Research on Average Reward Reinforcement Learning Algorithms

Dr. Yang Gao

National Laboratory for Novel Software Technology, Nanjing University

Nov. 5, 2006 **MLA06@Nanjing, China**

Outline

- **Case Study:** an Access-Control Queuing Task
- **Motivation and Background**
- **DP Methods for Average Rewards MDP**
- **RL Methods for Average Rewards MDP**
- **Our RL Method:** an Average Reward Reinforcement Learning Algorithm based on the Performance Potential of a Reference State
- **Future Works**

Case: an Access-Control Queuing Task

- *N* servers (*N*=10)
- Customers with four different priorities
  - The customers with priority 1, 2 and 4 arrives the queue server with the a probability of 0.2
  - The customers with priority 8 arrives with a probability of 0.4
  - When access to a server, the system gets a reward of 1, 2, 4, 8
- ***When a customer arrives at the queuing, servers decide to accept it or reject it?***

Motivation: why is not discounting

- Purpose of discounting
  - Such as economics, discounting can be used to represent "interest" earned on rewards, so that an action that generates an immediate reward will be preferred over one that generates the same reward some steps into the future.
  - In some sequence decision task, the goal can be transformed using the discounting method.

Sequence decision task

Background: Average Reward Markov Decision Processes

- MDP
  - S, A, $P_{xy}(a)$, $r(x,a)$
  - Glossary
    - Communicate, Recurrent
    - Transient: non-recurrent state

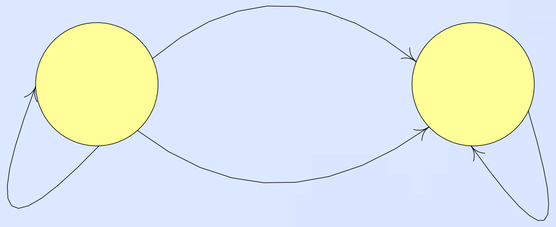
Any finite MDP must have recurrent states, since not all states can be transient.

If a recurrent state *x* communicates with another state *y*, then y has to be recurrent also.

MDP: glossary

- **Ergodic:** recurrent class of states.
- **Irreducible:** all states forms an ergodic class.
- **Period:** States in a given recurrence class all have the same period. / *aperiodic*
- **An *ergodic* or *recurrent* MDP**: The transition matrix corresponding to every policy has a single recurrent class.
- ***Unichain*:**
- ***Multichain*:** At least one policy whose transition matrix has two or more recurrent classes.

Example 1



•State **A** is transient under either policy(doing action **a1** or **a2** in state **A**).

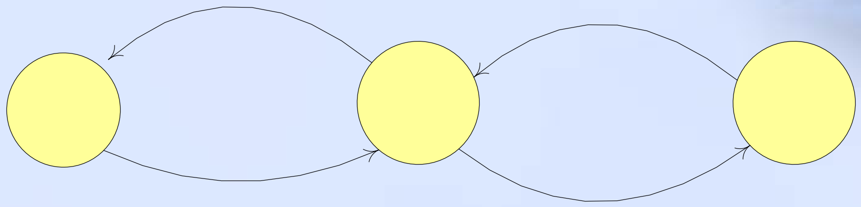
•State **B** is recurrent.

A

•Such recurrent single-state classes are often called *absorbing* states.

A1(5,0.5)

Example 2



•If action **a1** is taken in state **A**, the recurrent class is formed by **A** and **B**, and is periodic with period **2**.

A1(2,1)

•If action **a2** is taken in state **A**, the recurrent class is formed by **A** and **C**, and is periodic with period **2**.

A1(0,1)

Gain Optimality

$$\rho^{\pi}(x) = \frac{1}{N} \lim_{N\to\infty} E\left(\sum_{t=0}^{N-1} R_t^{\pi}(x) \right)$$

Gain-optimal: policy π^* is one that maximizes the average reward over all states, that is $\rho^{\pi^*}(x) \geq \rho^{\pi}(x)$ over all policies and states.

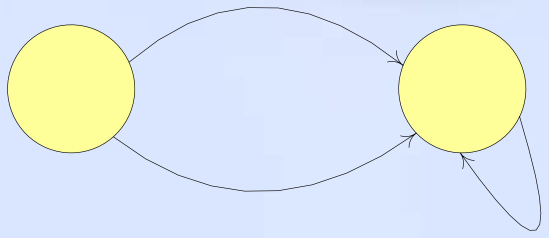
Gain Optimality

- **Since states in the recurrent class will be visited forever, the expected average reward cannot differ across these states.**
- **Since the transient states will eventually never be reentered, they can at most accumulate a finite total expected reward before entering a recurrent state, which vanishes under the limit.**

- For the example 1, the gain of the two policies are both -1;
- For the example 2, the gain of the two policies are both 1.

Bias Optimality

- Example 3
  - Both policies yield the same average reward.
  - Doing action **a1** is clearly preferable to doing **a2** in state **A**.



- Bias Optimality
  - Define a *value* function *V*: *S*→*R*.

$$V_\gamma^\pi(x) = \lim_{N\to\infty} E\left(\sum_{t=0}^{N-1}\gamma^t R_t^\pi(x)\right)$$

But, in average MDP, $\gamma = 1$?

Bias Optimality

- ***Average adjusted*** sum of rewards

$$V^{\pi}(x) = \lim_{N\to\infty} E\left(\sum_{t=0}^{N-1}\left(R_t^{\pi}(x) - \rho^{\pi}\right)\right)$$

- ***Bias value (relative value):*** represents the relative difference in total reward gained from starting in state ***x*** as opposed to some other state **s**.

$$V^{\pi}(x) - V^{\pi}(s) = \lim_{N\to\infty}\left\{E\left[\sum_{t=0}^{N-1} R_t^{\pi}(x)\right] - E\left[\sum_{t=0}^{N-1} R_t^{\pi}(s)\right]\right\}$$

Bias Optimality

Bias-optimal policy π^* :

If it is gain-optimal, and it also maximizes bias values.

Related works in dynamic programming

- DP: DP Methods for Average Rewards MDP
  - The policy iteration algorithm [Howard 1960]
  - Blackwell 1962, Bias optimality
  - Veinott 1969, N-discounted-optimality
  - Puterman 1994, Important book

Average Reward Bellman Equation

- Theorem 1: For any MDP that is either unichain or communicating, there exists a value function *V** and a scalar ρ* satisfying the equation

$$V^*(x) + \rho^* = \max_a \left[r(x,a) + \sum_y P_{xy}(a) V^*(y) \right]$$

- So the greedy policy achieves the optimal average reward.

Average Reward DP Algorithms

- Unichain Policy Iteration
  - Policy iteration iterates over two phases: policy evaluation and policy improvement.
  - Policy evaluation

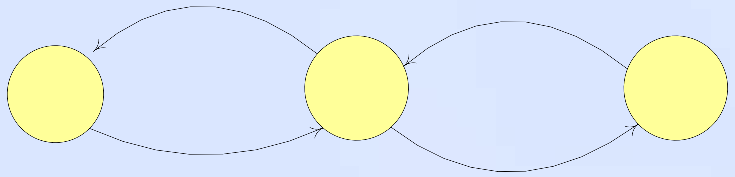
$$V^{\pi^k}(x)+\rho^{\pi^k}=r\left(s,\pi^k(x)\right)+\sum_y P_{xy}\left(\pi^k(x)\right)V^{\pi^k}(y)$$

  - Policy improvement

$$action=\arg\max_a\left(r(s,a)+\sum_y P_{xy}V^{\pi^k}(y)\right)$$

Average Reward DP Algorithms

- Unichain Policy Iteration
  - Does it produce bias-optimal polices, or only gain-optimal policies?
  - 1. Set V(A)=0
    - V(A)+1=2+V(B)→<u>V(B)=-1</u>
    - V(A)+1=0+V(C)→<u>V(C)=1</u>
  - 2. Policy 1: select **a2** in **A**.
  - 3. No policy improvement.
  - 4. Policy 1 is only gain-optimal and not bias-optimal.



Average Reward DP Algorithms

- Value Iteration
  - The difficulty with policy iteration is that it requires solving |S| equations at every iteration, which is computationally intractable when |S| is large.
  - Define *T*(*V*)(*x*), *T* is a ***monotone* mapping**.

$$T(V)(x) = \max_a \left[r(x,a) + \sum_y P_{xy}(a) V(y) \right]$$

Average Reward DP Algorithms

- Value Iteration
  - Note:
    - 1. The value iteration algorithm does not explicity compute the average reward, but this can be estimated as $V^{n+1}(x)$-$V^{n}(x)$ for large *n*.
    - 2. Disadvantage: the values *V*(*x*) can grow very large, causing numerical instability.
  - White's relative value iteration algorithm:

$$V^{k+1}(x) = T\left(V^{k}\right)(x) - T\left(V^{k}\right)(s)$$

Like policy iteration, value iteration cannot discriminate between the bias-optimal and gain-optimal policies in some MDP.

Average Reward DP Algorithms

- Asynchronous Value Iteration
  - Policy iteration & value iteration are both *synchronous*.
  - RL methods are *asynchronous*.

  - Convergence
    - The key is to ensure that the underlying mapping remains monotonic or a contraction with respect to the maximum norm.

Average Reward DP Algorithms

- Asynchronous Value Iteration
  - Jalali & Ferguson, 1990

The mapping is *monotonic*.

$$V^{t+1}(x) = T(V^t)(x) - \rho^t \qquad \forall x \neq s$$

$$V^t(s) = 0$$

  - ρ^t is the estimate of the average reward at time *t*, is independently estimated without using the relative values.

There are several ways of independently estimating average reward.

An Asynchronous Adaptive Control Method

- The above DP algorithms require
  - Complete knowledge of the state transition matrices
  - The expected payoffs for each action and state
- Jalali & Ferguson, 1990, A algorithm and B algorithm

$$1)\ action = \arg\max_a \left(r(s,a) + \sum_y P_{xy}^t(a) V^t(y) \right)$$

$$2)\ V^{t+1}(x) = T(V^t)(x) - \rho^t$$

$$3)\ \rho^{t+1} = \frac{K^{t+1}}{t+1}, \quad \text{where} \quad K^{t+1} = K^t + r(x,a)$$

4) Update the probabilty transition matrix entry $P_{xy}^t(a)$ using a maximum-likeihood estimator

An Asynchronous Adaptive Control Method

- Some discussion about B algorithm
  - 1. In order to guarantee convergence, the MDP is *identifiable* by an MLE-estimator.
  - 2. One modification: estimate the expected rewards $r(x,a)$ from sample reward.
  - 3. Another modification: Take random actions in step 1 (semi-uniform exploration)

Summary of average reward methods

ALGORITHM	GAIN OPTIMALITY
Unichain Policy Iteration	MDP is unichain
(Relative) Value Iteration	MDP is communicating
Asynchronous Relative Value Iteration	Dose not converge
Asynchronous Value Iteration with online Gain Estimation	A state *s* is reachable under every policy
Asynchronous Adaptive Control with online Gain Estimation	MDP is ergodic and MLE-Identifiable

History of average reward RL

- The study of average reward RL is currently at an early stage.
  - R-learning, the first average-reward RL method [Schwartz, ML1993]
    - Outperform discounted methods, such as Q-learning.
  - Modified R-learning, [Singh, AAAI1994]
  - A Model-based Algorithm for Bias-optimal, [Mahadevan, ML1996]
  - H-Learning, [Tadepalli, AI1998]
  - SMART, [Das, ManagementScience1999]
  - Relaxed SMART, [Gosavi, ML2004]
  - Q-P-Learning, [Gosavi, ML2004]
  - G-Learning, [Gao, SMC2006]

R-learning: A Model-Free Average Reward RL Method [Schwartz, ML1993]

$$R^{\pi}(x,a) = r(x,a) - \rho^{\pi} + \sum_{y} P_{xy}(a) V^{\pi}(y), \qquad V^{\pi}(y) = \max_{a} R^{\pi}(y,a)$$

$$1)\, R_{t+1}(x,a) \leftarrow (1-\beta) R_t(x,a) + \beta\left(\underline{r_{imm}(x,y)} - \rho_t + \max_{a} R_t(y,a)\right)$$

$$2)\, \rho_{t+1} \leftarrow (1-\alpha)\rho_t + \alpha\left[\underline{r_{imm}(x,a)} + \max_{a} R_t(y,a) - \max_{a} R_t(x,a)\right]$$

Important!

Some Variations on the basic R-learning

- Singh 1994[Singh, AAAI1994]
  - Adjust Bellman equation
  - Estimating average reward as the sample mean of the actual rewards
  - Updating the average reward on every step

1. Like all the preceding algorithms, it is unable to differentiate the bias-optimal policy from the gain-optimal policy.
2. Convergence Proof? *Monotonicity and Contraction.*

Some Variations on the basic R-learning

- Singh 1994[Singh, AAAI199

Not Max, But Next State!

$$B_\pi(V)(x_t) = R(x_t, \pi(x_t)) + V_t(x_{t+1})$$

- Algorithm 1

$$V_{t+1}(x_t) = (1-\alpha_t(x_t))V_t(x_t) + \alpha_t(x_t)(B_\pi(V_t)(x_t) - \rho_t)$$

where $\rho_0 = 0$, and

$$\rho_{t+1} = (1-\beta_t)\rho_t + \beta_t\left[B_\pi(V_t)(x_t) - V_t(x_t)\right]$$

- Algorithm 2

$$\rho_{t+1} = \frac{(t * \rho_t) + R(x_t, \pi(x_t))}{t+1}$$

Some Variations on the basic R-learning

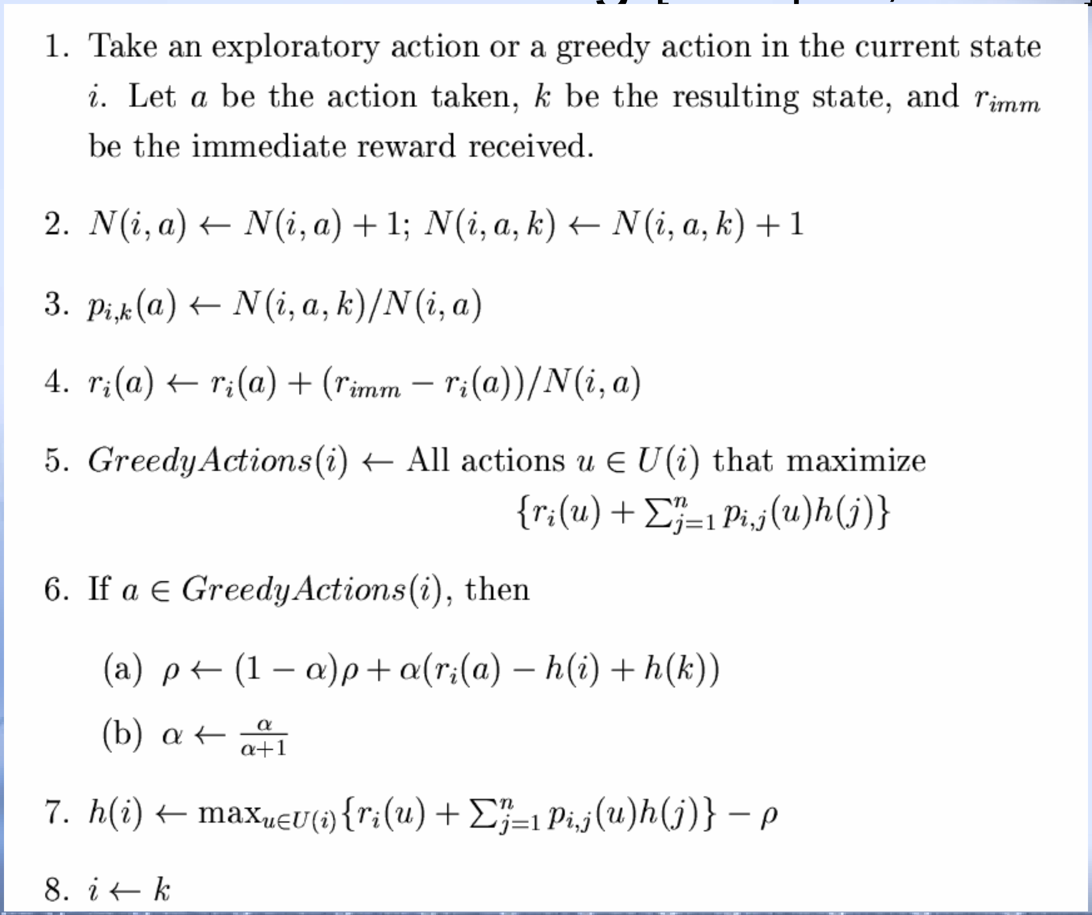
- Algorithm 3
  - The difference between Algorithm 3 and R-learning is that in R-learning the estimated average payoff is updated only when the greedy action is executed.
  - In Algorithm 3, the average payoff is updated with every action.

Algorithm 3 could be more efficient than R-learning since R-learning seems to waste information.

- Algorithm 4
  - Estimates average payoff like algorithm 2

Model-based Average Reward Reinforcement Learning

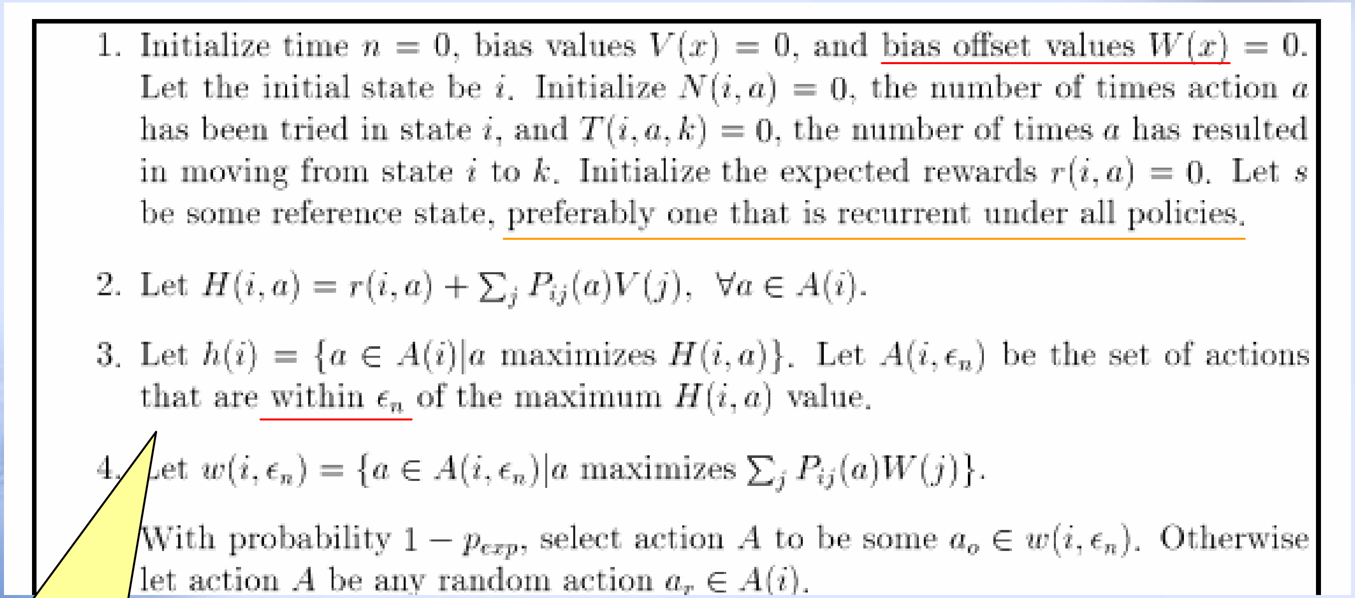
- Tadepalli & Ok: H-learning [Tadepalli, AI1998]



```
1. Take an exploratory action or a greedy action in the current state $i$. Let $a$ be the action taken, $k$ be the resulting state, and $r_{imm}$ be the immediate reward received.

2. $N(i,a) \leftarrow N(i,a)+1;\ N(i,a,k) \leftarrow N(i,a,k)+1$

3. $p_{i,k}(a) \leftarrow N(i,a,k)/N(i,a)$

4. $r_i(a) \leftarrow r_i(a) + (r_{imm} - r_i(a))/N(i,a)$

5. $GreedyActions(i) \leftarrow$ All actions $u \in U(i)$ that maximize
       $\{r_i(u) + \sum_{j=1}^{n} p_{i,j}(u)h(j)\}$

6. If $a \in GreedyActions(i)$, then

    (a) $\rho \leftarrow (1-\alpha)\rho + \alpha(r_i(a) - h(i) + h(k))$

    (b) $\alpha \leftarrow \frac{\alpha}{\alpha+1}$

7. $h(i) \leftarrow \max_{u \in U(i)}\{r_i(u) + \sum_{j=1}^{n} p_{i,j}(u)h(j)\} - \rho$

8. $i \leftarrow k$
```

A Model-based Algorithm for Bias-optimal

- Mahadevan, ML1996



1. Initialize time $n = 0$, bias values $V(x) = 0$, and bias offset values $W(x) = 0$. Let the initial state be i. Initialize $N(i,a) = 0$, the number of times action a has been tried in state i, and $T(i,a,k) = 0$, the number of times a has resulted in moving from state i to k. Initialize the expected rewards $r(i,a) = 0$. Let s be some reference state, preferably one that is recurrent under all policies.

2. Let $H(i,a) = r(i,a) + \sum_j P_{ij}(a)V(j), \ \forall a \in A(i)$.

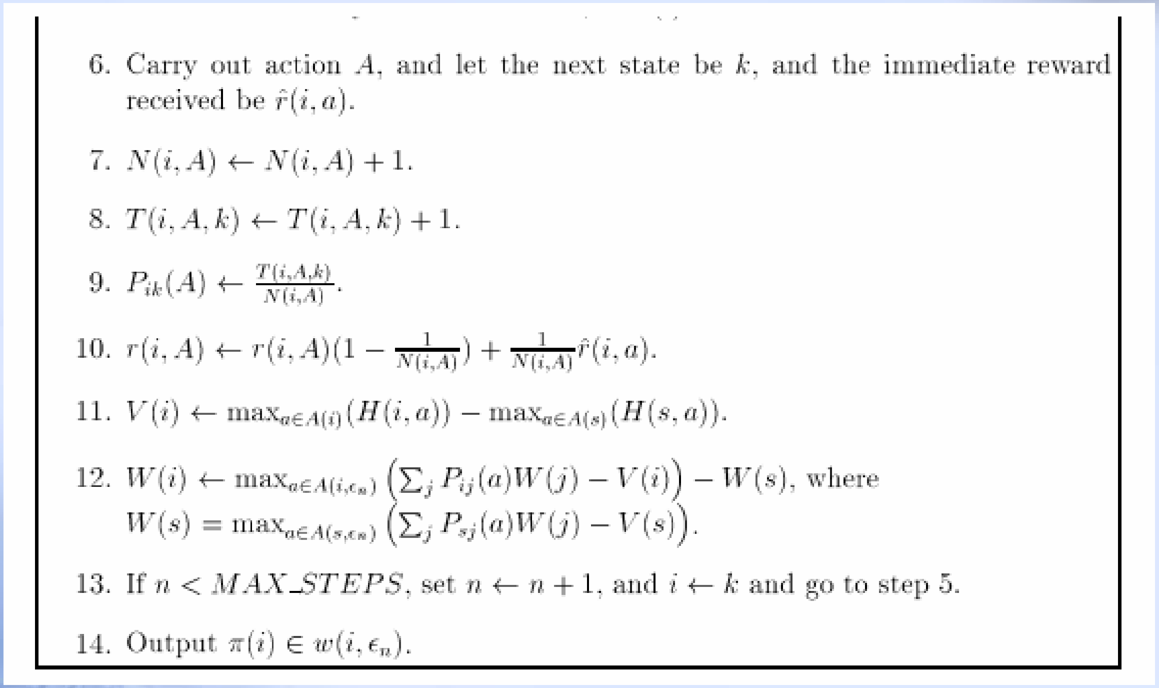
3. Let $h(i) = \{a \in A(i) | a \text{ maximizes } H(i,a)\}$. Let $A(i,\epsilon_n)$ be the set of actions that are within ϵ_n of the maximum $H(i,a)$ value.

4. Let $w(i,\epsilon_n) = \{a \in A(i,\epsilon_n) | a \text{ maximizes } \sum_j P_{ij}(a)W(j)\}$.

With probability $1 - p_{exp}$, select action A to be some $a_o \in w(i,\epsilon_n)$. Otherwise let action A be any random action $a_r \in A(i)$.

Let A be the set of actions

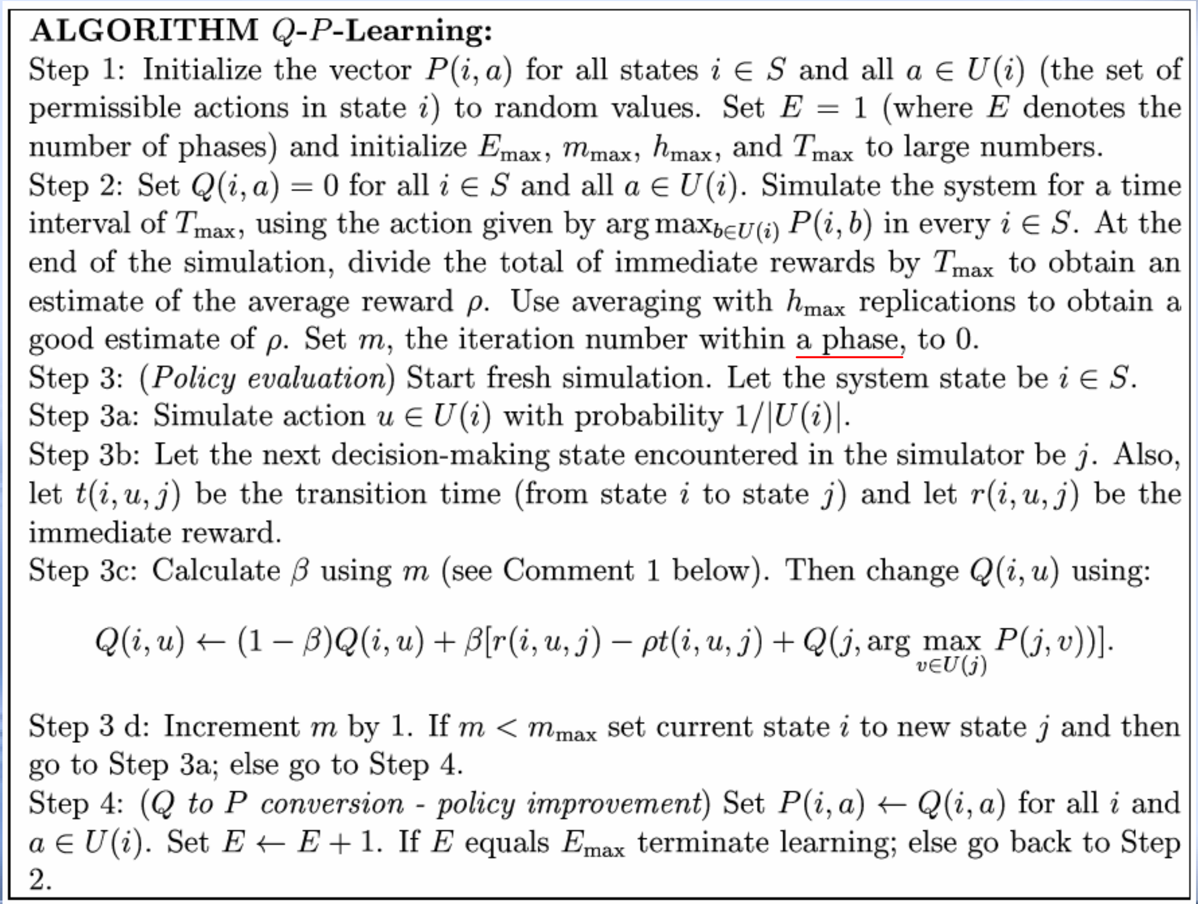
A Model-based Algorithm for Bias-optimal



```
6. Carry out action $A$, and let the next state be $k$, and the immediate reward received be $\hat{r}(i,a)$.
7. $N(i,A) \leftarrow N(i,A) + 1$.
8. $T(i,A,k) \leftarrow T(i,A,k) + 1$.
9. $P_{ik}(A) \leftarrow \frac{T(i,A,k)}{N(i,A)}$.
10. $r(i,A) \leftarrow r(i,A)(1 - \frac{1}{N(i,A)}) + \frac{1}{N(i,A)}\hat{r}(i,a)$.
11. $V(i) \leftarrow \max_{a \in A(i)}(H(i,a)) - \max_{a \in A(s)}(H(s,a))$.
12. $W(i) \leftarrow \max_{a \in A(i,\epsilon_n)} \left(\sum_j P_{ij}(a)W(j) - V(i)\right) - W(s)$, where
    $W(s) = \max_{a \in A(s,\epsilon_n)} \left(\sum_j P_{sj}(a)W(j) - V(s)\right)$.
13. If $n < MAX\_STEPS$, set $n \leftarrow n+1$, and $i \leftarrow k$ and go to step 5.
14. Output $\pi(i) \in w(i,\epsilon_n)$.
```

A Reinforcement Learning Algorithm based on Policy Iteration for Average Reward

- Gosavi, ML2004: *Q*-*P*-Learning



ALGORITHM Q-P-Learning:

Step 1: Initialize the vector $P(i,a)$ for all states $i \in S$ and all $a \in U(i)$ (the set of permissible actions in state i) to random values. Set $E = 1$ (where E denotes the number of phases) and initialize $E_{\max}$, $m_{\max}$, $h_{\max}$, and $T_{\max}$ to large numbers.

Step 2: Set $Q(i,a) = 0$ for all $i \in S$ and all $a \in U(i)$. Simulate the system for a time interval of $T_{\max}$, using the action given by $\arg\max_{b \in U(i)} P(i,b)$ in every $i \in S$. At the end of the simulation, divide the total of immediate rewards by $T_{\max}$ to obtain an estimate of the average reward ρ. Use averaging with $h_{\max}$ replications to obtain a good estimate of ρ. Set m, the iteration number within a phase, to 0.

Step 3: (*Policy evaluation*) Start fresh simulation. Let the system state be $i \in S$.

Step 3a: Simulate action $u \in U(i)$ with probability $1/|U(i)|$.

Step 3b: Let the next decision-making state encountered in the simulator be j. Also, let $t(i,u,j)$ be the transition time (from state i to state j) and let $r(i,u,j)$ be the immediate reward.

Step 3c: Calculate β using m (see Comment 1 below). Then change $Q(i,u)$ using:

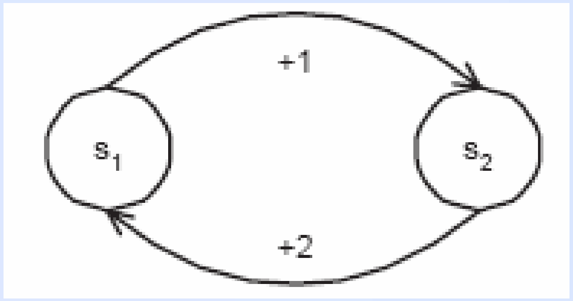
$$Q(i,u) \leftarrow (1-\beta)Q(i,u) + \beta[r(i,u,j) - \rho t(i,u,j) + Q(j, \arg\max_{v \in U(j)} P(j,v))].$$

Step 3 d: Increment m by 1. If $m < m_{\max}$ set current state i to new state j and then go to Step 3a; else go to Step 4.

Step 4: (*Q to P conversion - policy improvement*) Set $P(i,a) \leftarrow Q(i,a)$ for all i and $a \in U(i)$. Set $E \leftarrow E + 1$. If E equals $E_{\max}$ terminate learning; else go back to Step 2.

Average Reward Reinforcement Learning Algorithm based on a Reference State

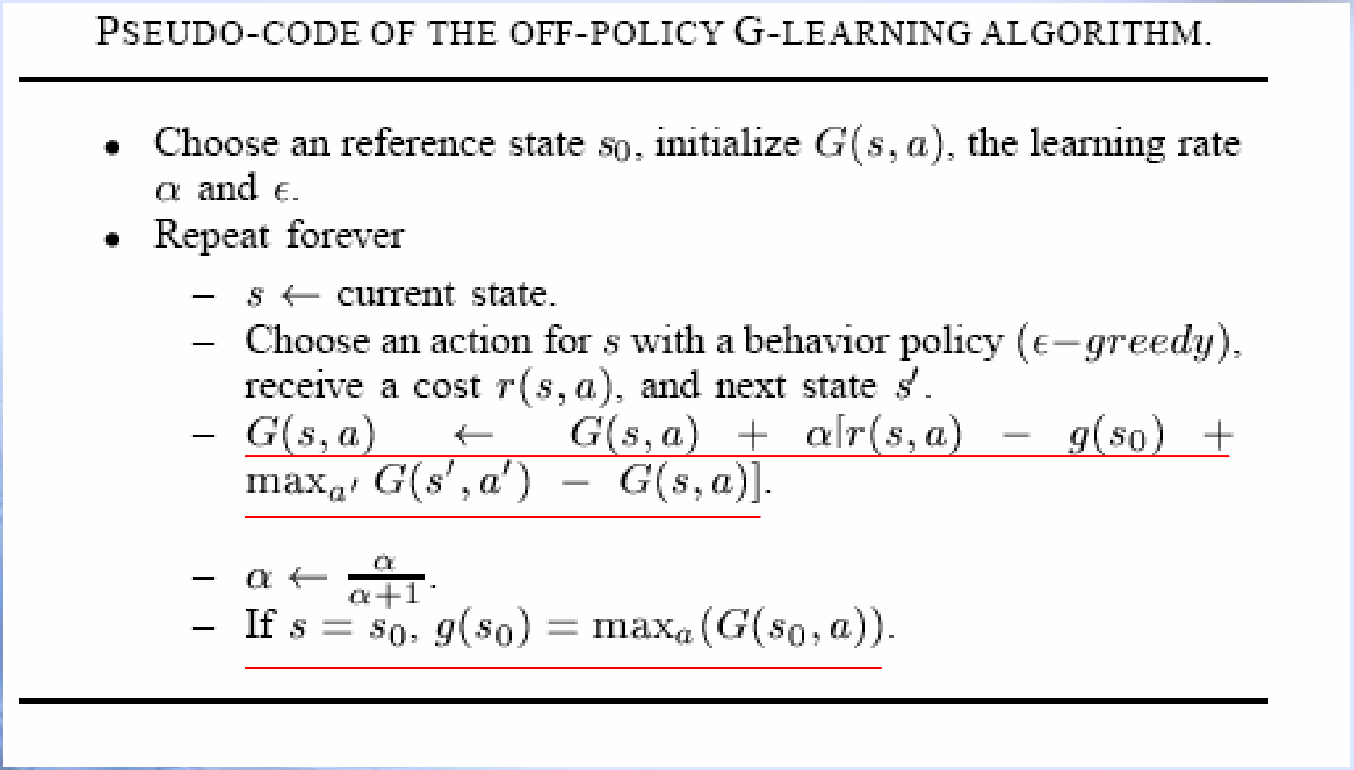
- Our RL algorithm



$$\begin{cases} b_1 = 1 - \rho + b_2 \\ b_2 = 2 - \rho + b_1 \end{cases} \rightarrow \begin{cases} g_1 = 1 - g_1 + g_2 \\ g_2 = 2 - g_1 + g_1 \end{cases}$$

b1-b2=-1/2 **g1=3/2, g2=2**

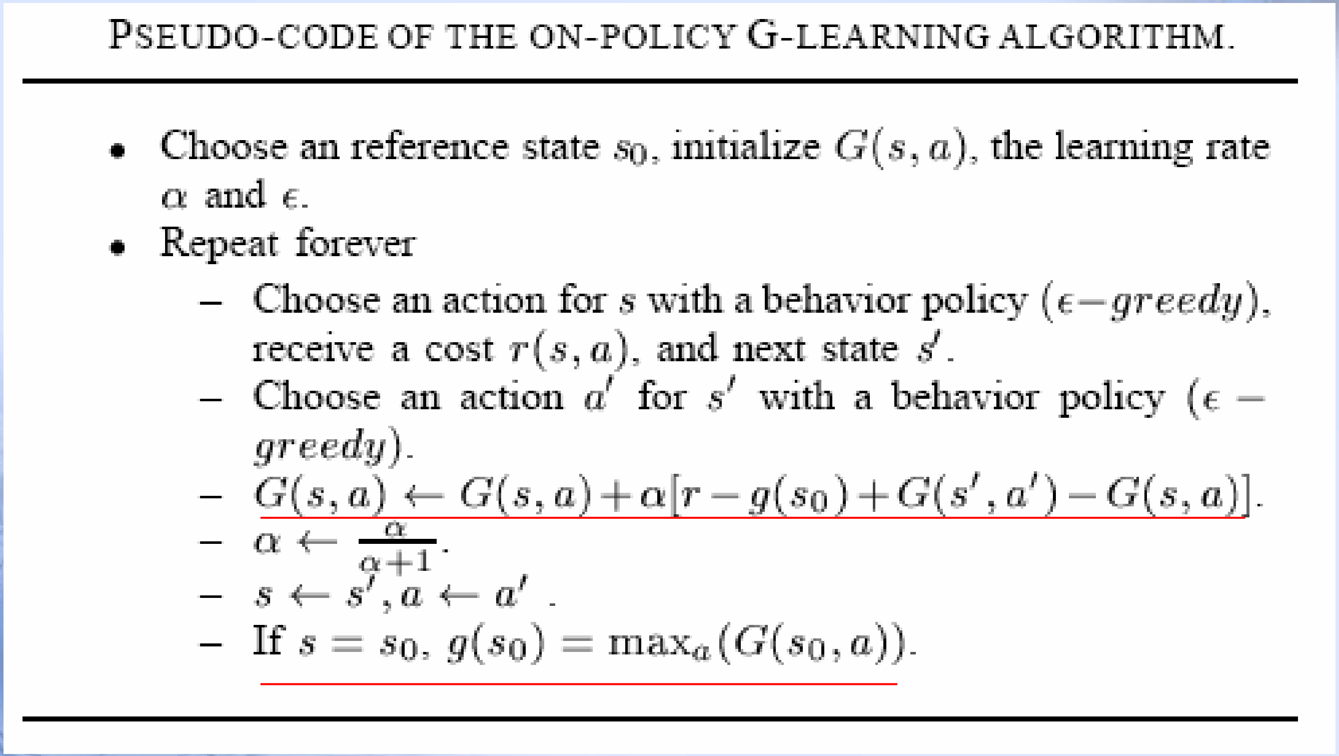
Off-policy G-learning



PSEUDO-CODE OF THE OFF-POLICY G-LEARNING ALGORITHM.

```
• Choose an reference state $s_0$, initialize $G(s,a)$, the learning rate $\alpha$ and $\epsilon$.
• Repeat forever
    – $s \leftarrow$ current state.
    – Choose an action for $s$ with a behavior policy ($\epsilon-greedy$), receive a cost $r(s,a)$, and next state $s'$.
    – $G(s,a) \leftarrow G(s,a) + \alpha[r(s,a) - g(s_0) + \max_{a'} G(s',a') - G(s,a)]$.
    – $\alpha \leftarrow \frac{\alpha}{\alpha+1}$.
    – If $s = s_0$, $g(s_0) = \max_a(G(s_0,a))$.
```

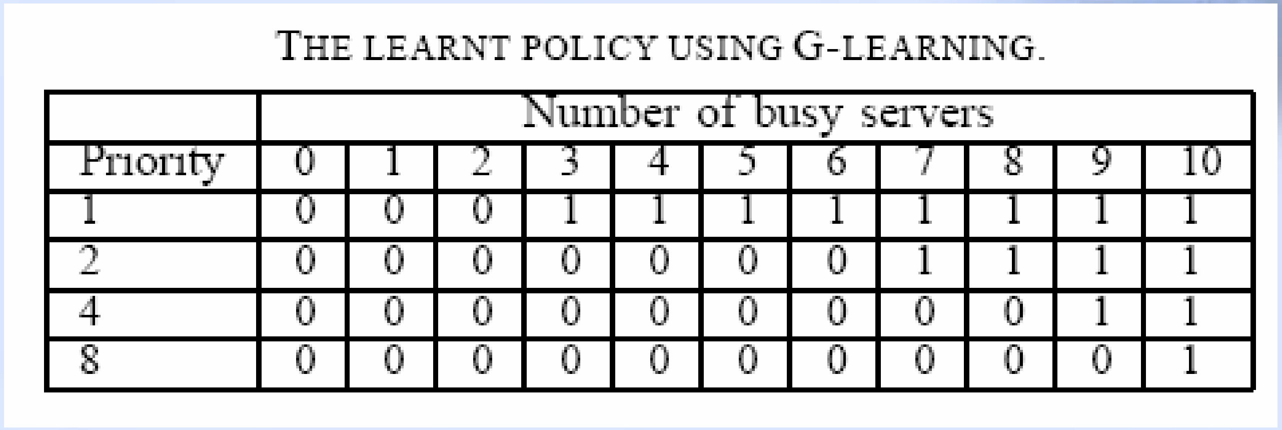
On-policy G-learning



PSEUDO-CODE OF THE ON-POLICY G-LEARNING ALGORITHM.

```
• Choose an reference state $s_0$, initialize $G(s,a)$, the learning rate $\alpha$ and $\epsilon$.
• Repeat forever
    – Choose an action for $s$ with a behavior policy $(\epsilon-greedy)$, receive a cost $r(s,a)$, and next state $s'$.
    – Choose an action $a'$ for $s'$ with a behavior policy $(\epsilon-greedy)$.
    – $\underline{G(s,a) \leftarrow G(s,a)+\alpha[r-g(s_0)+G(s',a')-G(s,a)]}$.
    – $\alpha \leftarrow \frac{\alpha}{\alpha+1}$.
    – $s \leftarrow s', a \leftarrow a'$ .
    – $\underline{\text{If } s = s_0,\ g(s_0) = \max_a(G(s_0,a))}$.
```

Case Study: The learnt policy using G-learning



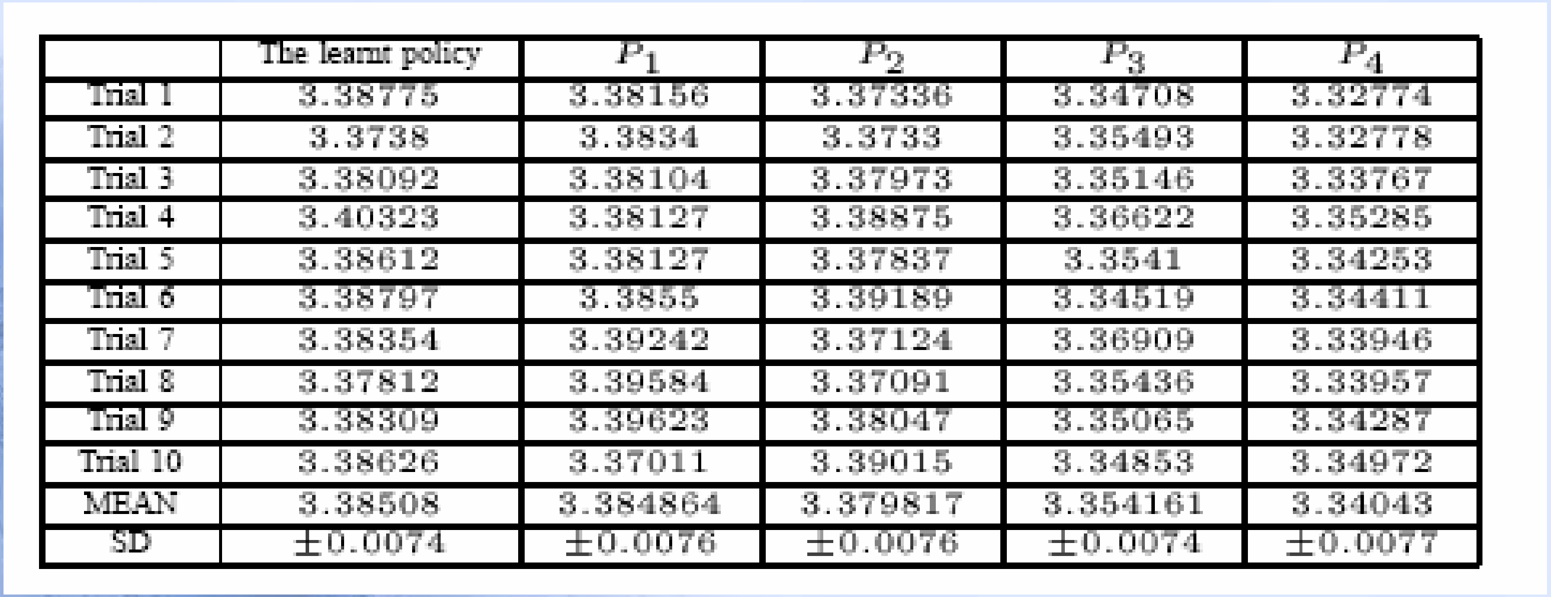
THE LEARNT POLICY USING G-LEARNING.

	Number of busy servers										
Priority	0	1	2	3	4	5	6	7	8	9	10
1	0	0	0	1	1	1	1	1	1	1	1
2	0	0	0	0	0	0	0	1	1	1	1
4	0	0	0	0	0	0	0	0	0	1	1
8	0	0	0	0	0	0	0	0	0	0	1

- P1: (2,1)→1 Reject
- P2: (4,1)→0 Accept
- P3: (8,4)→1 Reject
- P4: (9,4)→0 Accept

Comparison (1)

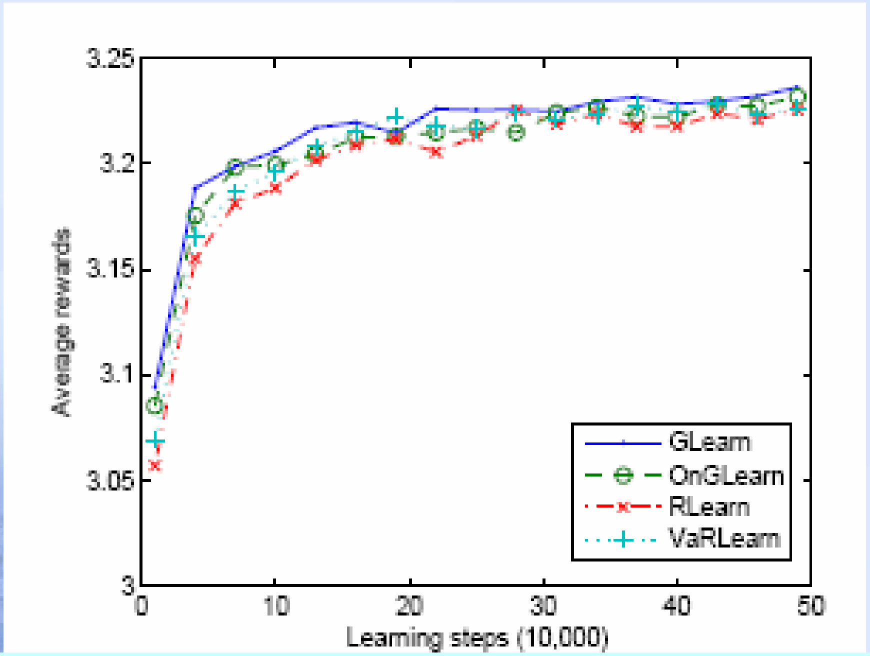
- Comparison of average rewards between the learnt policy and P1, P2,P3 & P4



	The learnt policy	P_1	P_2	P_3	P_4
Trial 1	3.38775	3.38156	3.37336	3.34708	3.32774
Trial 2	3.3738	3.3834	3.3733	3.35493	3.32778
Trial 3	3.38092	3.38104	3.37973	3.35146	3.33767
Trial 4	3.40323	3.38127	3.38875	3.36622	3.35285
Trial 5	3.38612	3.38127	3.37837	3.3541	3.34253
Trial 6	3.38797	3.3855	3.39189	3.34519	3.34411
Trial 7	3.38354	3.39242	3.37124	3.36909	3.33946
Trial 8	3.37812	3.39584	3.37091	3.35436	3.33957
Trial 9	3.38309	3.39623	3.38047	3.35065	3.34287
Trial 10	3.38626	3.37011	3.39015	3.34853	3.34972
MEAN	3.38508	3.384864	3.379817	3.354161	3.34043
SD	±0.0074	±0.0076	±0.0076	±0.0074	±0.0077

Comparison (2)

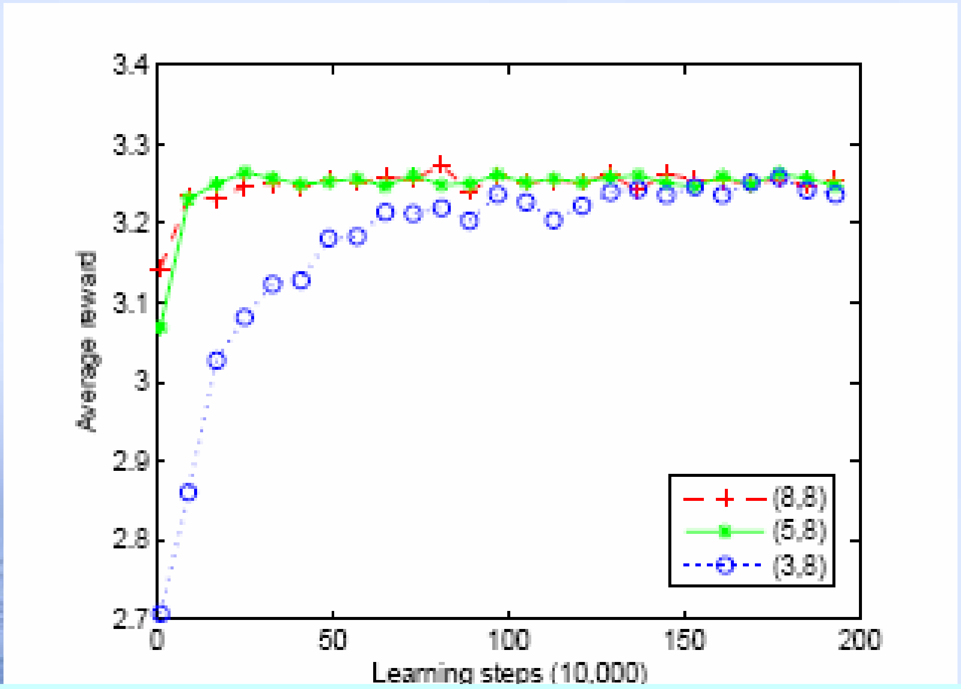
- Comparing the on-line learning performance between G-learning and R-learning



G-learning has the optimal learning performance than R-learning

Comparison (3)

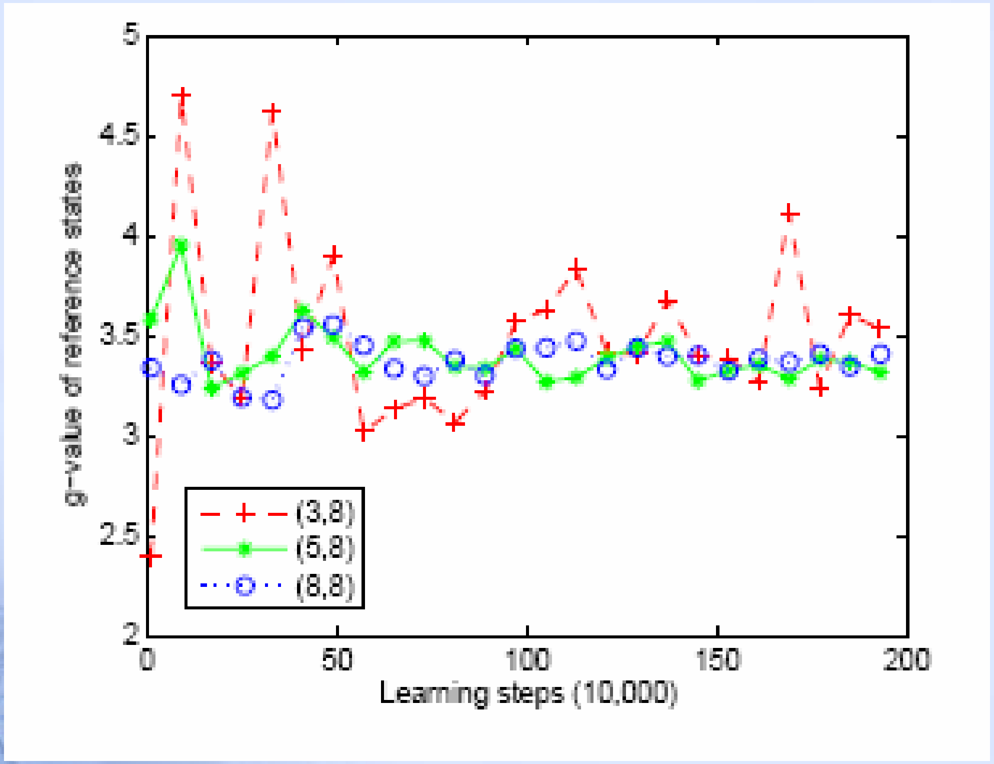
- Comparing the learning performance between different reference states



Choose the frequent visited states as reference state

Comparison (4)

- Comparing the g-value of different reference states



Summary

	Model-free	On-policy	Value iteration	Directly measure ρ	When update the ρ	Bias-optimal
R-learning	√	X	√	Adaptive	Greedy action	X
Variations on R-learning	√	√	√	Adaptive or directly	Every action	X
H-learning	X	N/A	√	Adaptive	Greedy action	X
A Model-based Algorithm for Bias-optimal	X	X	√	N/A	N/A	√
Q-P-learning	X	X	X	N/A	N/A	?
G-learning	√	Both	√	Reference state	Visit reference state	?

Direction for Future Research

- Bias-optimal RL Algorithms
- Value Function Approximation in Average Reward MDP
- Multi-Chain Average Reward Algorithms
- Modular Average Reward Methods
- Average Reward Methods in SMDP
- Difference between Every algorithm

Future works: Exploration Strategies

- Undirected
  - Undirected exploration methods do not use the results of learning to guide exploration; they merely select a random action some of the time.
  - Semi-Uniform Exploration
  - Boltzmann Exploration
- Directed
  - Use the results of learning to decide where to concentrate the exploration efforts.
  - Recency-based Exploration
  - Uncertainty estimation Exploration

References

- [Schwartz, ML1993] A. Schwartz. A reinforcement learning method for maximizing undiscounted rewards in: *Proceedings of the Tenth Annual Conference on Machine Learning*, 1993, pp. 298-- 305.
- [Howard, 1960] R. Howard. Dynamic Programming and Markov Processes. *MIT Press*, 1960.
- [Puterman, 1994] M. L. Puterman. Markov Decision Processes---Discrete Stochastic Dynamic Programming. *John Wiley & Sons*, Inc., New York, NY, 1994.
- [Singh, ICML1994] S. Singh. Reinforcement learning algorithms for average payoff Markovian decision processes. In *Proceedings of the 12th International Conference on Machine Learning*, pages 202--207. Morgan Kaufmann, 1994.
- [Tadepalli, 1994] Prasad Tadepalli, DoKyeong Ok. H-learning: A Reinforcement Learning Method to Optimize Undiscounted Average Reward , Technical report, Oregon State University, 1994
- [Mahadevan, ICML1994] Sridar Mahadevan. To Discount or Not to Discount in Reinforcement Learning: A Case Study Comparing R- Learning and Q-Learning. *Proceedings of the 11th International Conference on Machine Learning*, New Brunswick, NJ, pp. 164-172, July, 1994.

References

- [Jalali & Ferguson, 1990] A. Jalali and M. Ferguson. Computationally efficient adaptive control algorithms for Markov chains. In *Proceedings of the 28th IEEE Conference on Decision and Control*. Pp1283-1288, 1989.
- [Singh, AAAI1994] S. Singh. Reinforcement learning algorithms for average-payoff Markovian decision processes. In *Proceedings of the 12th AAAI*. MIT Press, 1994.
- [Mahadevan, ML1996] Sridhar Mahadevan. Average Reward Reinforcement learning: Foundations, Algorithms, and Empirical Results. *Machine Learning*, 22, 159-196, 1996.
- [Tadepalli, AI1998] P. Tadepalli and D. Ok, Model--Based Average Reward Reinforcement Learning. *Artificial Intelligence*, 100:177-224, 1998.
- [Gosavi, ML2004] Abhijit Gosavi. A Reinforcement Learning Algorithm based on Policy Iteration for Average Reward: Empirical Results with Yield Management and Convergence Analysis. *Machine Learning*, 55(1), 5-29, 2004.

References

- [Das, 1999] T. Das and A. Gosavi and S. Mahadevan and N. Marchalleck. Solving semi-markov decision problems using average reward reinforcement learning. *Management Science*, 45 (4):560-574, 1999.
- [Gao, SMC2006] Y. Gao, R. Zhou et al. G-learning: An Average Reward Reinforcement Learning Algorithm Based on the Performance Potential of a Reference State. *IEEE Trans. Syst., Man, Cybern. B, Cybern*. Submitted.

Thanks!

Any Questions?