

# A SURVAY OF REINFORCEMENT LEARNING METHODS IN THE WINDY AND CLIFF-WALKING GRIDWORLDS

Ryan J. Meuth

Department of Electrical and Computer Engineering  
University of Missouri at Rolla

## ABSTRACT

This report details the implementation of three Reinforcement learning methods, Monte Carlo, SARSA, and Q-Learning, and compares their performances in the Windy and Cliff-Walking Gridworlds.

## 1. INTRODUCTION

One of the simplest methods for finding the optimal policy in a unknown environment is the Monte Carlo Algorithm. This episodic algorithm explores the state-space, using a policy and receiving a reward  $R$  for the action taken in each state. These rewards are averaged, and a new policy is constructed from the resulting information. The policy that the algorithm follows while exploring may be either the current best policy (on-policy MC control) or an arbitrary exploration policy (off-policy MC control). Here, the On-Policy Monte Carlo Control algorithm is implemented in the Gridworld Environment using an  $\epsilon$ -soft policy.

The SARSA algorithm is an on-policy Temporal Difference control algorithm that examines the value of a transition between one state and the next. The power in the SARSA algorithm lies in it's efforts to predict the value of the next state by using the expected  $\epsilon$ -soft next state from the current policy. This allows the algorithm to learn quickly, and for infinite-time problems with decreasing  $\epsilon$  values, optimality is assured.

The Q-Learning algorithm is a variant of the SARSA algorithm where the prediction stage uses a greedy policy to determine the expected next state. This makes Q-learning an off-policy method, but the greedy policy is learned directly, leading to faster convergence to the optimal policy.

## 2. ENVIRONMENT

In the Gridworld, the agent is at one of a discrete set of grid positions. The actions are increments to the position components of the agent. Each may be changed by +1, -1, in one step, for a total of eight actions corresponding to the movements of a King Piece on a chessboard. Each episode

begins in one start state and ends when the agent reaches the goal. The rewards are -1 for each step that the agent stays in the bounds of the world, and -5 for if the agent attempts to leave the boundaries. Actually leaving the track is not allowed, but the position is always advanced by at least one cell along either the horizontal or vertical axes. Reaching the goal state elicits a reward of +5.[1]

Adding wind to the gridworld is accomplished by shifting resultant next states upward by a constant which varies from column to column. Stochastic Wind is generated by adding or subtracting 1 randomly from the specified wind constant.

An additional hazard in the gridworld is that of Cliffs. Cliffs are implemented as terminal states in the gridworld that result in a reward of -100. This makes cliffs highly undesirable regions that must be avoided to achieve optimality.

## 3. ON-POLICY MONTE CARLO CONTROL

Monte Carlo control algorithms are simple, episodic learning algorithms that require only *experience* from their environment to discover an optimal policy. On Policy Monte Carlo algorithms utilize an arbitrary or  $\epsilon$ -soft policy to generate episodes for optimal policy updates, while On Policy Monte Carlo algorithms use the current optimal policy estimate to generate episodes.

For this problem, the following On Policy Monte Carlo Control algorithm was implemented[1]:

Initialize, for all  $s \in S$ ,  $a \in A(s)$ :

$Q(s, a) \leftarrow$  arbitrary

$\pi(s) \leftarrow$  arbitrary

$Returns(s, a) \leftarrow$  empty list

Repeat for 10000 Iteratons:

For each start state:

Generate an Episode.

For each  $s, a$  appearing in the episode:

$R \leftarrow$  return following  $s, a$

Append  $R$  to  $Returns(s, a)$

$Q(s, a) \leftarrow$  Average( $Returns(s, a)$ )

For each  $s$  appearing in the episode:

$$\pi(s) \leftarrow \arg \max_a Q(s, a)$$

#### 4. SARSA CONTROL

The SARSA algorithm is an on-policy Temporal Difference control algorithm that examines the value of a transition between one state and the next.

For this problem, the following SARSA Control algorithm was implemented[1]:

Initialize, for all  $s \in S, a \in A(s)$ :

$$Q(s, a) \leftarrow \text{arbitrary}$$

$$\pi(s) \leftarrow \text{arbitrary}$$

Repeat for each episode:

Initialise  $s$

Choose  $a$  from  $s$  using an  $\epsilon$ -greedy policy on  $Q$

Repeat for each state in the episode:

Take action  $a$ , observe  $r$  and  $s'$

Choose  $a'$  from  $s'$  using an  $\epsilon$ -greedy policy on  $Q$

$$Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma Q(s', a') - Q(s, a)]$$

$$s \leftarrow s', a \leftarrow a'$$

#### 5. Q-LEARNING CONTROL

The Q-Learning algorithm is a variant of the SARSA algorithm where the prediction stage uses a greedy policy to determine the expected next state.

For this problem, the following Q-Learning Control algorithm was implemented[1]:

Initialize, for all  $s \in S, a \in A(s)$ :

$$Q(s, a) \leftarrow \text{arbitrary}$$

$$\pi(s) \leftarrow \text{arbitrary}$$

Repeat for each episode:

Initialise  $s$

Choose  $a$  from  $s$  using an  $\epsilon$ -greedy policy on  $Q$

Repeat for each state in the episode:

Take action  $a$ , observe  $r$  and  $s'$

Choose  $a'$  from  $s'$  using a greedy policy on  $Q$

$$Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma Q(s', a') - Q(s, a)]$$

$$s \leftarrow s', a \leftarrow a'$$

#### 6. EXPERIMENTAL RESULTS

The gridworld for this problem was set at 10 units by 8 units. Each algorithm was run for 10000 iterations on the windy gridworld, stochastic windy gridworld, the cliff gridworld, and stochastic wind cliff gridworld. For each algorithm,  $\epsilon = 0.1$ , and for SARSA and Q-Learning,  $\alpha = 0.1$  and  $\gamma = 0.1$ . For the windy gridworlds, a start point was set at (0,3) with the end point at (7,3) with wind constants of 1, 1, 1, 2, 2,

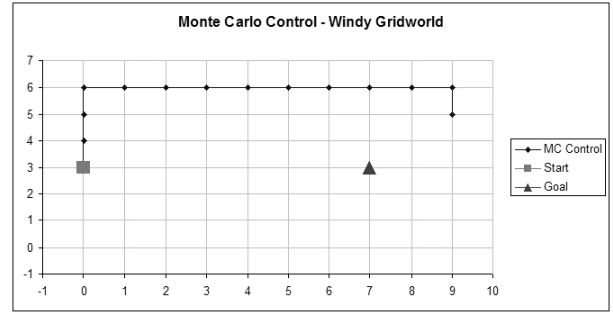


Fig. 1. Monte-Carlo Controlled Windy Gridworld Path

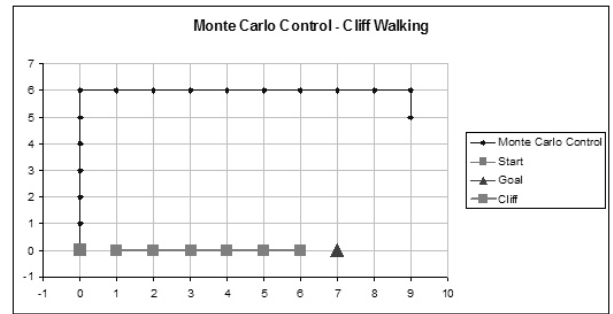


Fig. 2. Monte Carlo Controlled Cliff Walking Path

1 at columns 3-8 respectively. For the cliff gridworlds, the cliff was placed on the bottom edge of the world, between points 1 and 6, with a start point at (0,0) and goal at (7,0).

#### 7. CONCLUSIONS

As can be seen in figures 1 and 2, the Monte Carlo control algorithm is not well suited to the gridworld environment, as it is by no means guaranteed to terminate. Because of this the control algorithm is very likely to develop a looping path that does not reach the goal, as is the case in these first two figures.

The SARSA algorithm handles these situations much

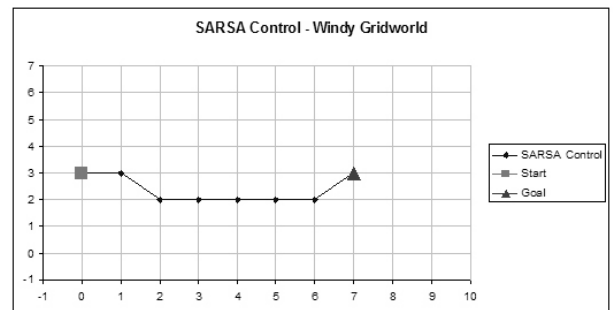
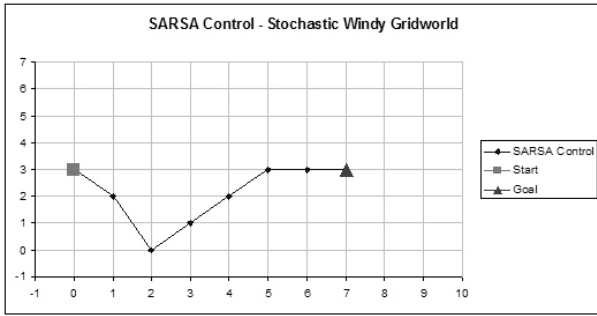
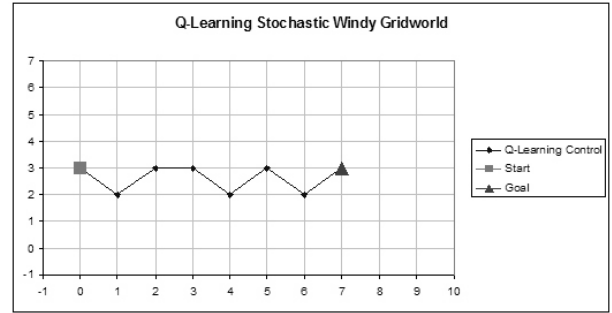


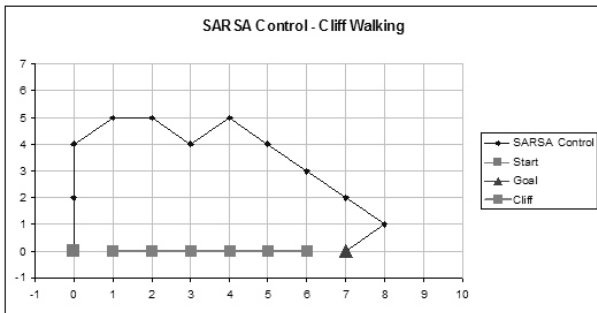
Fig. 3. SARSA Controlled Windy Gridworld Path



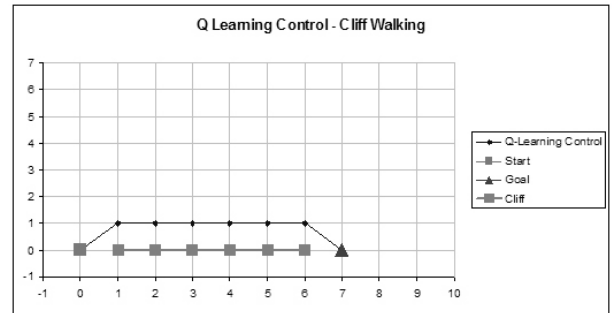
**Fig. 4.** SARSA Controlled Stochastic Windy Gridworld Path



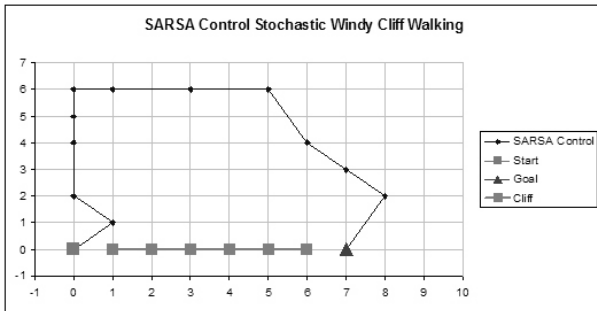
**Fig. 8.** Q-Learning Controlled Stochastic Windy Gridworld Path



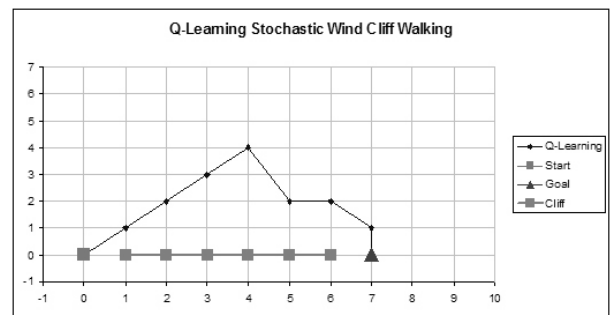
**Fig. 5.** SARSA Controlled Cliff Walking Path



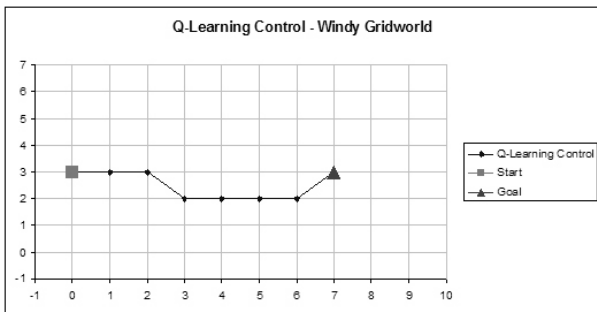
**Fig. 9.** Q-Learning Controlled Cliff Walking Path



**Fig. 6.** SARSA Controlled Stochastic Wind Cliff Walking Path



**Fig. 10.** Q-Learning Controlled Stochastic Wind Cliff Walking Path



**Fig. 7.** Q-Learning Controlled Windy Gridworld Path

better, and figures 3-6 show that it finds a path to the goal in all situations. In the case of the cliff walking environment, we see that the policy generated is very conservative, giving the cliff a wide berth. This conservatism increases when stochastic wind is added to the cliff walking gridworld, in order to compensate for extra uncertainty of the outcome of any given action. This effect is also illustrated in the stochastic windy gridworld scenario.

The Q-Learning algorithm performed comparably to SARSA in the windy gridworld, both algorithms arrived at a 7-step path to the goal. However, for all other scenarios, Q-learning is significantly more greedy, and therefore optimal. For example, in the cliff walking scenario, the resulting policy is very close to the edge of the cliff. Even with stochastic wind added to the cliff scenario, Q-learning provides a path that is more aggressive and superior than that provided by SARSA.

## 8. REFERENCES

- [1] Richard S. Sutton and Andrew G. Barto, "Reinforcement Learning" *The MIT Press*, 1998.