MS&E 310 Course Project II: Markov Decision Process

Nian Si
niansi@stanford.edu

Fan Zhang
fzh@stanford.edu

This Version: Saturday 2nd December, 2017

1 Introduction

Markov Decision Process (MDP) is a pervasive mathematical framework that models the optimal decision making process in complex dynamic systems. It is widely used to tackle problems such as dynamic programming and reinforcement learning. Among methodologies to solve MDP problem, value iteration method received much attention because of its simplicity and conceptual importance.

In this report we will analyze and implement six typical iterative algorithms for Markov decision process, i.e.

1. Value Iteration Method (VI)
2. Random Value Iteration Method (Random VI)
3. Random Value Iteration by Action Method(Random VIA)
4. Cyclic Value Iteration Method (Cyclic VI)
5. Randomly Permuted Cyclic Value Iteration Method (RPCyclic VI)
6. Adapted Random Value Iteration by Action Method (AdaRandom VIA)

Algorithm 1 to 5 is introduced in the course project proposal, while Algorithm 6 is proposed by ourselves as an improvement of Algorithm 3. The intuition behind Algorithm ~~1~~6 is that the empirical distribution of optimal actions will converge to an one point distribution, therefore it is unnecessary to keep a constant sample size.

The report is organized as follows. In section 2 we present the pseudocode of all the iterative methodology. In section 3, we provide some theoretical analysis, which ensures the convergence of value iteration method. In section 4, we present the result of numerical experiment that concerning with the convergence rate of our algorithms using simulated data.

2 Algorithms

In this section, we shall present the pseudocode of all the methods of interest in this report.

Algorithm 1 Value Iteration Method (VI)

```
1: Input: initial value $y \in \mathbb{R}^n$, error bound Error = 1000, cost $c_j$ and transition probability $p_j$.
2: while Error > $10^{-8}$ do
3:     For all $i = \overline{1,n}$, update the value
           $$y_i \leftarrow \min_{j \in \mathcal{A}_i} \{c_j + \gamma p_j^T y\}.$$
       in a parallel way.
4:     Update the Error as the maximum elementwise change in $y_i$.
5: Output: $y$.
```

Algorithm 2 Random Value Iteration Method (Random VI)

```
1: Input: initial value $y \in \mathbb{R}^n$, random sample size $w$, error bound EPS = 1000, cost $c_j$ and transition probability $p_j$.
2: while Error > $10^{-8}$ do
3:     Sample a subset of state $B$ of size $w$.
4:     For all $i \in B$, update the value
           $$y_i^{k+1} \leftarrow \min_{j \in \mathcal{A}_i} \{c_j + \gamma p_j^T y^k\}$$
       in a parallel way.
5:     Update the Error as the maximum elementwise change in $y_i$.
6: Output: $y$.
```

Algorithm 3 Random Value Iteration by Action Method (Random VIA)

```
1: Input: initial value $y \in \mathbb{R}^n$, random sample size $w$, error bound ERROR = 1000, the frequency vector $f_j^{(i)} = 1$ for all $i = \overline{1,n}$, $j = \overline{1,m}$, cost $c_j$ and transition probability $p_j$.
2: while Error > $10^{-8}$ do
3:     for $i = \overline{1,n}$ do
4:         Sample a subset of actions $\mathcal{A} \subset \mathcal{A}_i$ of size $w$, with probability proportional to frequency $f^{(i)}$.
5:         For all $i = \overline{1,n}$, update the value
               $$y_i^{k+1} \leftarrow \min_{j \in \mathcal{A}} \{c_j + \gamma p_j^T y^k\},$$
           in a parallel way, record $j^*$ as the index of the optimal strategy.
6:         Set $f_{j^*}^{(i)} \leftarrow f_{j^*}^{(i)} + 1$.
7:         Update the Error as the maximum elementwise change in $y_i$.
8: Output: $y$.
```

Algorithm 4 Cyclic Value Iteration Method (Cyclic VI)

```
1: Input: initial value $y \in \mathbb{R}^n$, error bound Error = 1000, cost $c_j$ and transition probability $p_j$.
2: while Error > $10^{-8}$ do
3:     for $i = \overline{1,n}$ do
4:         $$y_i \leftarrow \min_{j \in \mathcal{A}_i} \{c_j + \gamma p_j^T y\}.$$
5:     Update the Error as the maximum elementwise change in $y_i$.
6: Output: $y$.
```

Algorithm 5 Randomly Permuted Cyclic Value Iteration Method (RPCyclic VI)

```
1: Input: initial value $y \in \mathbb{R}^n$, error bound Error = 1000, cost $c_j$ and transition probability $p_j$.
2: while Error > $10^{-8}$ do
3:     Sample $\{k_i; i = \overline{1,n}\}$ as a random permutation of $\overline{1,n}$.
4:     for $i = \overline{1,n}$ do
5:         $$y_{k_i} \leftarrow \min_{j \in \mathcal{A}_{k_i}} \{c_j + \gamma p_j^T y\}.$$
6:     Update the Error as the maximum elementwise change in $y_i$.
7: Output: $y$.
```

Algorithm 6 Adapted Random Value Iteration by Action Method (AdaRandom VIA)

```
1: Input: initial value $y \in \mathbb{R}^n$, random sample size $w$, sample size updating rate $\gamma$, sample size
   lower bound $w_{min}$, error bound EPS = 1000, the frequency vector $f_j^{(i)} = 1$ for all $i = \overline{1,n}$, $j = \overline{1,m}$,
   cost $c_j$ and transition probability $p_j$.
2: while Error > $10^{-8}$ do
3:     for $i = \overline{1,n}$ do
4:         Sample a subset of actions $\mathcal{A} \subset \mathcal{A}_i$ of size $w$, with probability proportional to frequency
   $f^{(i)}$.
5:         For all $i \in B$, update the value
           $$y_i^{k+1} \leftarrow \min_{j \in \mathcal{A}} \{c_j + \gamma p_j^T y^k\},$$
   in a parallel way, record $j^*$ as the index of the optimal strategy.
6:         Set $f_{j^*}^{(i)} \leftarrow f_{j^*}^{(i)} + 1$.
7:         Update the Error as the maximum elementwise change in $y_i$.
8:     if $w > w_{min}$ then
9:         Update $w = \gamma \times w$.
10: Output: $y$.
```

3 Theoretical results

In this section, we present the theoretical analysis for different value iteration schemes. In order to analyze the residual error of value iteration method, we need a procedure to produce the accurate value of the stationary solutions to value iteration problem. In Theorem 1, we shall prove that there exist a linear programming problem whose optimal solution represents the optimal stationary policy to original MDP problem. Therefore, solving the dual linear programing problem yields the stationary solution to value iteration schemes.

Theorem 1. *For the minimization problem*

$$\begin{aligned}
&\min_x \sum_{j\in\mathcal{A}_1} c_j x_j + \cdots + \sum_{j\in\mathcal{A}_m} c_j x_j \\
&s.t. \sum_{j\in\mathcal{A}_1} (\mathbf{e}_1 - \gamma\mathbf{p}_j)\, x_j + \cdots + \sum_{j\in\mathcal{A}_m} (\mathbf{e}_m - \gamma\mathbf{p}_j)\, x_j = \mathbf{e}, \\
&x_j \geq 0, \forall j,
\end{aligned}$$

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

Proof: By adding up all the quality, we have

$$\mathbf{e}^T x = \frac{m}{1-\gamma}.$$

Therefore, the sum of all basic variable value is $\frac{m}{1-\gamma}$.

Then, we will show a BFS represents a policy. Let B denote the index set of basic variables, then $|B| = m$. Suppose B does not contain any state-action pair for a certain state k, then the $k-$th equality constraint fails to hold:

$$\sum_{j\in\mathcal{A}_k} x_j = 0 \neq 1 + \gamma \sum_{i\in S} \sum_{j\in\mathcal{A}_i} x_j p_{ji} \geq 1.$$

Therefore, B contains exactly one state-action pair for each state, and represents a policy of the discounted MDP problem.

Actually, for a BFS x^π,

$$\sum_{j\in\mathcal{A}_k} x_j = x_k^\pi = 1 + \gamma \sum_{i\in S} \sum_{j\in\mathcal{A}_i} x_j p_{ji} \geq 1.$$

$\square$

In Lemma 1, we prove that the value iteration scheme is a contraction mapping with Lipchitiz constant γ, where $0 < \gamma < 1$.

Lemma 1. *For the Value Iteration method: starting with any vector* y_0*, then iteratively update it*

$$y_i^{k+1} = \min_{j\in\mathcal{A}_i} \{c_j + \gamma p_j^T y_k\}, \forall i.$$

The contraction result holds:

$$\left\|y^{k+1} - y^*\right\|_\infty \le \gamma \left\|y^k - y^*\right\|_\infty, \forall k,$$

where y^* is the fixed-point or optimal value vector, that is,

$$y_i^* = \min_{j\in\mathcal{A}_i}\{c_j + \gamma \mathbf{p}_j^T y^*\}, \forall i.$$

Proof: Let j_i be the minimizer that

$$y_i^* = c_{j_i} + \gamma \mathbf{p}_{j_i}^T y^*. \tag{3.1}$$

Then, according to iteration formula, we have

$$y_i^{k+1} = \min_{j\in\mathcal{A}_i}\{c_j + \gamma p_j^T y^k\} \le c_{j_i} + \gamma \mathbf{p}_{j_i}^T y^k. \tag{3.2}$$

By taking difference of (3.1) and (3.2), we have

$$y_i^{k+1} - y_i^* \le \gamma \mathbf{p}_{j_i}^T y^k - \gamma \mathbf{p}_{j_i}^T y^* \le \gamma \left\|y^k - y^*\right\|_\infty.$$

Let j_i^k be the minimizer that

$$y_i^{k+1} = c_{j_i^k} + \gamma \mathbf{p}_{j_i^k}^T y^k. \tag{3.3}$$

Then according to fixed-point formula, we have

$$y_i^* = \min_{j\in\mathcal{A}_i}\{c_j + \gamma \mathbf{p}_j^T y^*\} \le c_{j_i^k} + \gamma \mathbf{p}_{j_i^k}^T y^*. \tag{3.4}$$

By taking difference of (3.3) and (3.4), we have

$$y_i^* - y_i^{k+1} \le \gamma \mathbf{p}_{j_i^k}^T y^* - \gamma \mathbf{p}_{j_i^k}^T y^k \le \gamma \left\|y^k - y^*\right\|_\infty.$$

Therefore,

$$\begin{aligned} |y_i^{k+1} - y_i^*| &\le \gamma \left\|y^k - y^*\right\|_\infty, \forall i, \\ \left\|y^{k+1} - y^*\right\|_\infty &\le \gamma \left\|y^k - y^*\right\|_\infty. \end{aligned}$$

□

Since $|y_i^{k+1} - y_i^*| \le \gamma \left\|y^k - y^*\right\|_\infty$ is entry-wise true, after update all the element of y all around, the resulting composite mapping is a contraction mapping with Lipchitz constant γ as well. Therefore, we have following theorem, which guarantees the value iteration scheme (VI) has expontial convergence rate. The convergence rate of Cyclic VI and PRCyclic VI can also be proved in a similar fashion.

Theorem 2. *VI, Cyclic VI and PRCyclic VI are have exponential convergence rate; Random VI is convergent with probability 1 if each state has a positive probability to be selected.*

Proof: The proof simply follows the result of classical contraction mapping principle. See, for example, Istrc et al. (1981). □

Lemma 2 in the following proves that the value iteration scheme (VI) is actually monotone.

Lemma 2. *In the VI method, if we have $y^{(1)} \geq y^{(2)}$, then we have the following entry-wise property:*

$$T(y^{(1)}) \geq T(y^{(2)}),$$

where

$$T(y)_i = \min_{j \in \mathcal{A}_i} \{c_j + \gamma \mathbf{p}_j^T y\}, \forall i.$$

Furthermore, In the VI method, if starting with any vector $y^0 \geq y^*$ and $y^0 \geq y^1$, then we have the following entry-wise monotone property:

$$y^* \leq y^{k+1} \leq y^k, \forall k.$$

Proof:

$$\begin{aligned} c_j + \gamma \mathbf{p}_j^T y^{(1)} &\geq c_j + \gamma \mathbf{p}_j^T y^{(2)}, \forall j \in \mathcal{A}_i, \forall i, \\ \min_{j \in \mathcal{A}_i} \{c_j + \gamma \mathbf{p}_j^T y^{(1)}\} &\geq \min_{j \in \mathcal{A}_i} \{c_j + \gamma \mathbf{p}_j^T y^{(2)}\}, \forall i \\ T(y^{(1)}) &\geq T(y^{(2)}). \end{aligned}$$

□

We now present a result about the uniform convergence rate in the following theorem, which implies all the element in vector y_i^k actually shares the same convergence rate as $k \to \infty$. This is an interesting phenomenon that we discovered in the numerical experiment at the beginning, so afterwards we proposed a conjecture that is eventually justified.

Theorem 3. *Assume that every state has only 1 optimal action, i.e.*

$$y_i^* > c_i + \gamma \mathbf{p}_j^T y^*,$$

for $j \in \mathcal{A}_i$ and $j \neq \arg\min_j (c_i + \gamma \mathbf{p}_j^T y^*)$. If Markov chain is primitive (or equivalently, irreducible and aperiodic), when every state chooses the stationary optimal action. Then, In the VI method, we have

$$\lim_{k \to \infty} \frac{\min_i |y_i^k - y_i^*|}{\max_i |y_i^k - y_i^*|} = 1,$$

if $\max_i |y_i^k - y_i^*| > 0$.

Proof: First let we consider the Markov chain case, in which every state only has one action.

$$y_i^{k+1} = c_i + \gamma \mathbf{p}_i^T y^k,$$

$$y_i^* = c_i + \gamma \mathbf{p}_i^T y^*.$$

By taking the difference of 2 equations, we have

$$y^{k+1} - y^* = \gamma \mathbf{p}_i^T \left(y^k - y^*\right).$$

In matrix form, we have

$$\begin{aligned} y^{k+1} - y^* &= \gamma P \left(y^k - y^*\right). \\ y^{n+1} - y^* &= \gamma^n P^n \left(y^1 - y^*\right). \end{aligned}$$

Because P is primitive, we have

$$P^n \to [\pi, \pi, \ldots, \pi]^T,$$

where π is the invariant distribution.

$$\lim_{k\to\infty} \frac{\min_i \left|y_i^k - y_i^*\right|}{\max_i \left|y_i^k - y_i^*\right|} = \frac{\left|\pi^T \left(y^1 - y^*\right)\right|}{\left|\pi^T \left(y^1 - y^*\right)\right|} = 1.$$

Now, we go back to the MDP problem. Suppose that

$$\begin{aligned} & c_{j_1} + \gamma \mathbf{p}_{j_1}^T y^* > c_{j_2} + \gamma \mathbf{p}_{j_2}^T y^*, \\ \Leftrightarrow \quad & c_{j_1} - c_{j_2} > \gamma \left(\mathbf{p}_{j_2} - \mathbf{p}_{j_1}\right)^T y^*. \end{aligned}$$

According to lemma 1, $\forall \epsilon$, $\exists N$, if $k > N$, we have $\left\|y^k - y^*\right\|_\infty < \epsilon$,

$$\gamma \left(\mathbf{p}_{j_2} - \mathbf{p}_{j_1}\right)^T y^k < \gamma \left(\mathbf{p}_{j_2} - \mathbf{p}_{j_1}\right)^T y^* + \gamma\epsilon \left\|\mathbf{p}_{j_2} - \mathbf{p}_{j_1}\right\|_1,$$

by taking ϵ enough small, we have

$$\begin{aligned} \gamma \left(\mathbf{p}_{j_2} - \mathbf{p}_{j_1}\right)^T y^k &< c_{j_1} - c_{j_2}, \\ c_{j_1} + \gamma \mathbf{p}_{j_1}^T y^k &> c_{j_2} + \gamma \mathbf{p}_{j_2}^T y^k. \end{aligned}$$

Therefore, after enough steps, we will only choose the optimal action in the MDP problem, which reduces to the Markov chain case. □

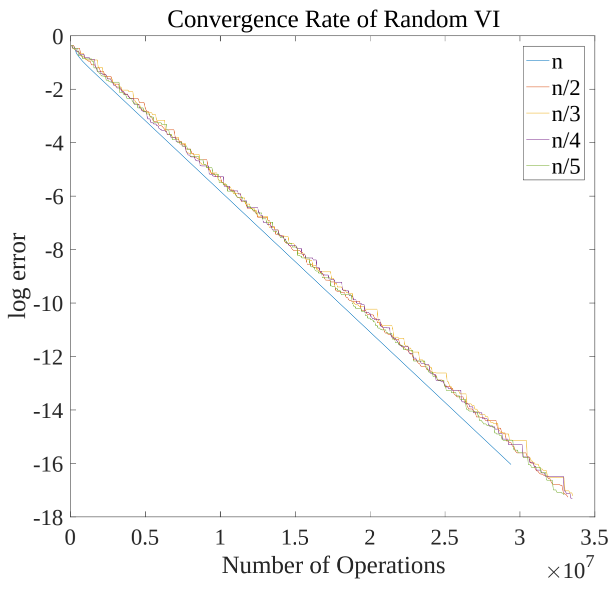
4 Numerical results

We present the numerical results in following three aspects. First, in Section 4.1, we test the influence of different sample sizes on convergence rate; Then, in Section 4.2 we compare the the convergence rate among different algorithms; Finally, in Section 4.3, we test the running time of different algorithms in terms different state size (n), action size (m) and the number of non-zero in each action.

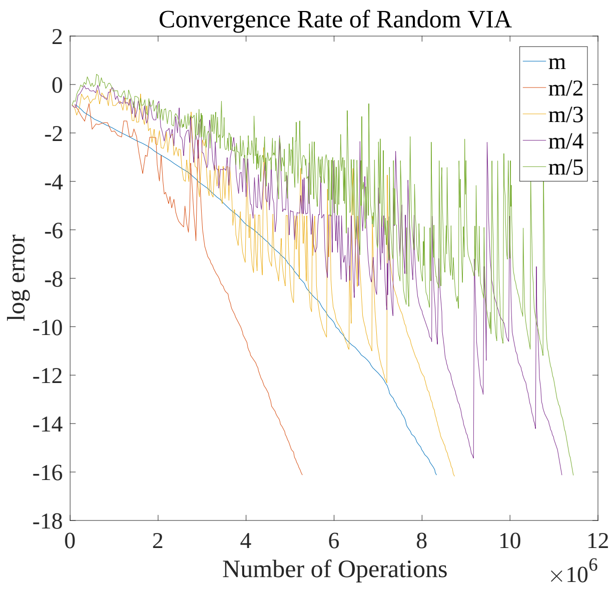
Throughout this section, we set the discount factor in MDP $\gamma = 0.9$. The error is measured by the distance generated by infinity norm between the vector and benchmark produced linear programming.

4.1 Sample size

In this section, we test the performance of sample size w for Random VI and Random VIA algorithm (Algorithm 2 and 3). We fix the state size $n = 100$, for each state we have $m = 20$ actions, and the transition probability p_j is a sparse vector with at most $nz = 5$ nonzero elements. We change the sample size w in both Random VI and Random VIA algorithm. The plot of log-error versus number of operations for different sample size is given in Figure 1, where number of operations is defined by $|B|nm \times step$ for Random VI and $|\mathcal{A}|n^2 \times step$ for Random VIA.



(a) Convergence Rate of Random VI (Algorithm 2)



(b) Convergence Rate of Random VIA (Algorithm 3)

Figure 1: Convergence plot of different sample sizes

In Figure 1(a), we learned that the convergence rate of Random VI is insensitive to sample size and if sample size is n which reduces to VI algorithm, there is no randomness and the convergence line does not fluctuate. Figure 1(b) reveals that the sample size $w = m/2$ yields relatively better performance for Random VIA algorithm. Too small sample size in Random VIA algorithm results in the problem of instability.

4.2 Convergence rate of Algorithms

We fix the state size $n = 100$, for each state we have $m = 20$ actions, and the transition probability p_j is a sparse vector with at most $nz = 5$ nonzero elements. We fix the sample size to be $n/2$ or

$m/2$ when implementing random algorithm. The we test the performance of all 6 algorithms given in Section 2. The convergence rate of different algorithms is depicted in Figure 2.

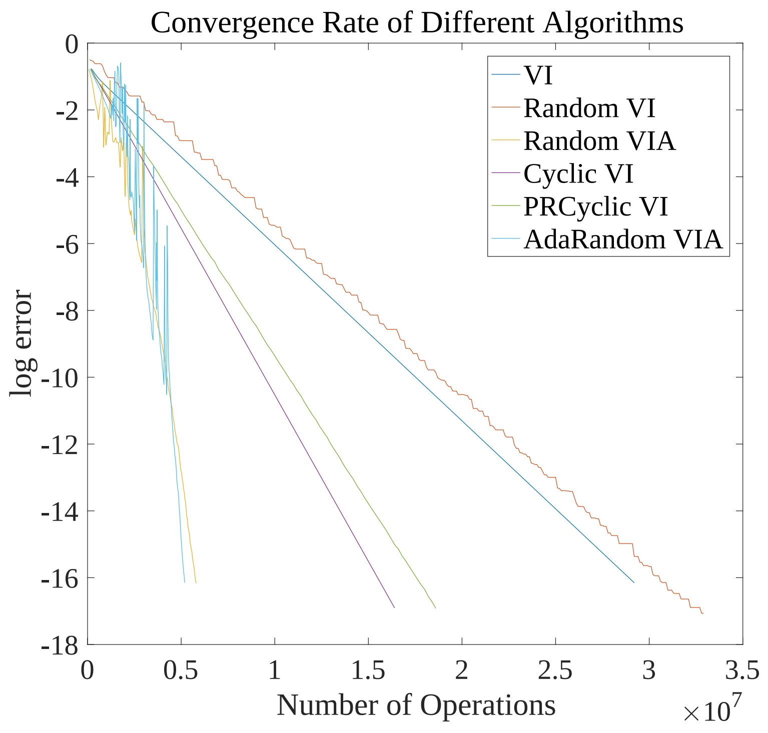


Figure 2: Convergence rate of Algorithms

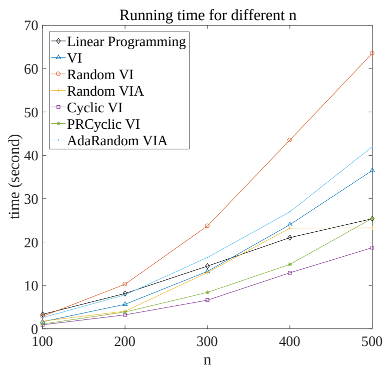
Learning from Figure 2, we want to highlight that our algorithm AdaRandom VIA (Algorithm 6) outperforms all the algorithm provided in the project proposal in terms of number of operations. However, AdaRandom VIA (Algorithm 6) and Random VIA (Algorithm 3) fluctuate a lot at the beginning. Among five suggested algorithms, Random VIA (Algorithm 3) possess the fastest convergence rate. The result is presented in Figure 3.

4.3 Running time in terms different state size (n), action size (m) and the number of non-zero (nz)

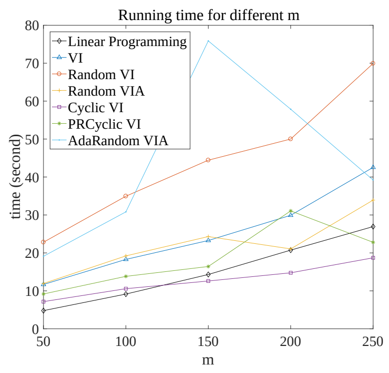
We set $n = 100 \sim 500, m = 250, nz = 40$ when testing tunning time for different size of states; $n = 500, m = 50 \sim 250, nz = 40$ when test running time for different size of actions; $n = 500, m = 250, nz = 40 \sim 200$ for different levels of sparsity. As a benchmark, we also compare the running time of solving linear programming by cvx package in Matlab.

From Figure 3(a) we learn that the running time has approximately quadratic growth rate when n grows, which perfectly matches our expectations. The running time of Random VIA slightly drops when n increases from 400 to 500, probably because of the instability of random algorithm.

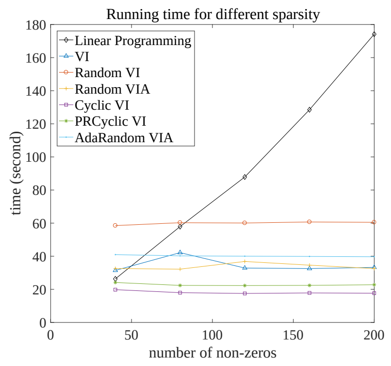
From Figure 3(b) we know the running time of value iterative scheme is approximately a linear function in m, except for our own algorithm AdaRandom VIA. The weird result of AdaRandom VIA can also be attributed to randomness and instability.



(a) Running time for different n



(b) Running time for different m



(c) Running time for different nz

Figure 3: Running time

From Figure 3(c) we know the efficiency of value iterative type algorithms is insensitive to sparsity. While the LP solver is sensitive to the sparsity structure. As a result, it is preferable to apply value iteration methodology to solve the SDP if the transition matrix is not sparse.

Among all the algorithm, Cyclic VI performs the best, followed by PRCyclic VI and Random VIA. It is probably because Cyclic VI use more information in each iteration. Random VI performs the worst. However, VI and Random VI are easily parallelled to improve performance.

We find the random algorithms spend more running time while using less numbers of operations, which is caused by the inefficiency of generating random number by Matlab.

5 Conclusion

In this report, we present 6 algorithms to solve MDP and give some theoretical results. Furthermore, we analyze the numerical performance compared to benchmark method linear programming. Here are our major finding,

1. All the algorithm have a exponential convergence rate.
2. Random VI is insensitive to the sample size, while Random VIA is sensitive to the sample size and sample size $w = m/2$ yields relatively better performance.
3. In terms of number of operations, Random VIA and his modified version AdaRandom VIA has the best performance.
4. In terms of running time, Cyclic VI and PRCyclic VI has the best performance.
5. The time of generating random number counts for a lot in total running time of random algorithms.

References

Istrc, V. et al. (1981). *Fixed point theory: an introduction*, volume 7. Springer.