MS&E 310 Project: Markov Decision Process

Maxwell Allman
mallman@stanford.edu

Jamie Kang
jamiekang@stanford.edu

December 11, 2017

1 Introduction

Markov Decision Processes (MDPs) can be formulated as the following primal and dual Linear Programming problems:

$$\begin{aligned} \min_{x} \quad & \sum_{j \in A_1} c_j x_j + \ldots + \sum_{j \in A_m} c_j x_j \\ \text{s.t.} \quad & \sum_{j \in A_1} (e_1 - \gamma p_j) x_j + \ldots + \sum_{j \in A_m} (e_m - \gamma p_j) x_j = e \qquad (1) \\ & x_j \geq 0, \forall j \qquad (2) \end{aligned}$$

$$\begin{aligned} \max_{y} \quad & e^T y = \sum_{i=1}^{m} y_i \\ \text{s.t.} \quad & y_1 - \gamma p_j^T y \leq c_j, j \in A_i \\ & \vdots \\ & y_i - \gamma p_j^T y \leq c_j, j \in A_i \qquad (3) \\ & \vdots \\ & y_m - \gamma p_j^T y \leq c_j, j \in A_m \end{aligned}$$

where m: number of states, A_i: set of all actions available in state i, p_j: state transition probabilities from state i to all states, c_j: immediate cost when action j is taken, γ: discount factor between 0 and 1, x_j: state-action frequency.

In the following sections, we prove some useful theoretical properties and examine the convergence rates of several Value Iteration variant methods.

2 Theoretical Results

Question 1. *Prove that in MDP LP formulation, every basic feasible solution represent a policy, i.e., the basic variables have exactly one variable from each state i. Furthermore, prove each basic variable value is no less than 1, and the sum of all basic variable values is* $\frac{m}{1-\gamma}$.

Proof. Since the LP has m constraints, each BFS has at most m non-zero values. Now, the ith constraint gives

$$\sum_{j \in A_i} x_j - \sum_{j \in A_1 \cup \ldots \cup A_m} \gamma (p_j x_j)_i = 1$$

Since $(p_j x_j)_i \geq 0$ for all j and i, this implies

$$\sum_{j \in A_i} x_j \geq 1$$

So, the basic variables must include at least one variable for each state i. But since there are at most m basic variables, the basic variables have exactly one variable from each state i. Thus, to satisfy $\sum_{j \in A_i} x_j \geq 1$, each basic variable must have value at least 1. Finally, the sum of the left hand sides of each of the m constraints is

$$S = \sum_{j \in A_1 \cup \ldots \cup A_m} (x_j - \sum_{i=1}^{m} (\gamma p_j x_j)_i) = \sum_{j \in A_1 \cup \ldots \cup A_m} (x_j - \gamma \sum_{i=1}^{m} (p_j)_i x_j))$$

But since the entries of p_j sum to 1, we get $\sum_{i=1}^{m} (p_j)_i = 1$, so

$$S = \sum_{j \in A_1 \cup \ldots \cup A_m} x_j - \gamma x_j = (1-\gamma) \sum_{j \in A_1 \cup \ldots \cup A_m} x_j$$

The sum of the left hand sides of the m constraints must equal the sum of the right hand sides, so

$$(1-\gamma) \sum_{j \in A_1 \cup \ldots \cup A_m} x_j = m$$

$$\implies \sum_{j \in A_1 \cup \ldots \cup A_m} x_j = \frac{m}{1-\gamma}$$

Since $\sum_{j \in A_1 \cup \ldots \cup A_m} x_j$ is the sum of all the basic variable values, the sum of all the basic variable values is $\frac{m}{1-\gamma}$. □

Question 2. *Prove the contraction result of the Value Iteration method.*

Proof. In order to prove $||y^{k+1} - y^*||_\infty \leq \gamma ||y^k - y^*||_\infty$, we first rewrite the LHS as:

$$\max_i \{ | \min_j (c_j + \gamma p_j^T y^k) - \min_j (c_j + \gamma p_j^T y^*) | \}$$

which is equivalent to

$$\max_i \{ | - \max_j (-c_j - \gamma p_j^T y^k) + \max_j (-c_j - \gamma p_j^T y^*) | \}.$$

Reorganizing the two terms we have

$$\max_i\{|\max_j(-c_j-\gamma p_j^T y^*)-\max_j(-c_j-\gamma p_j^T y^k)|\} \le \max_i\{|\max_j(-c_j-\gamma p_j^T y^*-(-c_j-\gamma p_j^T y^k))|\}.$$

This is due to: $\max f(x) - \max g(x) \le \max f(x) - g(x) \le \max(f(x) - g(x))$. Furthermore, the c_j terms cancel out each other, and thus we can simplify the above expression to

$$\begin{aligned}
\max_i\{|\max_j(\gamma p_j^T y^k - \gamma p_j^T y^*)|\} &= \gamma \max_i\{|\max_j(p_j^T y^k - p_j^T y^*)|\} \\
&\le \gamma \max_i\{\max_j |p_j^T y^k - p_j^T y^*|\} \\
&= \gamma \max_i |y^k - y^*| \max_j p_j^T \\
&\le \gamma \max_i |y^k - y^*| \\
&= \gamma ||y^k - y^*||_\infty
\end{aligned}$$

where the second last inequality is due to p_j being some proability measure, and thus $\max_j p_j^T \le 1$. □

Question 3. *Assuming that $y^0 \ge y^*$ and $y^0 \ge y^1$, prove the entry-wise monotone property: $y^* \le y^{k+1} \le y^k, \forall k$.*

Proof. For the statement of this problem to be true, we must assume that entry-wise, $y^1 \le y^0$. Now, as an induction hypothesis, suppose that entry-wise $y^k \le y^{k-1}$ for some k. Then

$$y_i^{k+1} = \min_{j\in A_i}\{c_j + \gamma p_j^T y^k\}$$

but since $y^k \le y^{k-1}$, and for all j the entries of p_j are non-negative, we have

$$p_j^T y^k \le p_j^T y^{k-1}$$

for all j. Thus,

$$y_i^{k+1} = \min_{j\in A_i}\{c_j + \gamma p_j^T y^k\} \le \min_{j\in A_i}\{c_j + \gamma p_j^T y^{k-1}\} = y_i^k$$

So, $y^{k+1} \le y^k$ entry-wise, and since $y^1 \le y^0$ entry-wise by assumption, we get by induction that $y^{k+1} \le y^k$ for all k.

Now, assume $y^k \ge y^*$ entry-wise. Then as before, for all j,

$$p_j^T y^* \le p_j^T y^k$$

so

$$y_i^{k+1} = \min_{j\in A_i}\{c_j + \gamma p_j^T y^k\} \ge \min_{j\in A_i}\{c_j + \gamma p_j^T y^*\} = y_i^*$$

Thus, $y^{k+1} \ge y^*$ entry-wise, and since $y^0 \ge y^*$, we have by induction that $y^k \ge y^*$ for all k. □

3 Value Iteration variants and numerical experiments

In this section, we introduce several variants of the Value Iteration method that may potentially improve the algorithm's convergence rate. We then perform some numerical experiments using MATLAB. Markov chains are randomly generated for the pre-specified parameters – number of state, number of actions per state, and number of next states an action result in. Namely, parameters such as probability matrix P, state-action pair matrix, cost $C(i, i', j)$(and thus action cost in each state i $c_j = \sum_j P(i, i', j) * C(i, i', j)$) are all randomly generated. For implementation details, please refer to our code. In order to obtain more accurate approximations, for each set of pre-specified parameters we perform Monte Carlo Simulation over several Markov chains.

Question 4.1. Random VI: *Rather than go through all state values in each iteration, we modify the VI method, such that in the kth iteration, we randomly select and update a subset of states B^k in each iteration.*

For this question and the next two questions, we wrote a program that generates a random Markov chain, with a given number states and actions per state. We will call the standard VI update method by the name "StandardVI." We compared the convergence rate of the RandomVI method with the StandardVI method by:

1. Generating a random Markov chain
2. Computing the fixed point y^* up to a very small tolerance by running StandardVI until subsequent updates only differ by the very small tolerance
3. Computing the convergence rate to y^* of StandardVI and RandomVI (for a given batch size), both in terms of number of iterations, and number of action costs computed (that is, the number of times $c_j + \gamma p_j^T y^k$ had to be computed).

We performed these three steps over a range of batch sizes, and we took an average of the convergence rates over several randomly generated Markov chains via Monte Carlo Simulation. The reason we compared the number of action costs computed until convergence, as well as the number of iterations, is that each iteration of RandomVI takes less computation time than an iteration of StandardVI, for batch sizes smaller than the total number of states. Furthermore, we believed the number of action costs computed was more representative of computational time than the total time our program took to converge, because the total time will be affected by our program implementation.

As depicted in Figures 1 and 2 we found that as the batch sizes become smaller, the number of iterations until convergence for RandomVI became much larger than for StandardVI, but over the range of batch sizes, the number of action costs computed for both methods remains very similar. We include plots of number of iterations until convergence and number of action costs computed until convergence, for RandomVI and StandardVI, with random Markov chains with 30 states and 20 actions per state. We also ran experiments over Markov chains with a wide range of numbers of states and actions per state, and observed the same behavior.

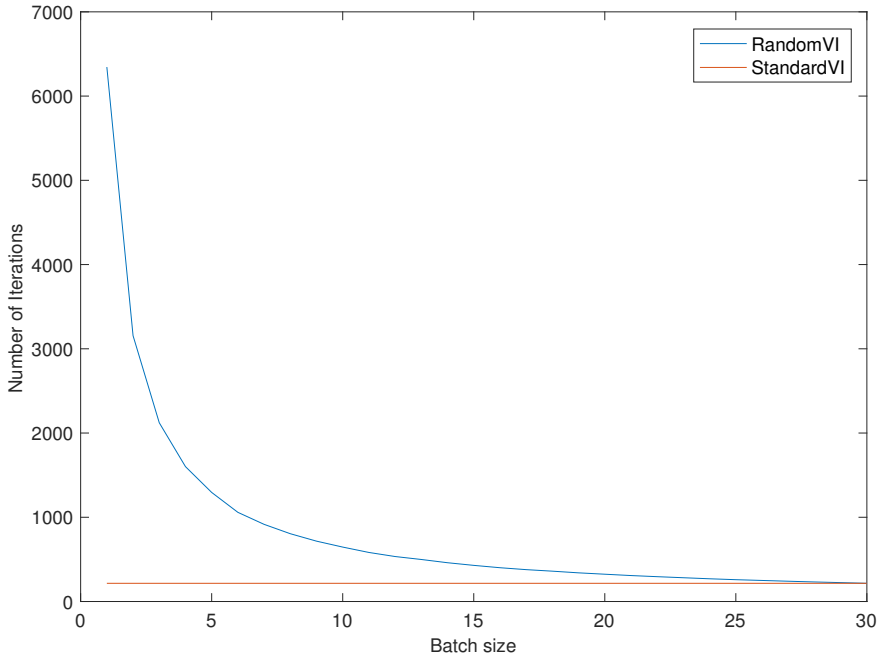

Figure 1: RandomVI vs StandardVI, 30 states and 20 actions: total number of iterations.

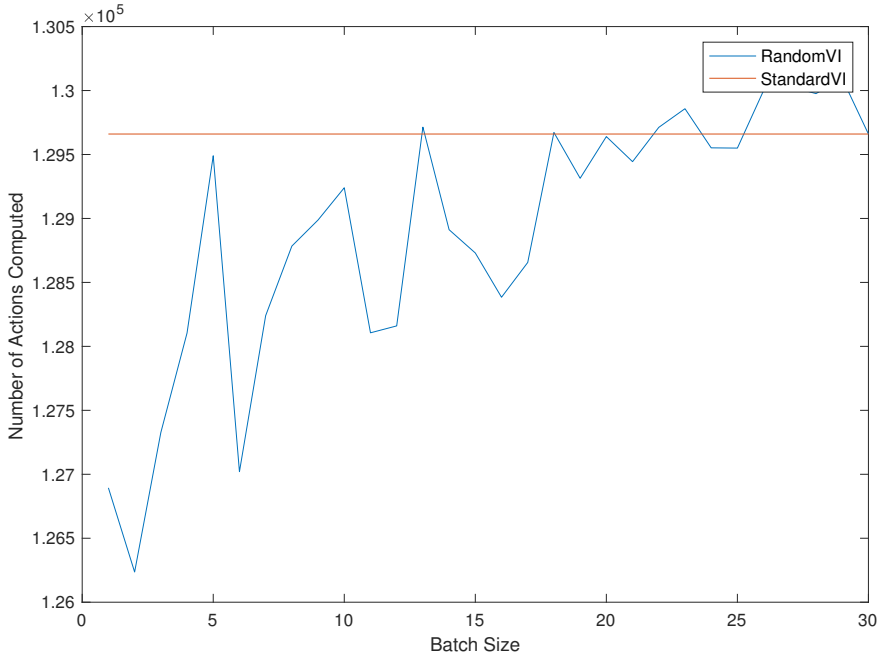

Figure 2: RandomVI vs StandardVI, 30 states and 20 actions: total number of computations.

Question 4.2. Empirical VI: *Suppose we build an empirical distribution for each action being selected as the winning action in the final policy: the probability of action j is the past frequency of action j is being selected as the arg min in the previous iterations, e.g., the the Bayes update where we start with a uniform distribution $\bar{p}^0$. Redo the computational experiments by randomly selecting B^k using the empirical distribution.*

We performed the same tests on this method as we did for RandomVI, but instead of varying the number of states that are updated in each iteration, we varied the number of actions that were sampled for each state in each iteration. In each iteration, we would sample actions using an updated empirical distribution $\sim$ multinomial (sample size, $\bar{p}$). When the action sample size was too small (less than 4), occasionally EmpiricalVI never seemed to converge, but other time it converged relatively quickly.

As shown in Figures 3 and 4 we found that for sample sizes above a certain threshold, the number of iterations required for EmpiricalVI and StandardVI are very similar. So, after this threshold, the number of action costs computed grows approximately linearly with the sample size, and the number of action costs computed is minimized for the sample size approximately the threshold value. As opposed to RandomVI, with the optimal sample size EmpiricalVI is significantly faster than StandardVI. We include plots of number of iterations until convergence and number of action costs computed until convergence, for EmpiricalVI and StandardVI, with random Markov chains with 30 states and 20 actions per state. Again, we also ran experiments over Markov chains with different numbers of states and actions per state, and observed the same behavior.

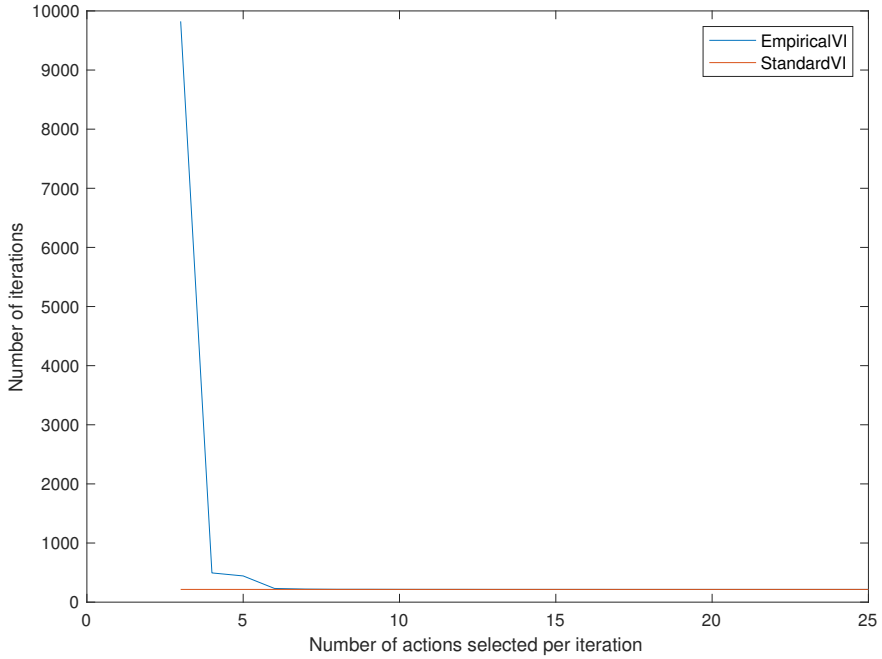

Figure 3: EmpiricalVI vs StandardVI, 30 states and 20 actions: total number of iterations.

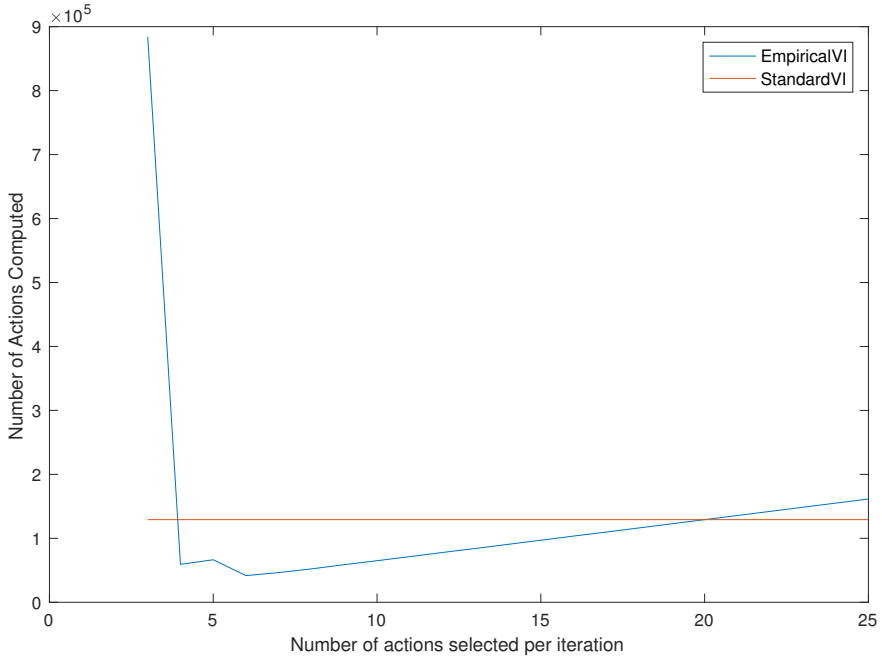

Figure 4: EmpiricalVI vs StandardVI, 30 states and 20 actions: total number of computations.

Question 5,6. Cyclic VI: *In the CyclicVI method, as soon as a state value is updated, we use it to update the rest of state values.* **RPCyclic VI:** *In the RP (Randomly Permuted) CyclicVI method, rather than with the fixed cycle order from 1 to m, we follow a random permutation order.*

We compared the convergence rates of StandardVI, RandomVI, EmpiricalVI, CyclicVI, and RPCyclicVI, in the same way as we did for RandomVI, over randomly generated Markov chains of varying numbers of states and actions per state. For the RandomVI method, the batch was fixed at 10, because the performance of this method does not appear to significantly vary with the batch size. For the EmpiricalVI method, for each generated Markov chain, we computed the optimal sample size through Monte Carlo Simulation (the sample size that had the fewest action cost computations until convergence), and recorded the number of action cost computations for this optimal sample size. We tested the methods on Markov chains with 30 states 10 to 30 actions per state, and on Markov chains with 20 actions per state and 10 to 30 states. We plotted our results in Figures 5 and 6.

First of all, we found that both CyclicVI and RPCyclicVI displayed improvements in convergence rate. Our numerical experiment showed that both of them have convergence rates about twice faster than those of StandardVI and RandomVI. Moreover, over the range of parameters we tested, RPCyclicVI appeared to converge slightly faster than CyclicVI does, but the difference was not as significant.

Overall, the method with the best performance was EmpricalVI (with the optimal sample size chosen), followed by CyclicVI, closely followed by RPCyclicVI, and StandardVI and RandomVI performed poorly. While EmpiricalVI had the best performance with the optimal sample size, in practice it may be computationally difficult to determine the optimal sample size. The performance of EmpiricalVI becomes better compared to the other methods as the number of actions per state grows, because only a small subset of the actions are sampled.

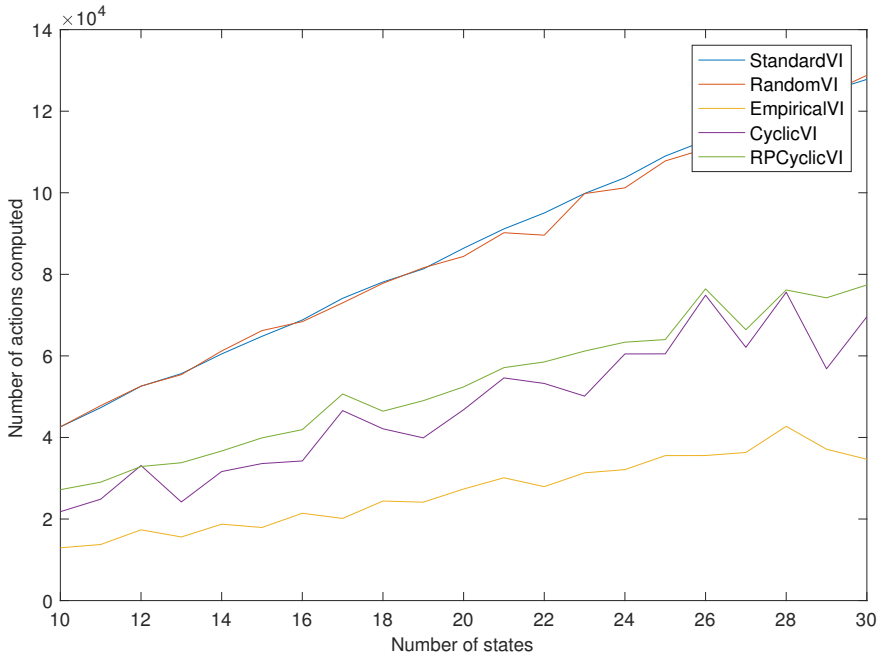

Figure 5: Comparison of Five VI Methods: total number of iterations.

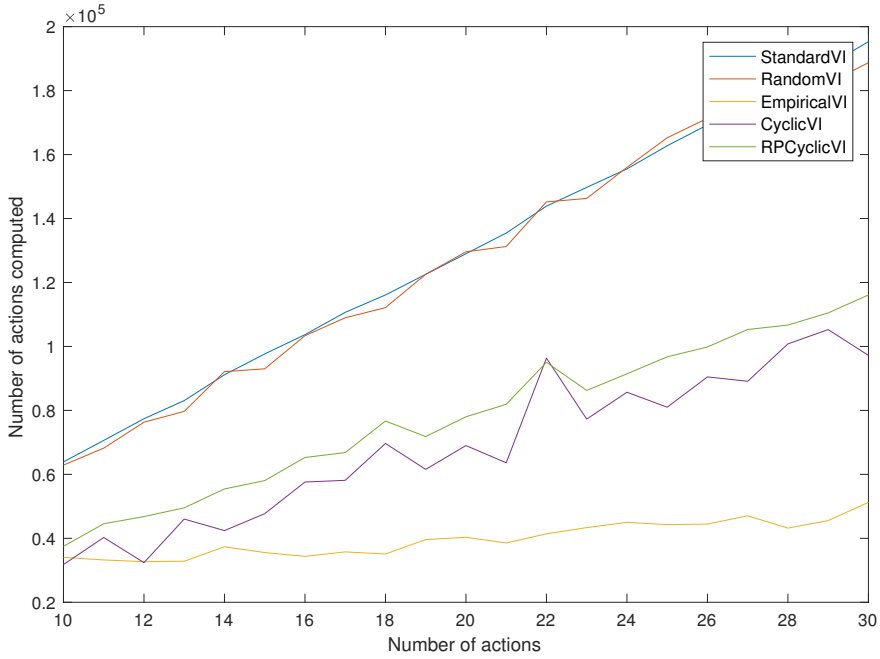

Figure 6: Comparison of Five VI Methods: total number of computations.

4 Remarks

In the previous section, we have numerically shown the impressive convergence rate improvements of EmpiricalVI, CyclicVI and RPCyclicVI. One possible future research direction could

be to theoretically prove these results. Moreover, introducing a faster algorithm to find the optimal sample size in EmpiricalVI will further enhance its convergence rate. On the other hand, RandomVI failed to display any significant improvement even when tested with various batch sizes. Investigating theoretical reasonings behind this could another interesting research direction.