Investigation of Maximization Bias in Sarsa Variants

Ganesh Tata
Department of Computing Science
University of Alberta
Edmonton, AB, Canada
gtata@ualberta.ca

Eric Austin
Department of Computing Science
University of Alberta
Edmonton, AB, Canada
eaustin@ualberta.ca

2021 IEEE Symposium Series on Computational Intelligence (SSCI) | 978-1-7281-9048-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/SSCI50451.2021.9660081

***Abstract*—The overestimation of action values caused by randomness in rewards can harm the ability to learn and the performance of reinforcement learning agents. This maximization bias has been well established and studied in the off-policy Q-learning algorithm. However, less study has been done for on-policy algorithms such as Sarsa and its variants. We conduct a thorough empirical analysis on Sarsa, Expected Sarsa, and n-step Sarsa. We find that the on-policy Sarsa variants suffer from less maximization bias than off-policy Q-learning in several test environments. We show how the choice of hyper-parameters impacts the severity of the bias. A decaying learning rate schedule results in more maximization bias than a fixed learning rate. Larger learning rates lead to larger overestimation. A larger exploration parameter leads to worse bias in Q-learning but less bias in the on-policy algorithms. We also show that a larger variance in rewards leads to more bias in both Q-Learning and Sarsa, but Sarsa is less affected than Q-learning.**

***Index Terms*—Reinforcement Learning, Sarsa, Maximization Bias**

I. Reinforcement Learning and Maximization Bias

Maximization bias is a source of error that affects reinforcement learning algorithms such as Q-learning [1]. In reinforcement learning, an agent learns to maximize a numeric reward signal. The agent maintains and updates estimates of the value of taking an action in a state using the rewards obtained through trial-and-error interaction with the environment. As the agent attempts to maximize rewards, the actions it takes and its future updates are based on the maximum of these estimates which can lead to the systematic overestimation of some action values, i.e., maximization bias. This bias can lead to poor performance and interfere with the agent's ability to learn because the agent's choice of action is determined by its action value estimates.

In this paper, we analyze this bias in the Sarsa algorithm and two of its variants, Expected Sarsa and n-step Sarsa. We forego studying Sarsa(λ) as ultimately our interest is how this bias manifests in agents with neural network architectures where eligibility traces are not used. Our contribution is a thorough empirical investigation into the impact of this bias on the learning performance of these algorithms and how this compares to the severity of the bias for Q-learning. We study the impact of algorithm hyper-parameters and environment dynamics on the severity of the bias. Our results provide insights into the nature of maximization bias and its effects on learning for on-policy algorithms such as Sarsa, which can inform design choices for algorithms and learning systems. We find that the on-policy Sarsa variants suffer from much less maximization bias in our test environments than Q-learning, which is off-policy. We show that the choice of learning rate impacts the severity of the bias, and a decaying learning rate schedule suffers from more bias than a fixed learning rate. We find that a higher rate of exploration results in more severe bias in Q-learning, whereas the opposite is true for the on-policy algorithms. Our experiments show that higher variance in rewards causes more severe bias but that Sarsa consistently suffers from less bias than Q-learning across variance levels.

Our work is not the first research on maximization bias in Sarsa. The Double Sarsa and Double Expected Sarsa algorithms have been developed in an attempt to eliminate this bias [2]. These algorithms were compared to Sarsa and Expected Sarsa on a single grid world environment. The double algorithms perform better when the rewards are stochastic with negative expected value, i.e., when we expect to see maximization bias. However, the analysis is limited to comparing these four algorithms on a single environment with a limited number of hyper-parameter settings. Our work differs from theirs in that we don't propose novel algorithms. Instead, our goal is to empirically investigate maximization bias in on-policy algorithms in arious settings and contrast this with the bias of Q-learning.

II. Background

We formalize the reinforcement learning problem as a Markov Decision Process (MDP) [3]. A MDP is defined by a set of states $\mathcal{S}$, a set of actions $\mathcal{A}$, a set of rewards $\mathcal{R} \subseteq \mathbb{R}$, and a four argument dynamics function p which gives the probability of transitioning to a state $s' \in \mathcal{S}$ with reward $r \in \mathcal{R}$ given that the agent is in state $s \in \mathcal{S}$ and takes action $a \in \mathcal{A}$. The agent's behaviour is determined by a policy π which gives the probability of selecting an action $a \in \mathcal{A}$ in a state $s \in \mathcal{S}$. In the tabular case, the agent maintains a table of action value estimates for each state-action pair $Q(s, a)$. It may be impractical, due to a very large number of states and actions, or impossible, due to continuous state and/or action spaces, to maintain an estimate for each state-action pair (s, a). In this case, the agent approximates the values using a parameterized function $\hat{q}(s, a, \mathbf{w})$ where $\mathbf{w}$ is the parameter vector, $\mathbf{x}(s, a)$ is the feature vector of (s, a), and the function is something like linear function approximation or a neural network.

The agent's objective is to maximize cumulative reward. The agent learns by updating the action value estimates with the rewards it receives as it interacts with the environment. The policy improves as the estimates become more accurate since it is derived from the action value estimates, e.g., the greedy policy selects the action with the maximum action value. The agent thus learns to better perform a given task through experience as its action value estimates are updated and its policy improves. This learning can occur in the episodic setting, where a task has an ending and an agent experiences multiple episodes, and the continuing setting, where the task does not end but continues without limit. We consider the episodic setting in this paper.

A. Maximization bias in Q-learning

Maximization bias was first studied in the Q-learning algorithm [1]. Q-learning is off-policy i.e., the behaviour policy that selects actions is different from the target policy whose value the agent is learning [4]. This target policy is the greedy policy that chooses the action with the largest estimated value. The behaviour policy guarantees continued exploration of the environment, such as the ε-greedy policy. This policy selects the greedy action with a probability 1 - ε and selects a random action with ε probability. After taking action a in state s and transitioning to state s' with reward r, the agent updates its value estimates with the following rule:

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

Here $\alpha \in [0,1]$ is the learning rate that controls the size of the update and the averaging of the updates in the estimate. The discount rate $\gamma \in [0,1]$ controls the degree to which future rewards are discounted relative to current rewards.

Maximization bias in Q-learning arises from the use of the maximum action value, $max_{a'} Q(s',a')$, in the update of the estimated value. The expected value of the maximum of a set of random variables has been proven to be greater than or equal to the maximum of the set of expected values for those variables [5]. This is proven to apply to the Q-learning update [1]. The result is an overestimation in the agent's action value estimate $Q(s,a)$ when there is stochasticity in the rewards, with the possibility that some state-action pairs are more affected than others. This can cause sub-optimal actions to be taken as actions are selected based on the estimated action values, and a low-value action may look better than a high-value action due to overestimation.

B. Maximization bias in Sarsa and its variants

Sarsa was introduced as "Modified Connectionist Q-learning" [6]. As this suggests, Sarsa is similar to Q-learning, but there is a fundamental difference: Sarsa is an on-policy learning algorithm. The policy that controls the agent's behaviour is the same policy whose value is being learned. After taking an action a in state s and transitioning to state s' with reward r, the agent selects another action a' and updates its value estimates with the following rule:

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

The policy being followed and learned must balance exploration and exploitation. One approach is to learn a ε-greedy policy with a small ε. The agent selects a random action with small probability ε, but with $1-\varepsilon$ probability, it selects the greedy action, ie. $argmax_a Q(s,a)$. Thus, the greedy action value is part of most updates and we expect this to cause maximization bias.

We also study two Sarsa variants. Expected Sarsa was first introduced as "$\overline{Q}$-learning" [7] and was shown to have advantages over both Sarsa and Q-learning [8]. Expected Sarsa can be implemented on- or off-policy. We consider the on-policy case with an ε-greedy policy because our focus in this paper is maximization in on-policy algorithms. In Expected Sarsa, the agent computes the expectation of the value over all possible actions a' under the ε-greedy policy instead of sampling a single action. This expectation is $(1-\varepsilon)max_{a'} Q(s',a') + \varepsilon/|\mathcal{A}| \sum_{a'} Q(s',a')$ and we can again see the max operator in the update which causes maximization bias.

The second variant of interest is n-step Sarsa. The Q-learning and Sarsa update rules only propagate value updates back one time step. However, the agent's action in the current time step affects more than just the next reward and state. Thus we want to update the action value for the current time step, taking into account the rewards obtained in multiple future time steps. n-step Sarsa allows this by specifying the number of time steps n to use in the updates. The action selection at each time step is the same as the regular Sarsa algorithm, so we expect to see maximization bias in n-step Sarsa.

C. Dealing with maximization bias

Maximization bias causes the agent's action value estimates to overestimate the true value of the state-action pair. This can impact the learning and performance of an agent as these value estimates determine the agent's policy. For example, if the agent follows the greedy policy, then the action taken will have the highest action value. However, overestimation will not necessarily harm performance. Consider an agent with the true action values. Adding a constant to all values will cause these values to be overestimates, but the greedy policy will still select the best action as before.

Maximization bias can harm learning in practice. A Q-learning agent was shown to suffer from maximization bias and, as a result, exhibited poor performance (low average reward) and poor learning (the rewards obtained improved little over time) when deployed in a couple of simple environments with stochastic rewards [1]. The agent struggled to learn even though these were finite MDPs with a small number of states and actions.

Overestimation of action values does not necessarily lead to poor performance. Optimism in the face of uncertainty can be a legitimate strategy for an agent. For example, one method to induce exploration is "optimistic initial values"

where each action value estimate is initialized with a value that is much higher than its true value. [9]. As the agent gains experience, the value estimates become lower and more accurate in proportion to the number of visits to a state-action pair. Thus, the agent favours less frequently experienced state-action pairs as they have higher values which drive exploration and, in some cases, better learning. Algorithm designers have developed methods to control overestimation and maximization bias since they can be both harmful and beneficial.

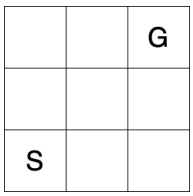
The Double Q-learning algorithm combats maximization bias by maintaining two value estimates rather than a single estimate which is, in expectation, an overestimate [1]. At each time step, one value estimate is used to find the max action value and the other value estimate is used in the actual update. This leads to underestimation rather than overestimation and performs better than Q-learning in environments designed to induce harmful maximization bias. However, just as overestimation can be harmful depending on the context, underestimation can also impede learning. In other environments, Q-learning performs better than Double Q-learning [10].

Several methods have been developed to give designers more control over the level of overestimation or underestimation. Weighted Double Q-learning uses a combination of the single Q-learning estimate, which tends to overestimate, and the Double Q-learning estimate, which tends to underestimate, in an attempt to eliminate the bias [11]. Averaged DQN uses an average of multiple value estimates to reduce variance in deep Q-learning, which reduces the maximization bias but does not eliminate it. [12]. Maxmin Q-learning was developed with the recognition that the impact of maximization bias on performance is dependent on the environment and provides a parameter to control bias [13]. When the expectation of the reward is negative with high variance, overestimation hurts performance. When the expectation of the reward is positive with high variance, overestimation helps performance. The algorithm uses the max action value of the minimum of a number of estimates in the update. As the number of estimates increases, the bias switches from an overestimate in expectation to an underestimate, so the number of estimates used is a parameter that controls the bias in a way suitable for a given environment.

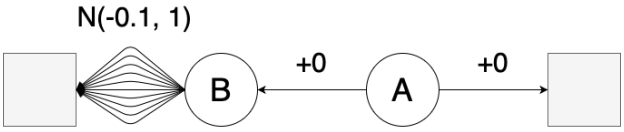
III. Experimental Setup

We conduct experiments in three environments previously used for investigating maximization bias in Q-learning. The 3x3 Grid World and Mini Roulette have discrete state spaces and can be solved with tabular methods. Mountain Car has a continuous state space so must be solved with function approximation.

a) 3x3 Grid World: In this small Grid World [1], the starting state S is at the bottom-left and the goal state G is at the top right. In each state, the agent can take one of four actions: up, down, left, and right. In all states except G, these actions result in a move one state in that direction unless doing so would move the agent outside the grid world, in which case it does not move. All actions outside of the goal state G result in a reward of +10 or -12 with equal probability. For simplicity, when introducing modifications to the reward variance later, we refer to this reward setting as (10, -12). The agent receives a reward of +5 and terminates after taking any action in G.



(a) 3x3 grid world environment. Agent begins episode in state S and can take actions up, down, left, and right. Agent receives reward of +10 or -12, each with 50% probability, on each time step except for in goal state G where every action results in +5 reward and the termination of the episode.



(b) Mini roulette environment. Agent begins episode in state A and can take action right, terminating the episode with 0 reward, or left, transitioning to state B with 0 reward. In B, the agent can take one of 10 actions which each terminate the epsiode with a reward randomly sampled from a Normal distribution of mean -0.1 and variance 1.

Fig. 1: The two environments used for the tabular setting.

b) Mini Roulette: This environment [9] consists of two non-terminal states, A and B. The agent begins in state A where it can take one of two actions, left or right. The right action yields a reward of 0, and the episode terminates. The left action moves the agent to state B with a reward of 0. From B, the agent moves to the terminal state with any one of ten actions. The reward for all ten actions is drawn from a Normal distribution with mean -0.1 and variance 1.

c) Mountain Car: In this environment the agent learns to drive an under-powered car up a mountain. It has a continuous two-dimensional state space with velocity bounded between -0.07 and 0.07 and position between -1.2 and 0.6. There are three actions: accelerate left, accelerate right, do nothing. We use the AI Gym implementation 'MountainCar-v0' with two changes. We modify the reward of -1 per time step by adding randomly sampled noise to induce maximization bias. And the agent does not update its value function if an episode is cutoff rather than ending at the goal state.

IV. Results

In this section, we present the results of our experiments and the answers to our research questions.

A. Experiments in the 3x3 Grid World

Previous work in this environment ran for 10,000 steps with $\gamma = 0.95$, $\varepsilon = 1/\sqrt{n(s)}$ where $n(s)$ is the number of times state s has been visited, and a decaying α of either $1/n(s,a)$ or $1/n(s,a)^{0.8}$ where $n(s,a)$ is the number of times action

a has been taken in state s [1]. We use these settings as a starting point in our investigation.

We extend this by performing experiments with different types and values for the hyper-parameters. We perform runs with decaying learning rates $\alpha = 1/n(s,a)^p$ where $p \in [0,1]$. We also perform runs with fixed learning rates where $\alpha \in [0,1]$. We perform runs with various fixed $\varepsilon \in [0,1]$. As in previous work, we use the maximum action value for the start state $max_a Q(S_0, A)$ as a measure of the maximization bias itself and the average reward as a measure of the impact of maximization bias on performance. We experiment with different values of n for n-step Sarsa. We also investigate the impact of environment dynamics by experimenting with different levels of randomness in the rewards. All experiments are averaged over 200 runs for each hyper-parameter setting.

B. Results in the 3x3 Grid World

Do Sarsa and its variants suffer from maximization bias and how does this compare to Q-learning?

We know from the definition of maximization bias and the Sarsa update that Sarsa should experience maximization bias. We want to understand how the severity of this bias differs between on-policy and off-policy algorithms in different environments and for different hyper-parameter settings. For a fair comparison in the 3x3 Grid World, we determined the best decaying learning rate for each algorithm and compared their average reward and maximum action value of the start state in figure 2.

We observe that the $max_a Q(S_0, a)$ is overestimated by orders of magnitude during early learning for Sarsa, Expected Sarsa, and 3-step Sarsa. The true value of $max_a Q(S_0, a)$ is 0.36, but these on-policy algorithms estimate a value closer to 10, which indicates that they suffer from maximization bias. However, the overestimation is much less than that of Q-Learning, whose estimate reaches around 30. The less severity of the bias is reflected in the average reward, which is higher for the Sarsa variants. Expected Sarsa and Sarsa perform similarly, and 3-step Sarsa performs best, reflecting the advantage that multi-step methods can have over single-step algorithms. [9].

What is the primary reason for the large difference between Q-learning and the Sarsa variants? Both $\gamma = 0.95$ and $\varepsilon = 1/\sqrt{n(s)}$ for all algorithms. The decaying α used is the best performing for each algorithm. Q-learning is indeed learning the greedy policy while the Sarsa variants are learning the ε-greedy policy. However, ε decays and approaches 0 so the policies being learned converge. In certain environments, the optimal greedy policy and optimal ε-greedy policy can be different, such as Cliff-Walking and Windy Grid World [8]. However, optimal policies in both of our test environments will be the same: take the shortest path to the goal state in the 3x3 Grid World and take the right action in the start state in the Small MDP.

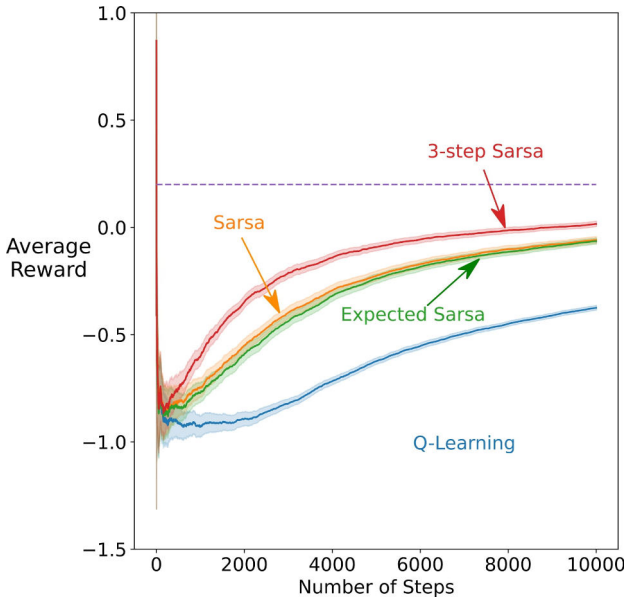
We hypothesize that the cause of the difference is the algorithm being on-policy versus off-policy. In the off-policy Q-learning algorithm, the update uses the maximum action value of the next state for every update. When the agent is suffering from maximization bias, these maximum action values are overestimates. In contrast, the on-policy algorithms incorporate non-greedy action values in their updates. In Sarsa and n-step Sarsa, with ε probability, a non-greedy action value of the next state is used in the update target. In Expected Sarsa, the next state's expected value contains both greedy and non-greedy action values. These non-greedy action values are less than or equal to the greedy action, but since the agent is overestimating, incorporating these lower values into the estimate moves the value estimate closer to the actual lower value.

What is the impact of the learning rate α on maximization bias in Sarsa and its variants?

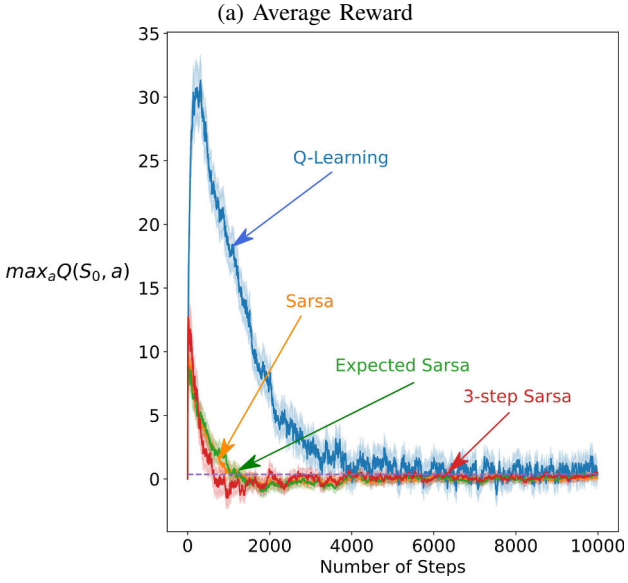
To determine the impact of α on the severity of maximization bias, we ran sweeps over various values of both fixed and decaying α. Figure 3 shows the learning rate sweep for Q-learning and Sarsa. The most interesting result of these experiments is the big difference in the bias between decaying learning rates and fixed learning rates. We can see that the largest overestimated values belong to the decaying α. The fixed α also overestimate initially, but the magnitude of the bias is dwarfed by that of the decaying α. We initially tested a range of reasonable fixed α against decaying rates of $1/n(s,a)$ and $1/n(s,a)^{0.8}$ which were previously used in the work on maximization bias in Q-learning [1]. After seeing the gap in performance between these two types of learning rates, we extended our analysis to include more decaying learning rates.

Figure 4 summarizes these results for Sarsa and shows the sensitivity of its performance on the 3x3 Grid World for different fixed and decaying learning rates. The sensitivity plot shows that the impact of maximization bias on algorithm performance is much more sensitive to the decaying learning rate than the fixed learning rate. For learning rates that decay more quickly, such as $\alpha = 1/n(s,a)^{0.8}$, the overestimation of $max_a Q(S_0, a)$ is higher. During early learning, the agent suffers from maximization bias as it has yet to gain enough experience to overcome the distorting effect of a few randomly high rewards. For decaying learning rate schedules, the value of α is highest during these initial time steps, which causes the bias to be large. As the agent continues to gain experience in the environment, its value estimates improve, resulting in a reduction in bias. However, for decaying α, these later updates are relatively small compared to the earlier updates. If α decays quickly, this later learning is very slow. If α decays slowly, the agent can overcome this bias more quickly since the size of the later updates will not be too small relative to the earlier updates. However, this slow decay leads to a large value of α even after the bias has been overcome, resulting in noisy updates. Thus, slower decay also leads to higher RMSE after 10,000 steps. Experiments with n-step Sarsa and Expected Sarsa showed similar results.

Decaying learning rates have the theoretical advantage of convergence in the limit for stationary problems. This is not true of fixed learning rates. However, this lack of convergence is actually a desirable property when the environment is non-

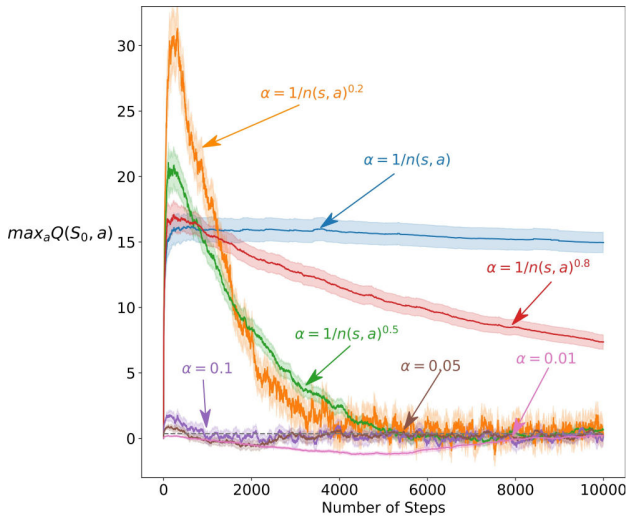


(a) Average Reward

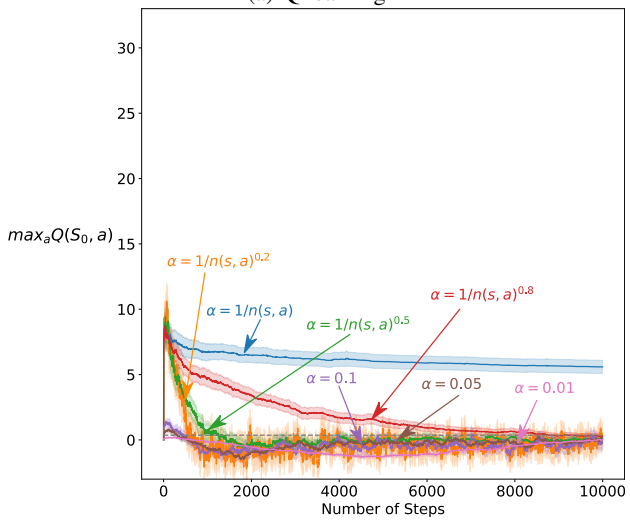


(b) Max Action Value of start state

Fig. 2: Average reward and max action value of start state for all algorithms with the best performing decaying alpha in the 3x3 Grid World environment. Q-learning $\alpha = 1/n(s,a)^{0.2}$, Sarsa $\alpha = 1/n(s,a)^{0.5}$, Expected Sarsa $\alpha = 1/n(s,a)^{0.5}$, 3-step Sarsa $\alpha = 1/n(s,a)^{0.5}$. We use $\varepsilon = 1/\sqrt{n(s)}$ as there is not large difference in performance between the fixed and decaying ε and with the decaying ε the policies being learned converge. Error bars plotted with +/-2 standard error.



(a) Q-learning



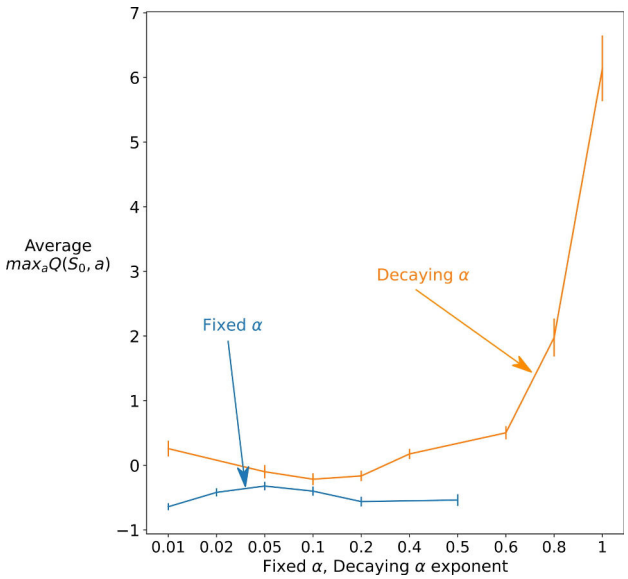
(b) Sarsa

Fig. 3: Max action value of start state for Q-learning and Sarsa in 3x3 Grid World environment for various α. $\varepsilon = 1/\sqrt{n(s)}$. Error bars plotted with +/-2 standard error.

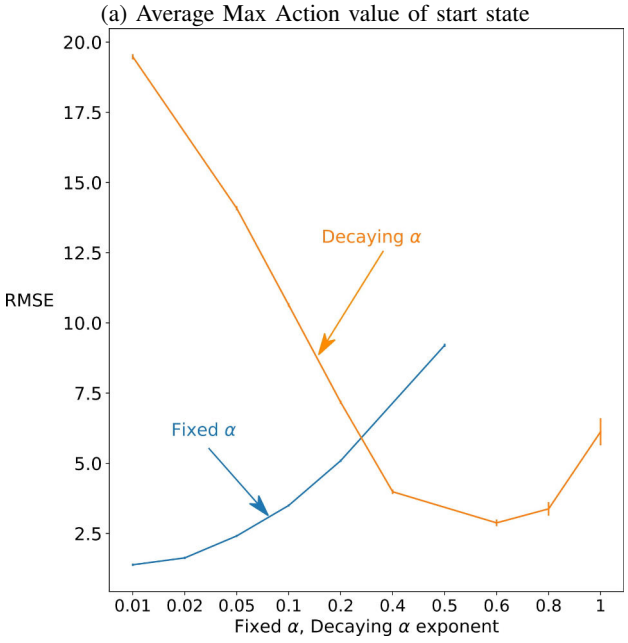
stationary, which is the more common situation in reinforcement learning [9]. Our results show that maximization bias is less severe and thus less of a problem when using a fixed learning rate which is more common in practice than the decaying learning rates originally used to demonstrate this bias.

What is the impact of the exploration parameter ε on maximization bias in Sarsa and its variants?

To investigate the impact of ε, we report the performance of all algorithms for different values of ε with $\alpha = 1/n(s,a)^{0.8}$. Figure 5 shows these plots for Q-learning and Sarsa. We observe that higher ε values are detrimental to Q-Learning's performance, whereas higher ε values improve Sarsa's performance. Higher ε values lead to more exploration, which means that the agent visits less frequently visited states more often. For such states, the α value would be large since $n(s,a)$ would be lower and the *maximum* action values may be overestimates. Thus this greater exploration increases the severity of the maximization bias. For Sarsa, high ε values lead to a higher probability of using a non-greedy action value in the update since it is an on-policy algorithm. As previously discussed, these non-max updates may help overcome the bias as they are less likely to use overestimated values. Expected Sarsa and



(a) Average Max Action value of start state



(b) RMSE

Fig. 4: Learning rate sensitivity plot for Sarsa in 3x3 Grid World. The x-axis is the rate itself for fixed learning rates, i.e. $\alpha = x$, and is the exponent for decaying learning rates, i.e. $\alpha = 1/n(s,a)^x$. The left plot shows the average max action value of start state. The right plot shows the RMSE of estimated value with respect to the optimal max action value of start state ($max_a Q_*(S_0, a) = 0.36$). Concave shape of fixed α in left plot result of large α learning and overcoming bias quickly, small α learning too slowly to even suffer from bias, and α in-between suffering from bias but overcoming it more slowly so staying overestimated for more of the 10,000 steps.

n-step Sarsa exhibit similar behaviour.

What is the impact of reward variance on maximization bias in Sarsa and its variants?
To investigate the impact of reward variance, we modify the 3x3 Grid World such that the reward obtained in each non-terminating step is given by $(10 + x, -12 - x)$ where $x \in \{0, 30, 60\}$. Modifying the reward in this way ensures that $\mathbb{E}[R_t] = -1$ for all non-terminating time steps. We perform a hyper-parameter sweep for each reward setting to find the best fixed learning rate for each algorithm in terms of $max_a Q(S_0, a)$. We observe that increasing reward variance leads to greater maximization bias for both algorithms and lower average reward, and this bias is greater for Q-learning than Sarsa. The results for reward are less significant, which reflects even greater amounts of bias can still be overcome by the last step of the runs. It is notable that for lower learning rates, the average $max_a Q(S_0, a)$ is closer to the optimal value for all reward settings. This is simply due to the algorithm learning very slowly, not that it has suffered and overcome the bias.

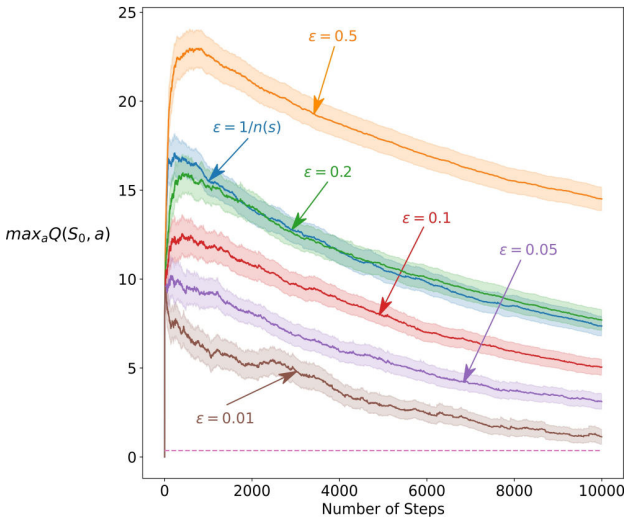
C. Experiments with Mini Roulette

We again use the hyper-parameter settings of previous work as a starting point with $\gamma = 1$ $\varepsilon = 0.1$ and $\alpha = 0.1$ [9]. We don't limit ourselves to these settings and conduct experiments over a range hyper-parameters to help answer our questions. A run consists of 300 episodes. We perform 1000 runs in this environment. Given the simplicity of the environment, the measure of maximization bias we use is the proportion of times the left action (which has negative expected value but can look good due to maximization bias) was taken in the start state.

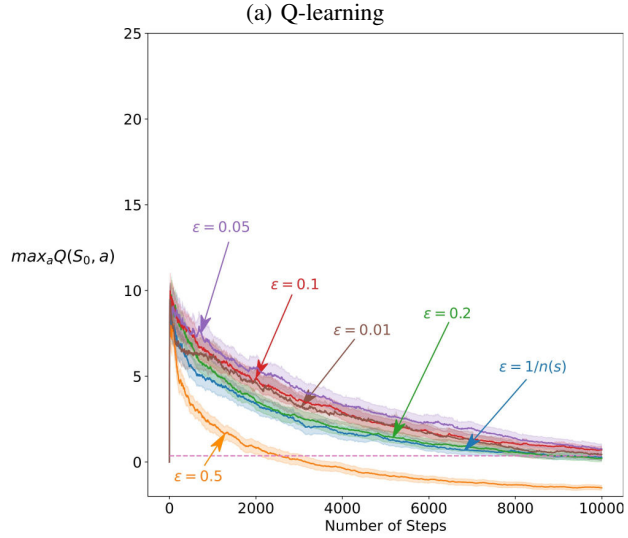
D. Results with Mini Roulette

Do Sarsa and its variants suffer from maximization bias and how does this compare to Q-learning?
We see a similar pattern in the Mini Roulette environment as in the 3x3 Grid World with Sarsa and its variants suffering less bias than Q-learning. The optimal action is to go right in the start state, and for the first step the proportion of left actions is 50% since both action values are initialized to 0. The proportion of left actions exceeds 80% for all tested learning rates for Q-learning before the agent learns to go right. The Sarsa agent does not hit a left action proportion of over 80% for any learning rate, and the best α maxes out at 65%. Expected Sarsa and 2-step Sarsa perform even better.

What is the impact of the learning rate α on maximization bias in Sarsa and its variants?
We swept over the same α used in the 3x3 Grid World. The results we see with Mini Roulette are consistent with those from the 3x3 Grid World. The worst performing α are the decaying rates that decay too quickly, and very small fixed rates. The quickly decaying learning rates learn more early on when suffering from maximization bias and relatively little later on when the bias must be overcome. The very small fixed rates learn too slowly.



(a) Q-learning



(b) Sarsa

Fig. 5: Max action value of start state for Q-learning and Sarsa in 3x3 Grid World environment for various ε. $\alpha = 1/n(s,a)^{0.8}$. Using this α causes large overestimation which makes it easier to see the differences between the ε. Error bars plotted with +/-2 standard error.

What is the impact of the exploration parameter ε on maximization bias in Sarsa and its variants?
We swept over the same ε used in the 3x3 Grid World experiments. The results for Sarsa and its variants are consistent with the previous experiments. Greater ε correspond to better initial performance, although the smaller values eventually outperform as they correspond to better policies. This supports our hypothesis that including the random actions in updates help overcome maximization bias which explains the difference in bias between on-policy and off-policy algorithms. However, the results for Q-learning are the opposite of what we found in the 3x3 Grid World. Here, smaller ϵ exhibit more bias. More experimentation is needed to confirm or falsify our hypothesis.

E. Experiments with Mountain Car

We use the Mountain Car environment to extend our analysis to a setting more similar to those found in practice. Mountain Car requires function approximation. We use a tile coding of eight 8x8 tilings. We compare Q-learning and Sarsa both with a fixed $\alpha = 0.1/8$ and the Adam optimizer for learning rate adaptation. Adam hyper-parameters used are $\epsilon = 10^{-8}$, $\beta_m = 0.9$, $\beta_v = 0.999$, and step-size = 0.001. For both, we test using the standard -1 reward per step as a baseline and then with -1 + x where x is sampled from a normal distribution of mean 0 and variance 10. This random reward is used to induce maximization bias. All agents have $\epsilon = 0.1$. Each experiment is averaged over 200 runs of 1000 episodes, where an episode is cutoff after 200 steps. We use reasonable hyper-parameters values rather than comprehensive sweeps as these experiments are just a first step into the function approximation domain rather than a thorough investigation.

F. Results with Mountain Car

We can see in figure 6a that Q-learning and Sarsa have the same performance on constant reward Mountain Car and both are similarly negatively impacted when random rewards and thus bias are introduced. The curves for Q-learning and Sarsa with random reward seem to diverge slightly in later episodes but with overlapping error bars. Since an actual difference in performance would be evidence for our on-policy vs. off-policy hypothesis, we conducted additional experiments with more episodes. However, the difference did not become significant by 1500 episodes, so we conclude that this small divergence is due to randomness.

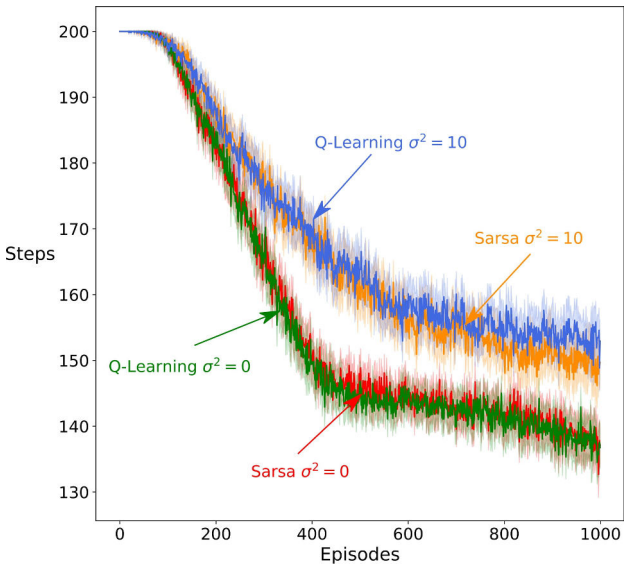
We can see from figure 6b that the pattern is similar when the algorithms use the Adam optimizer for step size adaptation. Q-learning and Sarsa have the same performance when rewards are constant or random. However, the gap in performance between the reward regimes is much larger than when the agents use a fixed α. The performance with constant rewards indicates that the Adam hyper-parameter choices are not bad for learning. However, without proper tuning, we cannot say that maximization bias is worse when using the Adam optimizer. We can conclude that maximization bias is still a concern in the function approximation setting with both the fixed α and the Adam optimizer, which is more common in practice than the tabular setting and decaying learning rates.

ACKNOWLEDGMENT

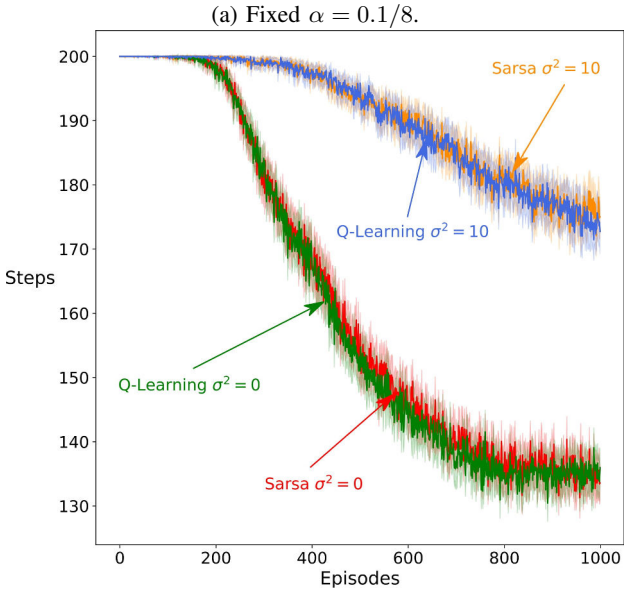
We thank Dr. Adam White and Andrew Jacobsen for providing constructive feedback on our experiments.

EXTENSIONS AND FUTURE WORK

More work is needed to confirm or falsify our hypothesis that on-policy algorithms are less susceptible to maximization bias. This could be done using off-policy Expected Sarsa with varying distances between the behaviour and target policies to control the degree of off-policy learning. A more thorough investigation of maximization bias is needed in the function approximation setting, including with neural networks. Finally,



(a) Fixed $\alpha = 0.1/8$.



(b) Adam optimizer.

Fig. 6: Q-learning and Sarsa algorithms run on the Mountain Car environment with both a fixed learning rate and using the Adam optimizer. The curves labelled with $\sigma^2 = 0$ correspond to standard Mountain Car with reward of -1 per step while $\sigma^2 = 10$ corresponds to reward of -1 plus Gaussian noise of mean 0 and variance 10. Plotted with error bars of +/-2 standard error.

proper hyper-parameter tuning for the Adam optimizer on more environments is needed to properly establish whether it is more or less robust to maximization bias than fixed α.

Conclusion

Our empirical investigation into maximization bias in Sarsa and its variants has revealed a potential fundamental difference in how on-policy and off-policy algorithms are affected by the bias in our two test environments. Off-policy Q-learning suffers from worse bias than on-policy Sarsa and its variants in our tabular setting experiments. We have shown that the choice of learning rate α can have a significant impact on the severity of the bias and that using the Adam optimizer does not prevent the bias. Furthermore, we have shown that the level of exploration ε can impact the bias in different ways for on-policy and off-policy algorithms, with on-policy algorithms benefiting from and off-policy algorithms disadvantaged by larger ε in some environments. Our results demonstrate that Sarsa also suffers from worse maximization bias as the reward variance increase, but that this increase in bias is less severe than for Q-learning.

References

[1] H. van Hasselt, "Double q-learning," *Advances in Neural Information Processing Systems*, vol. 23, pp. 2613–2621, 2010.

[2] M. Ganger, E. Duryea, and W. Hu, "Double sarsa and double expected sarsa with shallow and deep learning," *Journal of Data Analysis and Information Processing*, vol. 4, no. 4, pp. 159–176, 2016.

[3] M. L. Puterman, "Markov decision processes," *Handbooks in Operations Research and Management Science*, vol. 2, pp. 331–434, 1990.

[4] C. J. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[5] J. E. Smith and R. L. Winkler, "The optimizer's curse: Skepticism and postdecision surprise in decision analysis," *Management Science*, vol. 52, no. 3, pp. 311–322, 2006.

[6] G. Rummery and M. Niranjan, "On-line q-learning using connectionist systems," *Technical Report CUED/F-INFENG/TR 166*, 1994.

[7] G. John, "When the best move isn't optimal: Q-learning with exploration," *Proceedings of the Twelfth National Conference on Artificial Intelligence*, 1994.

[8] H. van Seijen, H. van Hasselt, S. Whiteson, and M. Wiering, "A theoretical and empirical analysis of expected sarsa," *2009 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning*, pp. 177–184, 2009.

[9] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. The MIT Press, 2018. [Online]. Available: http://incompleteideas.net/book/the-book-2nd.html

[10] H. van Hasselt, "Insights in reinforcement learning: Formal analysis and empirical evaluation of temporal-difference learning algorithms," Ph.D. dissertation, SIKs, the Dutch Research School for Information and Knowledge Systems, 2010.

[11] Z. Zhang, Z. Pan, and M. J. Kochenderfer, "Weighted double q-learning," *International Joint Conference on Artificial Intelligence*, pp. 3455–3461, 2017.

[12] O. Anschel, N. Baram, and N. Shimkin, "Averaged-dqn: Variance reduction and stabilization for deep reinforcement learning," *International Conference on Machine Learning*, pp. 176–185, 2017.

[13] Q. Lan, Y. Pan, A. Fyshe, and M. White, "Maxmin q-learning: Controlling the estimation bias of q-learning," *8th International Conference on Learning Representations*, 2020.