

# A COMPARISON OF POLICY ITERATION AND VALUE ITERATION ALGORITHMS APPLIED TO THE SOLUTION OF INFINITE HORIZON MARKOV DECISION PROCESSES

Ryan Meuth

University of Missouri- Rolla  
Department of Electrical and computer Engineering  
1870 Miner Circle,  
Rolla, MO, 65401

## ABSTRACT

This paper documents the evaluation of the performance of policy iteration and value iteration algorithms as applied to infinite-horizon Markov decision processes through comparative and quantitative analysis of the two algorithms on a common set of Markov decision problems.

## 1. INTRODUCTION

To evaluate the performance of policy iteration and value iteration algorithms, two infinite-horizon Markov decision problems are formulated, the algorithms are implemented and applied with varying parameters, and the results presented.

## 2. MARKOV DECISION PROCESSES

Markov Decision Processes consist of a set of states  $\mathbf{S}$ , which may be discrete or continuous, a set of actions  $\mathbf{A}$ , which may also be discrete or continuous, a set of transition probabilities  $\mathbf{P}$ , which define the probability of transitioning from state  $s$  to  $s'$  given action  $a$ , where  $s$  and  $s'$  are elements of  $\mathbf{S}$  and  $a$  is an element of  $\mathbf{A}$ . For each action that can be taken in each state, there is an associated reward  $\mathbf{r}$  which is an element of  $\mathbf{R}$ , the set of all rewards for each state-action pair. In a Markov decision process, the current state depends only on the previous state and the action taken there, with no influence from the history of states and actions previously. In this way, we can define  $\pi$  as a policy, or decision rule, which defines which action to take, given the current state of the process. Thus, maximizing the total reward during the time in the system is the measure for finding the optimal policy,  $\pi^*$ . Towards this end, several algorithms have been developed to solve this problem, namely Backward Induction for MDPs with finite horizon lengths, and Value Iteration and Policy Iteration for MDP's of infinite horizon lengths.

## 2.1 Value Iteration

Gauss-Seidel Value Iteration finds a numerical solution to the MDP by the method of successive approximation. It is not guaranteed to find the optimal decision rule for infinite-horizon problems, but is able to find a  $\mathcal{E}$ -optimal decision rule, which is often close enough to a globally optimal solution to be of use.

### 2.1.1 Value Iteration Algorithm [1]

1. Specify  $v^0(s)$  for all  $s \in S, \mathcal{E} > 0$  and set  $n=0$ .
2. Set  $j=1$  and go to 2a.
  - a. Compute  $v^{n+1}(s_j)$  by

$$v^{n+1}(s_j) = \max_{a \in A} \left\{ r(s_j, a) + \lambda \left[ \sum_{i < j} p(s_i | s_j, a) v^{n+1}(s_i) + \sum_{i \geq j} p(s_i | s_j, a) v^n(s_i) \right] \right\}$$

- b. If  $j = N$  where  $N$  is equal to the size of the state space, go to 3. Else increment  $j$  by 1 and return to step 2a.

3. If  $\|v^{n-1} - v^n\| < \frac{\mathcal{E}(1-\lambda)}{2\lambda}$

Goto step 4. Otherwise increment  $n$  by 1 and return to step 2.

4. For each  $s \in S$ , choose

$$d^\mathcal{E} \in \arg \max_{a \in A_s} \left\{ r(s, a) + \sum_{j \in S} \lambda p(j | s, a) v^{n+1}(j) \right\}$$

To find the  $\mathcal{E}$ -optimal decision rule.

## 2.2 Policy Iteration

In contrast to Value Iteration, where the values of the states are approximated, and then the policy is found based on state-values, policy iteration modifies the policy first, then estimates the value of the policy. The policy is then improved and the algorithm repeats until a stopping criteria is reached.

### 2.2.1 Policy Iteration with Gauss-Seidel Approximation and Action Elimination [1].

1. Initialization: Select  $\epsilon > 0, \delta > 0$  and  $M > 0$  and set  $n=0$ . For each  $s \in S$ , let

$$d_0(s) \in \arg \max_{a \in A_s} \{r(s, a)\}$$

Set

$$v^0(s) = \frac{1}{1-\lambda} \Lambda(r_{d_0})$$

$$u_0^0(s) = r_{d_0}(s) + \frac{\lambda}{1-\lambda} \Lambda(r_{d_0})$$

$$\Lambda(Bv^0) = 0 \quad \Upsilon(Bv^0) = sp(r_{d_0})$$

2. Policy Evaluation

- a. If  $M=0$ , set  $k=0$  and go to 2e. Otherwise, set  $j=1, k=1$ , and go to 2b.
- b. Set

$$u_n^k(s_j) = r_{d_n}(s_j) + \lambda \sum_{i < j} p_{d_n}(s_i | s_j) u_n^k(s_i) + \lambda \sum_{i \geq j} p_{d_n}(s_i | s_j) u_n^{k-1}(s_i)$$

- c. If  $j=N$  go to step 2d. Otherwise increment  $j$  by 1 and return to step 2b.
- d. If  $k=M$  or  $sp(u_n^k - u_n^{k-1}) < \delta$  go to step 2e. Otherwise, increment  $k$  by 1, set  $j=1$ , and return to step 2b.
- e. Set

$$m_n = k, v^{n+1}(s) = u_n^k(s), \forall s \in S$$

and go to step 3.

3. Action Elimination

- a. For each  $s \in S$  let  $E_s^{n+1}$  be the set  $a \in A_s^n$  for which

$$r(s, a) + \lambda \sum_{j \in S} p(j | s, a) v^n(j) < v^{n+1}(s) + \frac{\lambda^{m_n+1}}{1-\lambda} \Lambda(Bv^n) - \frac{\lambda}{1-\lambda} \Upsilon(Bv^n)$$

$$\text{Set } A_s^{n+1} = A_s - E_s^{n+1}$$

- b. If for each  $s \in S$ ,  $A_s^{n+1}$  has only one element, then stop and set  $d^*(s)$  equal to  $A_s^{n+1}$ , otherwise, continue.

4. Policy Improvement

For each  $s \in S$  and all  $a \in A_s^{n+1}$ , compute and store

$$L(s, a) v^{n+1} \equiv r(s, a) + \lambda \sum_{j \in S} p(j | s, a) v^{n+1}(j)$$

Set

$$u_{n+1}^0(s) = \max_{a \in A_s^{n+1}} \left\{ r(s, a) + \lambda \sum_{j \in S} p(j | s, a) v^{n+1}(j) \right\}$$

Let

$$d_{n+1}(s) \in \arg \max_{a \in A_s^{n+1}} \left\{ r(s, a) + \lambda \sum_{j \in S} p(j | s, a) v^{n+1}(j) \right\}$$

And Set,

$$\Lambda(Bv^{n+1}) = \min_{s \in S} \{u_{n+1}^0(s) - v^{n+1}\}, \Upsilon(Bv^{n+1}) = \max_{s \in S} \{u_{n+1}^0(s) - v^{n+1}\}$$

5. Stopping Criteria

If

$$sp(Bv^{n+1}) < \frac{1-\lambda}{\lambda} \epsilon$$

Then Stop. Set  $d_\epsilon = d_{n+1}$ . Otherwise increment  $n$  by 1 and return to step 2.

## 3. PROBLEM FORMULATION

### 3.1 Simple Bandit Problem

The first problem formulated for analysis of the algorithms is a simple bandit problem with the following characteristics:

“A decision maker observes a discrete-time system which moves between states {s1, s2, s3, s4} according to the following probability matrix:

$$P = \begin{bmatrix} 0.3 & 0.4 & 0.2 & 0.1 \\ 0.2 & 0.3 & 0.5 & 0.0 \\ 0.1 & 0.0 & 0.8 & 0.1 \\ 0.4 & 0.0 & 0.9 & 0.9 \end{bmatrix}$$

At each point of time, the decision maker may leave the system and receive a reward of  $R=20$  units, or alternatively remain in the system and receive a reward of  $r(s_i) = i$ , if the system occupies state  $s_i$ . If the decision maker decides to remain in the system, its state at the next decision epoch is determined by  $P$ . Assume a discount rate of  $\lambda = 0.9$ .” [1(275)]

Given this problem description, the problem can be formulated with 5 states – the four system states described above and a fifth terminal state representing quitting the system, resulting in  $S = \{s1, s2, s3, s4, s5\}$ .

The problem has two actions, C and Q. C represents staying in the system and letting the system transition as determined by  $P$ , and Q represents quitting the system and transitioning to  $s5$  with probability 1.0. The transition probabilities then become

$$P(s, C) = \begin{bmatrix} 0.3 & 0.4 & 0.2 & 0.1 & 0.0 \\ 0.2 & 0.3 & 0.5 & 0.0 & 0.0 \\ 0.1 & 0.0 & 0.8 & 0.1 & 0.0 \\ 0.4 & 0.0 & 0.0 & 0.6 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}$$

And,

$$P(s, Q) = \begin{bmatrix} 0.0 & 0.0 & 0.0 & 0.0 & 1.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 1.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 1.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 1.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix}$$

The reward structure for action C is as follows:

$$R(s, C) = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 \\ 2 & 2 & 2 & 2 & 0 \\ 3 & 3 & 3 & 3 & 0 \\ 4 & 4 & 4 & 4 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

And the reward structure for action Q becomes:

$$R(s, Q) = \begin{bmatrix} 20 & 20 & 20 & 20 & 0 \\ 20 & 20 & 20 & 20 & 0 \\ 20 & 20 & 20 & 20 & 0 \\ 20 & 20 & 20 & 20 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

### 3.2 The Lazy Adaptable Lion

The second problem formulated for evaluation has the following characteristics:

“An adult female lion requires about 6kg of meat per day and has a gut capacity of 30kg; which means that, with a full guy, it can go six days without eating. Zebras have an average edible biomass of 164kg, large enough to meet the food demands of many lions. If a female selects to hunt in a group, the probability of a successful kill per chase has been observed to increase with group size to a point, and that each chase consumes 0.5kg of food. The probabilities of kill are given by

$$\begin{aligned} Pk(1) &= 0.15 & Pk(4) &= 0.40 \\ Pk(2) &= 0.33 & Pk(5) &= 0.42 \\ Pk(3) &= 0.37 & Pk(6) &= 0.43 \end{aligned}$$

Where  $P(n)$  represents the kill probability per chase for lions hunting in a group of size  $n$ .”[1 (73)]

In this problem, the objective is to maximize the probability of survival over an infinite number of days, assuming that the lioness is able to hunt once per day. If a kill occurs, meat is shared equally among the hunting party.

In this case, the set of states are taken as the quantity of meat in the lioness’ stomach, discretized to single kilograms. The state  $s0$  is a terminal state yielding a reward of 0, while all other states yield a reward of 1. Transition probabilities are given by finding the successor state given the action, and assigning the probability of kill ( $Pk(a)$ ) to that transition.  $1-Pk(a)$  is assigned to the

transition from a state to itself, and all other probabilities are set to zero. The action set is taken as the size of the hunting party, up to 6 members.

#### 4. ALGORITHM PERFORMANCE

To measure the performance of the algorithms, convergence data was collected for values of  $\mathcal{E}$  between 0.1 and 0.00001 in orders of magnitude.

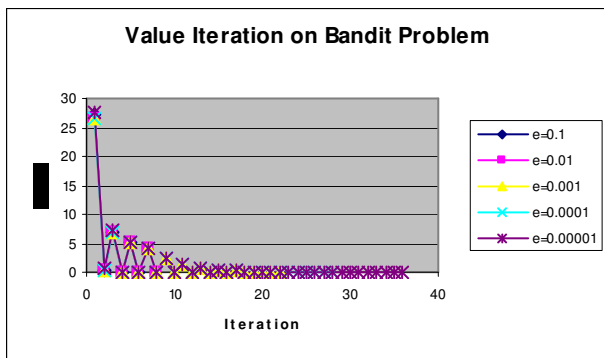
##### 4.1 Value Iteration

###### 4.1.1 Simple Bandit Problem

For this problem, Value Iteration converged on the following policy:

- S0: 1
- S1: 1
- S2: 0
- S3: 0
- S4: 0

With the Following Performance:



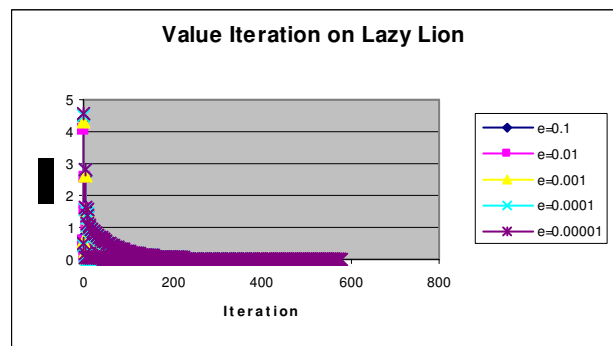
###### 4.1.2 Lazy Adaptable Lion Problem

For this Problem, Value iteration converged on the Following Policy:

- S0: 0
- S1: 5
- S2: 4
- S3: 3
- S4: 2
- S5: 1
- S6: 3
- S7: 5
- S8: 4
- S9: 3
- S10: 2
- S11: 4
- S12: 1
- S13: 5
- S14: 4
- S15: 3
- S16: 0

- S17: 2
- S18: 1
- S19: 5
- S20: 4
- S21: 1
- S22: 3
- S23: 2
- S24: 1
- S25: 5
- S26: 2
- S27: 4
- S28: 3
- S29: 2
- S30: 1

With the Following Performance:



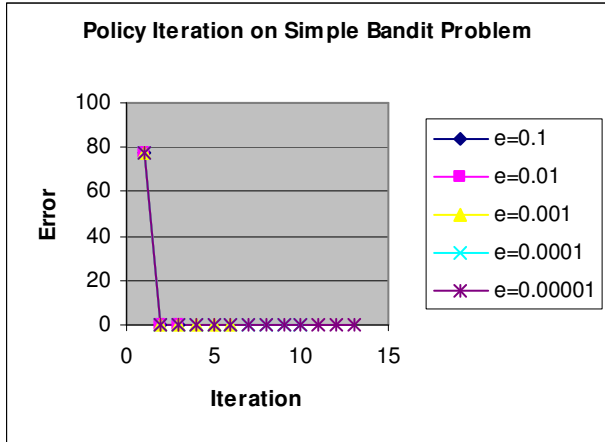
##### 4.2 Policy Iteration

###### 4.2.1 Simple Bandit Problem

For all values of  $\mathcal{E}$ , Policy Iteration converged quickly to the following policy:

- Policy:
- S0: 1
  - S1: 1
  - S2: 1
  - S3: 0
  - S4: 0

Convergence Graph:



#### 4.2.2 Lazy Adaptable Lion Problem

For this Problem, Policy Iteration found the Optimal Policy in a single iteration utilizing Action Elimination.

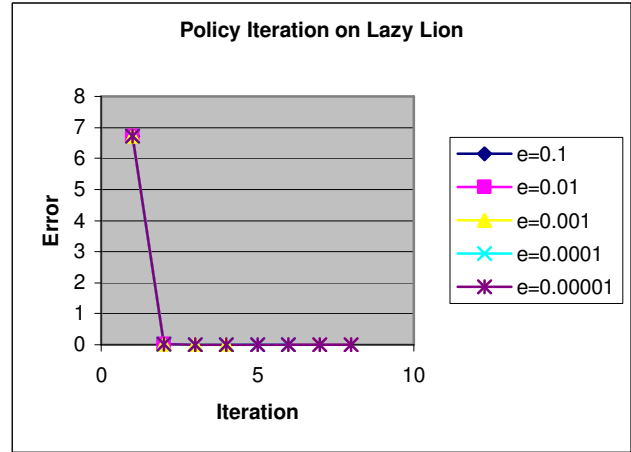
Policy:

- S0: 3
- S1: 1
- S2: 1
- S3: 1
- S4: 1
- S5: 1
- S6: 1
- S7: 1
- S8: 1
- S9: 1
- S10: 1
- S11: 1
- S12: 1
- S13: 1
- S14: 1
- S15: 1
- S16: 1
- S17: 1
- S18: 1
- S19: 1
- S20: 1
- S21: 1
- S22: 1
- S23: 1
- S24: 1
- S25: 1
- S26: 2
- S27: 1
- S28: 1
- S29: 1
- S30: 1

With Action Elimination turned off, and  $\lambda > 0.75$ , policy iteration fails to converge within programmatic limits, though it does find the same policy as above.

For  $\lambda \leq 0.75$ , Policy iteration converges quickly.

Convergence Graph:



## 5. CONCLUSIONS

The use of Policy Iteration to solve infinite-horizon MDPs is clearly superior to Value iteration, as for each of the problems and  $\epsilon$  values, Policy Iteration converged on a solution within an order of magnitude or 2 less than value iteration.

## 6. REFERENCES

- [1] Puterman, Martin L., *MARKOV DECISION PROCESSES: Discrete Stochastic Dynamic Programming*, John Wiley & Sons, Inc., New York, NY, 1994.