CS6700 - Programming Exercise II

Sabarinath N P (CS10B020)

1.1 Objective

To find the optimal policy for herd management problem using dynamic programming techniques. The optimal policy is characterized by the maximum money that can be generated from the given herd.

1.2 Problem Description

Given a Herd that can accomodate a maximum of 12 cows. There are 3 types of cows, namely,

- young
- breedable
- old

And, given the expected utility and selling price of each type of cow along with the transition probabilities and the offspring probabilities, the aim is to determine the optimal action, i.e., the number of cows to sell at each state so that the overall money generated over the time is maximized.

1.3 Implementation

The problem of deriving the optimal policy boils down to calculating the state-to-state transition probability matrix efficiently as the dynamic programming methods to derive the optimal actions for each state, is well known [1]. States are represented as $\langle s_{young}, s_{breedable}, s_{old} \rangle$. And action or number of cows of each type being sold is represented as $\langle a_{young}, a_{breedable}, a_{old} \rangle$. Each states is given a numerical value in base 13, with number of young cows forming the most significant digit, for representation convenience. Since the actions are deterministic, for given state and action, an *afterstate* can be defined.

With this representation and the given transition probabilities, the probability of transition from a state (or an afterstate) to another state can be derived as follows,

- **Evolution step**: Compute the transition probability matrix $PE_{s_a s''}$ for afterstate $\rightarrow$ state, which involves only the transition among types or evolution of cows. In this step, the size of the herd is preserved.
- **Breeding step**: Compute the transition probabilty matrix $PB_{s'' s'}$ for state $\rightarrow$ state, which involves only the production of *young* type cows. In this step, $s_{breedable}$ and s_{old} remains constant, while the s_{young} increases depending on $s_{breedable}$ and the total number of cows in the herd. And the sum of probabilities of transitioning to the invalid states, i.e., states with more than 12 cows, is added to the probability of transitioning to the state $\langle 12 - s_{breedable} - s_{old}, s_{breedable}, s_{old} \rangle$.

- **Double Transition**: To efficiently compute the transition probability matrix $P_{ss'}$ involving afterstate → evolution_state → state, the above two matrices can be multiplied to obtain the required, state (or afterstate) → state transition probability matrix.

$$P_{ss'} = P_{s_a s'} = PE_{s_a s''} * PB_{s'' s'}$$

Given the afterstate and the next state, reward $R_{ss'}$ for the transition can be computed using state information, utility and payoff values. Having computed the required transition probability matrix $P_{ss'}$, the simulation of policy and value iteration is quite straight forward given the algorithms. The correctness of the computed probability matrices M_{ij} can verified by,

$$\sum_{i=0}^{454} M_{si} = 1 \qquad \forall(s) \in S$$

1.4 Simulation

Given $P_{ss'}$ and $R_{ss'}$, the following algorithms which follow dynamic programming paradigm is run for different discount parameters.

- Policy Iteration
- Value Iteration

The discount parameter γ is varied as [0.3, 0.5, 0.7, 0.9] and the above two methods are run for each value of γ. The convergence parameter θ is set to 10^{-9} for both the methods.

1.5 Observation

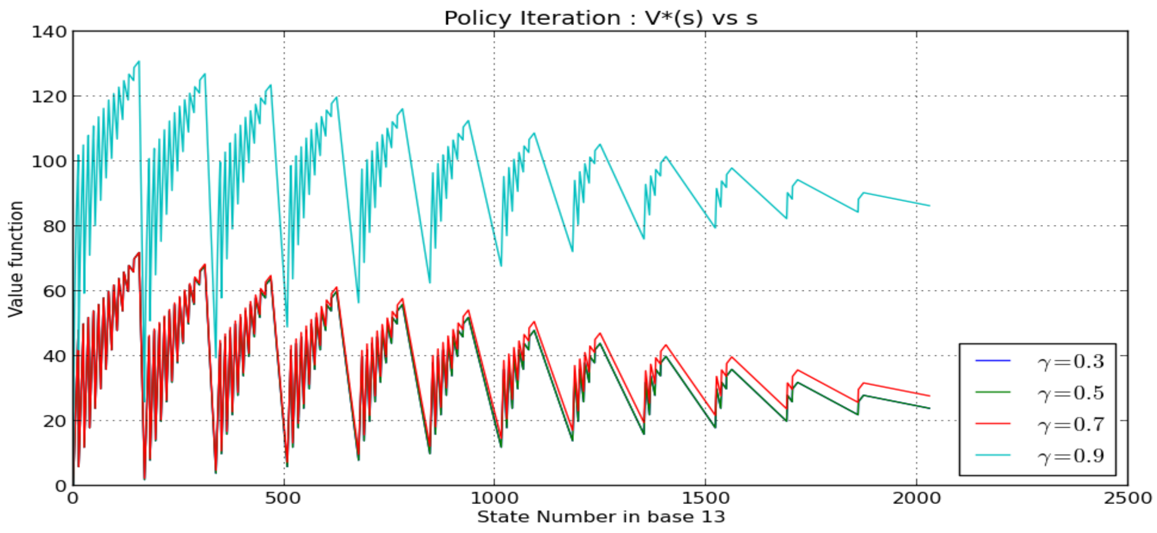
The above mentioned algorithms are run and the convergence results are observed. It is noted that Policy Iteration takes relatively very less number of iterations to converge for large discount parameter γ when compared to Value Iteration. The number of iterations taken in each of the simulation is shown in the following table,

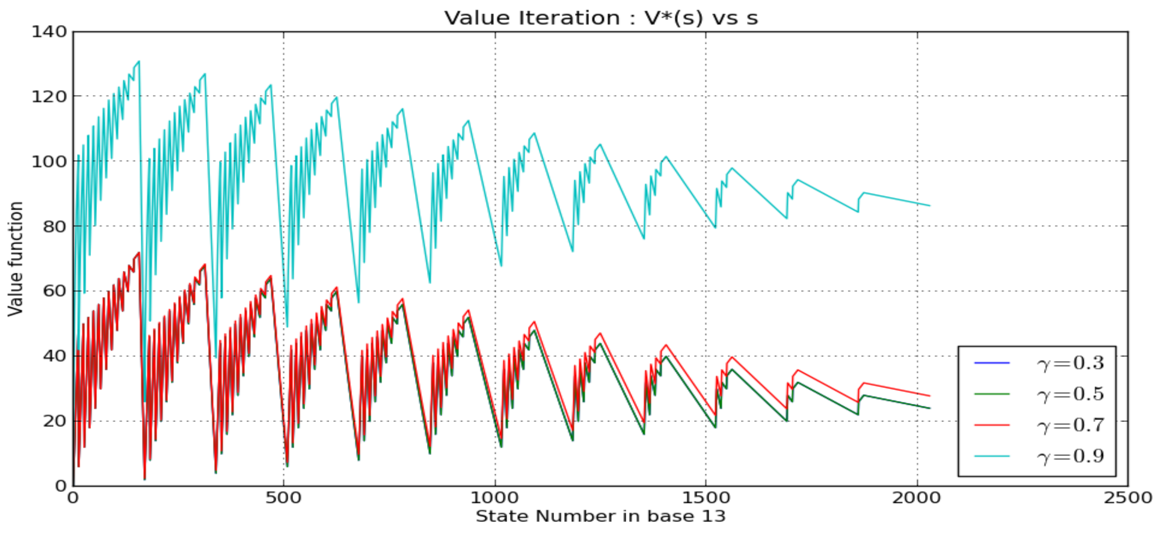
Table 1: Convergence Measure: $\langle Method, \gamma \rangle$ vs Number of Iterations

γ	P_i (P_s)	V_s
0.3	1(2)	2
0.5	1(2)	2
0.7	2(52)	51
0.9	3(221)	142

P_i is the number of *Policy Improvement* iterations performed.
P_s is the number of *Policy Evaluation* sweeps performed.
V_s is the number of *Value Iteration* sweeps performed.

Given below the optimal value function $V^*(s)$ plot of both the algorithms for different discount parameter γ.

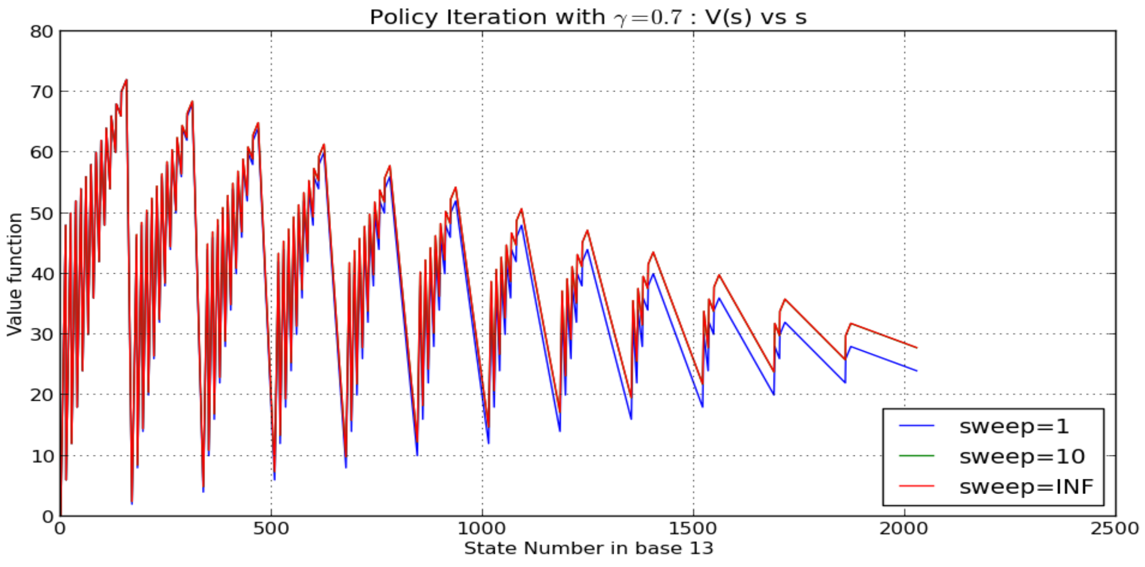


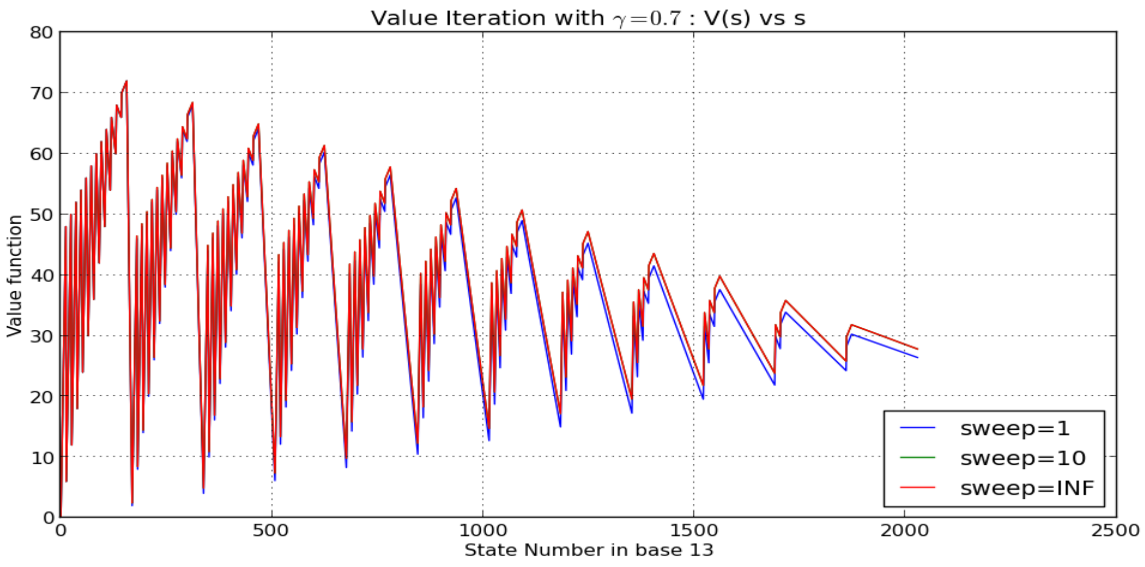


From the above two plots, it can be observed that,

- Given the discount parameter, both the methods converge to same optimal value function.
- For γ=0.3 and γ=0.5, same optimal value function is obtained in each of the methods.
- $V^*(s)$ increases with the γ value for both the algorithms.

Since both the algorithms converge to the optimal value function V^* in single step for γ=0.3 and γ=0.5, their plots are ignored. Given below the value funtion $V^k(s)$ plot after certain number of sweeps in both the algorithms when γ is set to 0.7.

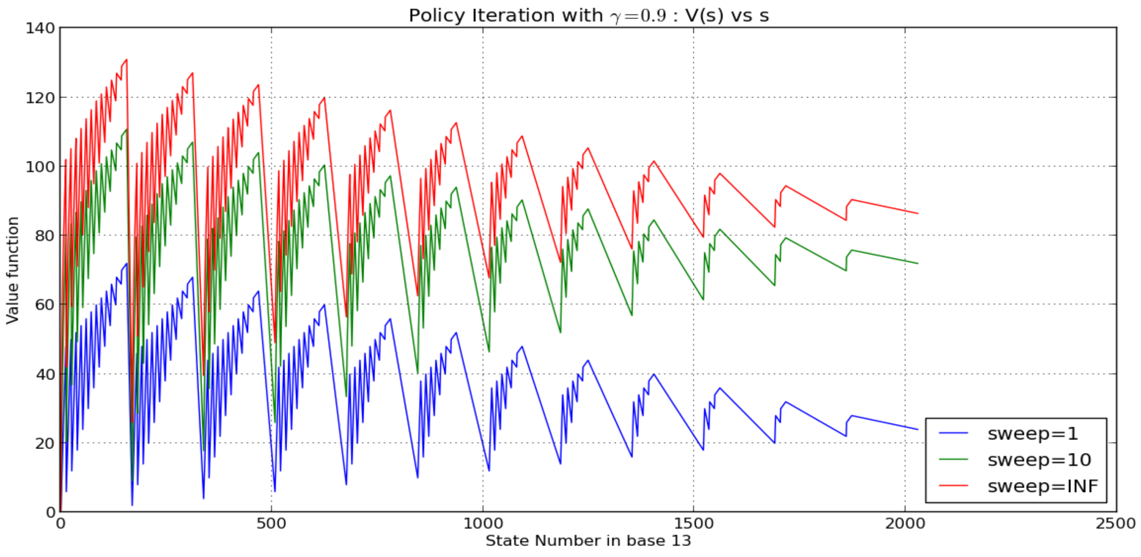


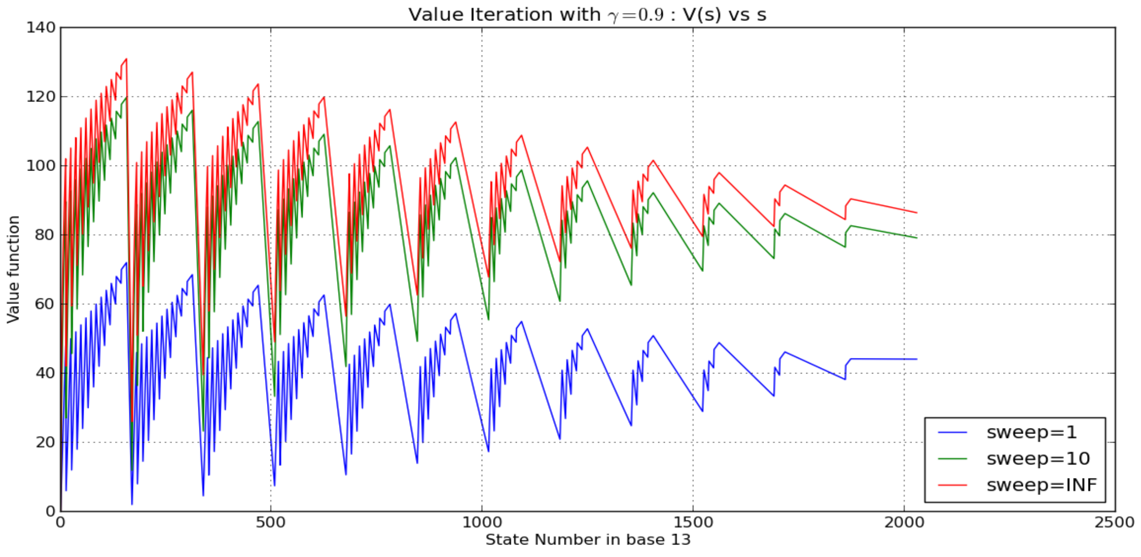


From the above two plots, it can be observed that,

- $V^k(s)$ increases with the increase in number of sweeps k.
- $V^{10}(s) \approx V^*(s)$ in both algorithms.

Given below the value funtion $V^k(s)$ plot after certain number of sweeps in both the algorithms when γ is set to 0.9.





From the above two plots, it can be observed that,

- $V^k(s)$ increases with the increase in number of sweeps k.
- Number of sweeps taken to converge to optimal value function increases with γ value.

1.6 Sample Results

The optimal next action for the following states in computed using both the algorithms. And it is observed that both the algorithms converge to the same optimal next action. The following tables enumerates it.

Table 2: Optimal next action

γ	$State : \langle 4,7,1\rangle)$	$State : \langle 1,3,6\rangle)$	$State : \langle 9,2,1\rangle)$
0.3	$\langle 4,7,1\rangle$	$\langle 1,3,6\rangle$	$\langle 9,2,1\rangle$
0.5	$\langle 4,7,1\rangle$	$\langle 1,3,6\rangle$	$\langle 9,2,1\rangle$
0.7	$\langle 0,7,1\rangle$	$\langle 0,3,6\rangle$	$\langle 0,2,1\rangle$
0.9	$\langle 0,3,1\rangle$	$\langle 0,0,2\rangle$	$\langle 0,1,1\rangle$

From the plots, it can be observed that γ=0.9 results in highest optimal value function. Hence, it is better the perform optimal next action as predicted by the algorithms when γ=0.9.

Also, from the above all plots, it can be clearly observed that there is exists a unique state with maximum optimal value. That state is $\langle 0,12,0\rangle$.

Bibliography

[1] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction.* Cambridge, MA: MIT Press, 1998.