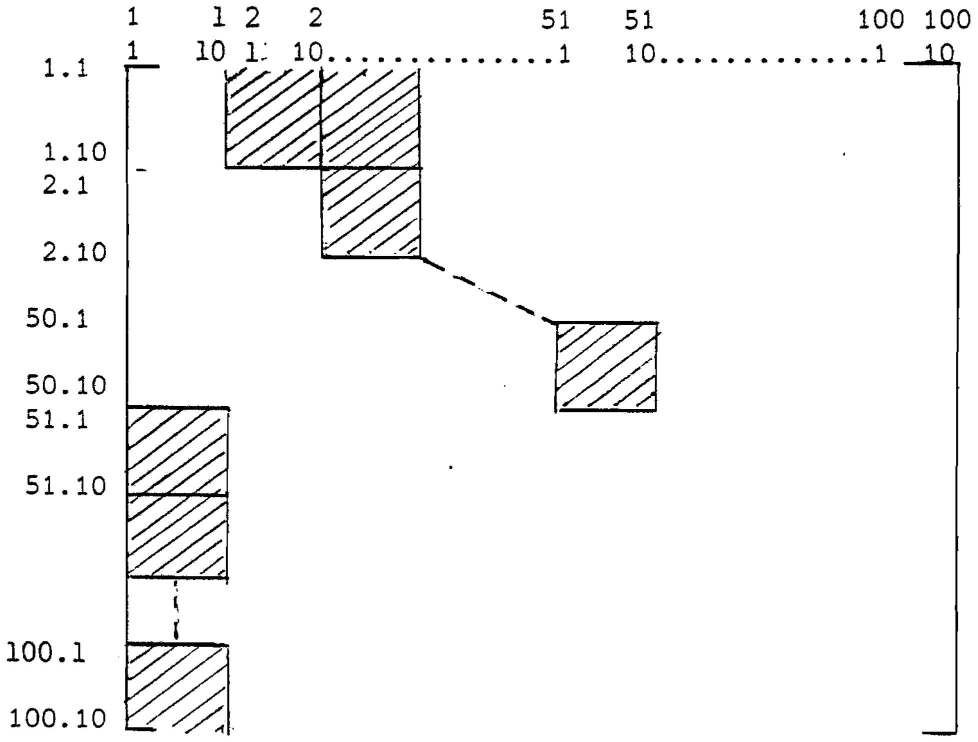


Figure 3.2. The figure shows the structure of the matrix P^f for $f((i,j)) = 0$ for $i \leq 50$ and $f((i,j)) = 1$ for $i > 50$ and $j \in (1,2,\ldots 10)$

3.3 A hard-cash inventory system

For details we refer to [26].
In this problem a cash-money-inventory system is considered. On one hand customers deposit money into the bank (negative demand) while on the other hand they cash money to do some of their (small) payments (positive demand). So the bank has to take care that enough hard cash is available. However, too much money means a loss of interest.
The options for the bank are to order or deposit money at the main-office. This possibility is available at the end of every morning, delivery is almost immediately. In the meantime emergency transports of money are possible against relatively high costs. It appeared that the positive or negative demand for money in a "normal" week has a stable but stochastic behaviour, that differs over the days and within a day between morning and afternoon. The week has been divided in 10 periods, representing the mornings and afternoons of the workdays.
By considering for each period 30 possible cash-levels, we can denote the state space by

$S = ((1,0),(1,1)\ldots(1,29),(2,0)\ldots(2,29),(3,0),\ldots(10,29))$,
where (i,j) indicates period i and cash-level j.
The decisions are the amounts to order or to deposit at the main bank. These are supposed to be taken at the beginning of the even periods (the afternoons). The average number of possible decisions at these points in time is about 20.
Transition probabilities have been determined with the demand distribution of hard cash by customers. Costs consist of ordering and deposit costs, the inventory costs (loss of interest) and costs of emergency orders which could be placed at the main bank, if the bank runs out of hard cash during the periods. The structure of a transition matrix is sketched in figure 3.3.

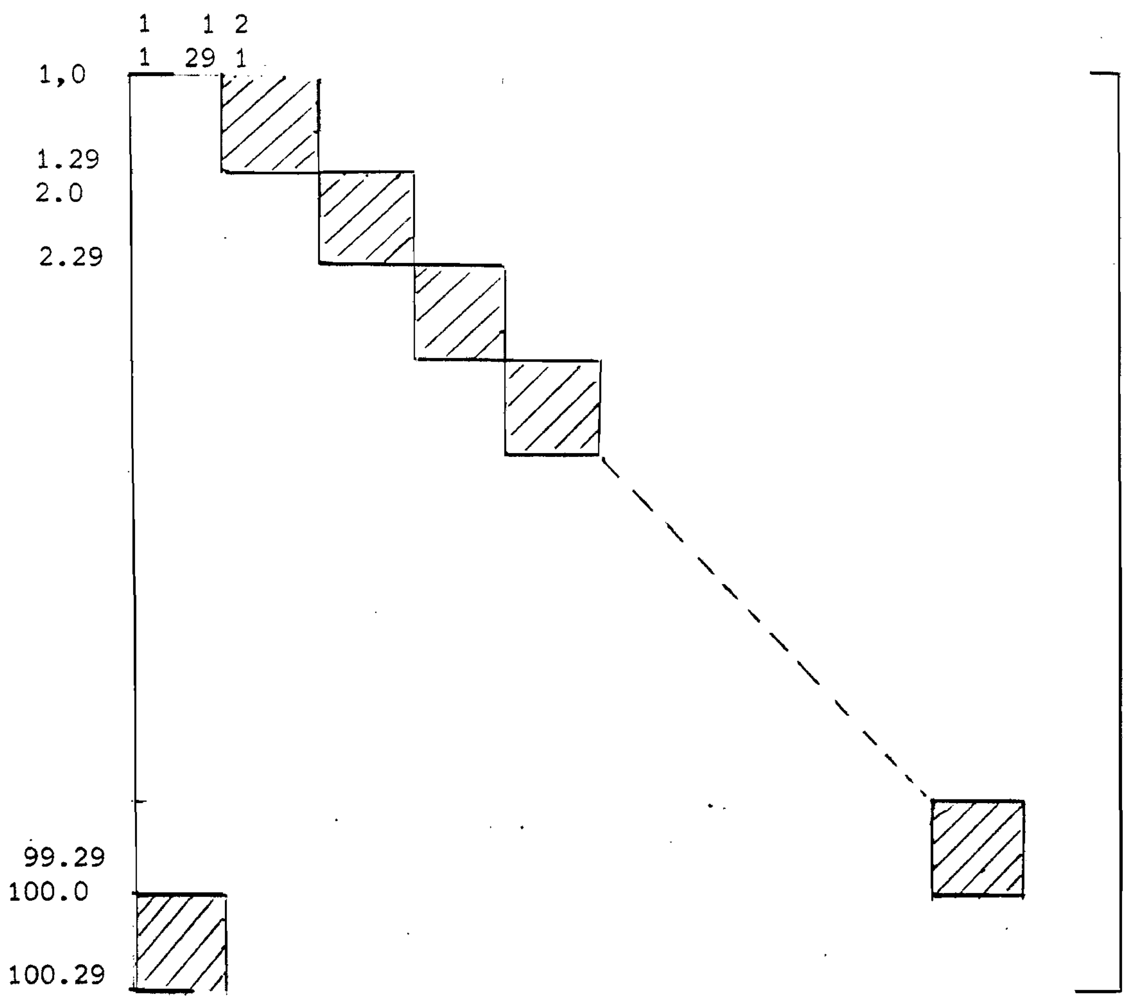


Figure 3.3 The figure shows the structure of a transition matrix P^f. The structure given above is independent of f.

3.4 A Three point inventory system

For details see [27]. We consider a three point inventory system, as outlined in Figure 3.4, at equidistant points in time.

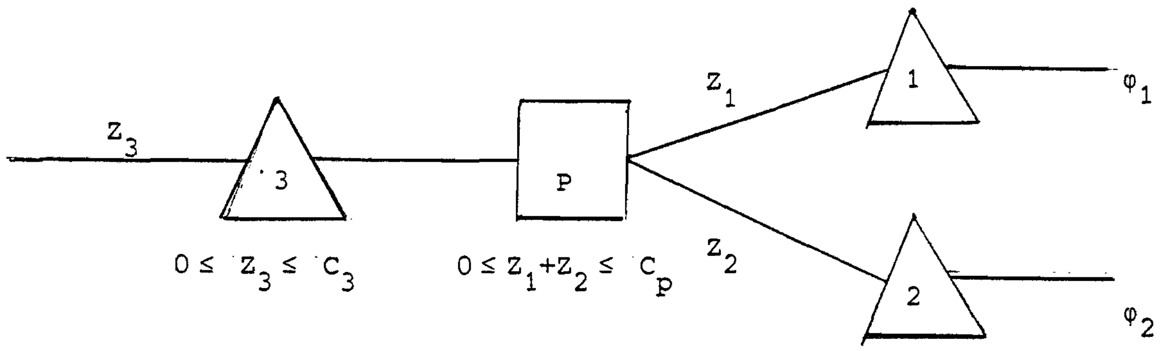


Figure 3.4: a three-point inventory system.

In warehouse 1 and 2 a product is stored. The product in 1 and 2 is produced by production unit P, which has maximal capacity C_P. In warehouse 3 an essential part of the final product is stored. Of this part up to C_3 can be ordered at a time. Backlogging is allowed in warehouse 1 and 2 but not in 3. The delivery times for 1, 2 and 3 are equal to one period. The states of the system can be defined by a triple (X_1, X_2, X_3) which gives the inventory in each of the respective warehouses.
The decisions exist of the amounts Z_1 and Z_2 ordered by warehouses 1 and 2 at the production unit P and the amount Z_3 of the subunit ordered by warehouse 3.
The transition probabilities are determined by the demand at the two warehouses which is given by its distributions φ_1 and φ_2. The costs are constituted by inventory, ordering, and stock out costs.
We considered 1000 different inventory level combinations and in the average 73 decisions in each state.

4. Numerical results and comments

Before analyzing the required computational effort for each of the problems, we will give some technical information. The numerical results given in this section are achieved with the Burroughs B7700 computer of Eindhoven University of Technology. We chose for the programs a maximal processing time of 300 CPU-seconds. The programs that were stopped after 300 CPU-seconds are indicated in the tables with, EMP, exceeded maximum processing time. Especially for the larger problems 3 and 4 we see that a number of methods required more than 300 CPU-seconds.
The numerical information is given in table 4.1 - 4.4. The first column indicates the solution procedure that is constructed by combining the variants as discussed in section 2.

On the basis of the tables 4.1 - 4.5 we will discuss some of the numerical results and relate them with the structure of the problem. However, first some remarks about the con-

structed algorithms will be made.
The four problems were first solved by using the standard successive approximation method with MacQueen bounds.
In these algorithms the structure of the problem was exploited by using the idea expressed in (8) and (9).
As might be expected the advantage of using this structure was most clear for problem 4. For this example the number of 73000 (i,a)-combinations was reduced to 2060 relevant combinations. This reduction was achieved by taking into account also the capacity limit for the production unit and the order restriction for warehouse 3. For this problem it appeared in fact that using the structure was essential.
Next, in algorithm 2, the action elimination procedure was incorporated to check how the advantage of this procedure as indicated in (8) was diminished by using the structure of the problem. We did not run this variant for problem 4 since the effect (EMP) was foreseeable at that time.
Thirdly, the Gauss-Seidel variant has been computed with "Gauss-Seidel" bounds (G.S.B.). Again problem 4 was not processed because it was clear that it could not be processed within 300 CPU-seconds.
In the remaining algorithms we combined the advantage of MacQueen bounds and Gauss-Seidel procedures by alternating a number of Gauss-Seidel steps with one standard successive approximation step. This enabled us to use the MacQueen bounds. In the tables this is indicated e.g. by 50 G.S.1 and 1 S.S.A. with MQB.
Depending on the structure we used G.S.1 or G.S.2 or both variants alternatingly. Similar results can be obtained by reordering the state space.
For numerical evidence see Porteus [22]. The use of the Gauss-Seidel variant was in example 3 essential for achieving a solution in a reasonable time. This was caused by the typical periodic (upper triangular) structure. For almost periodic problems the second largest eigenvalue is still almost equal to β. So, especially if β is close to 1, the number of required iterations might be rather large. In the first example

with a Gauss-Seidel variant, it produced the best result.
For the three point inventory problem it worked only efficiently in the combination where the structure of the problem could still be exploited. The number of real bisection iterations is given together with the total number of iterations.

Concluding, one may say that we did not provid a recipe according to which a solution procedure should be chosen for a certain (class of) problem(s).
However, we discussed some devices which might help substantially in finding a suitable solution procedure. Moreover, we showed that exploiting the structure of a problem can be essential for constructing good algorithms.

REFERENCES

[1] Bartmann, D., "A method of bisection for discounted Markov decision problems". Zeitschrift für Oper.Res. 23 (1979), 275-287.

[2] Bellman, R., A markovian decision process. J. Math. Mech. 6 (1957), 679-684.

[3] Blackwell, D., "Discounted dynamic programming". Ann.Math. Statist. 36 (1965), 226-235.

[4] Denardo, E.V., "Separable Markovian decision problems". Management Sci. 14 (1968), 279-289.

[5] d'Epenoux, F., Sur un problème de production et de stockage dans l'aléatoire. Rev.Franc.Rech.Opér. 14 (1960), 3-16.

[6] Hastings, N.A.J., Some notes on dynamic programming and replacement. Oper.Res.Q.19 (1968), 453-464.

[7] Hastings, N.A.J., A test for nonoptimal actions in undiscounted finite Markov decision chains, Management Sci. 23 (1976), 87-91.

[8] Hastings, N.A.J. and J.A.E.E. van Nunen, The action elimination algorithm for Markov decision processes, 161-170 in H.C. Tijms, J. Wessels (eds.), Markov decision theory, MC-tract 93, Mathematical Centre, Amsterdam, 1977.

[9] Howard, R.A., Dynamic programming and Markov decision processes, Cambridge (Mass.), M.I.T.-Press, 1960.

[10] Hübner, G., Improved procedures for eliminating suboptimal actions in Markov programming by the use of contraction properties in transactions of the 7th Prague Conference on Information theory, Statistical decision functions. Random Processes, Prague, Academia 1977, 257-263.

[11] MacQueen, J., A modified dynamic method for Markovian decision problems, J. Math.Anal.Appl.14 (1966), 38-43.

[12] MacQueen, J., A test for suboptimal actions in Markovian decision problems. Oper.Res. 15 (1967), 559-561.

[13] Manne, A.S., Linear programming and sequential decisions. Management Sci. 6 (1960), 259-267.

[14] Morton, T. and W. Wecker, Discounting ergodicity and convergence for Markov decision processes, Management Sci. 23 (1977), 890-900.

[15] Morton, T.E., Undiscounted Markov Renewal programming via modified successive approximations. Oper.Res. 19 (1971), 1081-1089.

[16] Van Nunen, J.A.E.E., Contracting Markov decision processes. Amsterdam, Mathematical Centre (Mathematical Centre Tract 71), 1976.

[17] Van Nunen, J.A.E.E., A set of successive approximation methods for discounted Markovian decision problems. Zeitschrift für Oper.Res. 20 (1976), 203-208.

[18] Van Nunen, J.A.E.E. and S. Stidham jr., Action dependent stopping times and Markov decision processes with unbounded rewards. To appear in O.R.-Spectrum.

[19] Van Nunen, J.A.E.E. and M. Puterman, On computing optimal policies for G/M/S Quening systems, Working paper no. 715, April 1980. Faculty of Commerce, University of British Columbia, Vancouver, Canada.

[20] Porteus, E.L., Some bounds for discounted sequential decision processes, Management Sci. 18 (1971), 7-11.

[21] Porteus, E.L., Bounds and transformations for discounted finite Markov decision chains. Oper.Res.23 (1975), 761-784.

[22] Porteus, E.L., "Improved iterative computation of the expected discounted return in Markov and semi-Markov chains", Research paper no. 443, Stanford University, 1978.

[23] Puterman, M.L. and M.C. Shin, Modified policy iteration algorithms for discounted Markov decision problems. Management Sci. 24 (1978), 1127-1137.

[24] Reetz, D., Solution of a Markovian decision problem by overrelaxation. Z. Oper. Res. 17 (1973), 29-32.

[25] Wessels, J., Stopping times and Markov programming, 575-585 in Transactions of the 7th Prague Conference on Information theory, Stat,Decision Functions, Random Process, Prague, Academia 1977, 575-585.

[26] Wessels, J., Markov decision processes: Implementation aspects. Memorandum Cosor 80-14. Eindhoven University of Technology, Department of Mathematics. Eindhoven.

[27] Wijngaard, J. and R.A.A.M. Geilleit, A heuristic method for an inventory problem with two stages and two final products. Research Report, Eindhoven University of Technology, Department of Industrial Engineering (1981) to appear.

Table 4.1

HOWARD AUTOREPLACEMENT PROBLEM			
METHOD	Computation Time	Standard iterations	Average time of an iteration
1. SSA+MQB	CPU 1.93 I/O 1.08	76	.025
2. SSA+MQB+AE	CPU 3.60 I/O 1.08	76	0.047
3. GS1+GSB	CPU 7.23 I/O 1.22	289	.025
4. 15*(GS1+GS2) SSA+MQB	CPU 3.92 I/O 1.08	120	.026
5. GS1.VI15 SSA+MQB	CPU 1.19 I/O 1.08	328	.0034
6. DSE+GS1 SSA+MQB	CPU 3.78 I/O 1.21	116	.032
7. RSE+GS1 SSA + MQB	CPU 3.53 I/O 1.08	133	.026
8. BISECTION SSA	CPU 2.73 I/O 1.26	94_{12}	.032
9. BISECTION GS2	CPU 1.57 I/O 1.08	44_{16}	.035

Table 4.1 Comparison of some methods for Howard's Autoreplacement problem, # states 41; # actions in each state is 41 and $\beta = 0.97$ relative error 10^{-4}.

Table 4.2

HASTINGS REPLACEMENT PROBLEM			
METHOD	Computation time	# Standard iterations	Average time of an iter.
1. SSA+MQB	CPU 194.10 I/O 2.43	3146	.062
2. SSA+MQB	CPU 308.34 I/O 1.12 EMP	2700	.114
3. GSI+GSB	CPU 213.61 I/O 2.56	4299	.049
4. 100*(GS1+GS2) SSA+MQB	CPU 35.99 I/O 2.43	910	.039
5. GS1+GS2+VI 15$^+$+ SSA+MQB	CPU 41.78 I/O 2.43	1209	.035
6. 50*(GS1+GS2+DSE) SSA+MQB	CPU 6.15 I/O 2.43	104	.059
7. 50*(GS1+GS2+RSE) SSA+MQB	CPU 7.74 I/O 2.42	104	.074
8. BISECTION SSA	CPU 267.89 I/O 4.23	4122 7	.649
9. BISECTION GS1+GS2	CPU 7.37 I/O 2.16	53 29	.139

Table 4.2. Comparison of four methods for the replacement problem of Hastings. # states 1000; # actions in each state 2; β = .998; relative error 10^{-4}.

HARD CASH INVENTORY PROBLEM			
METHOD	Computation Time	# Standard iterations	Average time of an iteration
1. SSA + MQB	CPU 306.21 I/O 5.49 EMP	400	.077
2. SSA + MQB + AE	CPU 308.87 I/O 144 EMP	330	.93
3. GS2 + GSB	CPU 301 I/O 4.99 EMP	380	.79
4. 50*(GS1+GS2) SSA+MQB	CPU 303.23 I/O 4.995 EMP	390	.78
5. 50*GS2 +100 VI SSA + MQB	CPU 307 I/O 1.359 EMP	970	0.32
6. 100 GS2+DSE SSA + MQB	CPU 84.34 I/O 1.89	204	0.41
7. 100 GS2+RSE SSA + MQB	CPU 84.75 1.75	204	0.41
8. BISECTION SSA	CPU 296 EMP	387 5	.76
9. BISECTION GS2	24.94 1.67	26 23	.96

Table 4.3. Comparison of some methods for the "hard cash" inventory problem. # states 300; # action in each state 20 for even periods; β = .999; relative error 10^{-4}.

THREE POINT INVENTORY PROBLEM			
METHOD	Computation Time	# Standard iterations	Average time of an iteration
1. SSA + MQB	CPU 37.24 I/O 2.74	18	1.51
2. SSA + MQB + AE	-	-	-
3. GS1 + GSB	-	-	-
4. 5 (GS1+GS2) + 5VI ISSA + MQB	CPU 300.51 I/O 1.215 EMP	30	10.02
5. GS1 + GS2 + 5 VI ++ ISSA + MQB	CPU 300.101 I/O 1.26 EMP	197	1.52
6. 5 *(GS1+GS2+DSE) SSA + MQB	CPU 296.31 I/O 1.21 EMP	18	16.44
7. 5 (GS1+GS2+RSE) SSA + MQB	CPU 302.86 I/O 1.21 EMP	18	16.78
8. BISECTION SSA	CPU 59.16 I/O 2.25	28 21	2.11
9. BISECTION GS1 + GS2	CPU 303.10 I/O 1.12 EMP	19 3	15.95

Table 4.4 Comparison of several methods for the three point inventory problem. #states 1000; average #action for each state 73; β = .997 relative error 10^{-3}.