A Theoretical and Empirical Analysis of Expected Sarsa

Harm van Seijen, Hado van Hasselt, Shimon Whiteson and Marco Wiering

Abstract— **This paper presents a theoretical and empirical analysis of Expected Sarsa, a variation on Sarsa, the classic on-policy temporal-difference method for model-free reinforcement learning. Expected Sarsa exploits knowledge about stochasticity in the behavior policy to perform updates with lower variance. Doing so allows for higher learning rates and thus faster learning. In deterministic environments, Expected Sarsa's updates have zero variance, enabling a learning rate of 1. We prove that Expected Sarsa converges under the same conditions as Sarsa and formulate specific hypotheses about when Expected Sarsa will outperform Sarsa and Q-learning. Experiments in multiple domains confirm these hypotheses and demonstrate that Expected Sarsa has significant advantages over these more commonly used methods.**

I. Introduction

In *reinforcement learning* (RL) [1], [2], an agent seeks an optimal control policy for a sequential decision problem. Unlike in supervised learning, the agent never sees examples of correct or incorrect behavior. Instead, it receives only positive and negative rewards for the actions it tries. Since many practical, real world problems (such as robot control, game playing, and system optimization) fall in this category, developing effective RL algorithms is important to the progress of artificial intelligence.

When the sequential decision problem is modeled as a *Markov decision process* (MDP) [3], the agent's policy can be represented as a mapping from each state it may encounter to a probability distribution over the available actions. In some cases, the agent can use its experience interacting with the environment to estimate a model of the MDP and then compute an optimal policy via off-line planning techniques such as dynamic programming [4].

When learning a model is not feasible, the agent can still learn an optimal policy using *temporal-difference* (TD) methods [5]. Each time the agent acts, the resulting feedback is used to update estimates of its action-value function, which predicts the long-term discounted reward it will receive if it takes a given action in a given state. Under certain conditions, TD methods are guaranteed to converge in the limit to the optimal action-value function, from which an optimal policy can easily be derived.

Harm van Seijen is with the Integrated Systems group, TNO Defense, Safety and Security, The Hague (email: harm.vanseijen@tno.nl). Hado van Hasselt is with the Intelligent Systems Group, Utrecht University, Utrecht (email: hado@cs.uu.nl). Shimon Whiteson is with the Intelligent Autonomous Systems Group, University of Amsterdam, Amsterdam (email:s.a.whiteson@uva.nl) and Marco Wiering is with the Department of Artificial Intelligence, University of Groningen, Groningen (email: mwiering@ai.rug.nl)

The research reported here is part of the Interactive Collaborative Information Systems (ICIS) project, supported by the Dutch Ministry of Economic Affairs, grant nr: BSIK03024.

In *off-policy* TD methods such as Q-learning [6], the *behavior policy*, used to control the agent during learning, is different from the *estimation policy*, whose value is being learned. The advantage of this approach is that the agent can employ an exploratory behavior policy to ensure it gathers sufficiently diverse data while still learning how to behave once exploration is no longer necessary. However, an *on-policy* approach, in which the behavior and estimation policies are identical, also has important advantages. In particular, it has stronger convergence guarantees when combined with function approximation, since off-policy approaches can diverge in that case [7], [8], [9] and it has a potential advantage over off-policy methods in its *on-line* performance, since the estimation policy, that is iteratively improved, is also the policy that is used to control its behavior. By annealing exploration over time, on-policy methods can discover the same policies in the limit as off-policy approaches.

The classic on-policy TD method is Sarsa [10], [11], which is named for the five components employed in its update rule: the current state and action s_t and a_t, the immediate reward r, and the next state and action s_{t+1} and a_{t+1}. The use of a_{t+1} introduces additional variance into the update when the estimation policy is stochastic, as is typically the case for on-policy methods like Sarsa. This additional variance can slow convergence. For this reason, Sutton and Barto proposed, in a little-noted exercise in their classic book [2, Exercise 6.10], a variation on Sarsa designed to reduce variance in the updates. Instead of simply using a_{t+1}, this variation computes an expectation over all actions available in s_{t+1}. Though the resulting algorithm, which we call Expected Sarsa, may offer substantial advantages over Sarsa, it has never been systematically studied and is not widely used in practice.

In this paper, we present a theoretical and empirical analysis of Expected Sarsa. On the theoretical side, we show that Expected Sarsa shares the same convergence guarantees as Sarsa and thus finds the optimal policy in the limit under certain conditions. We also show that Expected Sarsa has lower variance in its updates than Sarsa and demonstrate which factors contribute to this gap.

On the empirical side, we compare the performance of Expected Sarsa with the performance of Sarsa and Q-learning. We formulate two hypotheses about the performance difference between Expected Sarsa and these two methods and confirm them using two benchmark problems: the cliff walking problem and the windy grid world problem. We also present results in additional domains verifying the advantages of Expected Sarsa in a broader setting.

II. Background

The sequential decision problems addressed in RL are often formalized as MDPs, which can be described as 4-tuples $\langle S, A, T, R\rangle$ where

- $\mathcal{S}$ is the set of all states the agent can encounter,
- $\mathcal{A}$ is the set of all actions available,
- $\mathcal{T}(s,a,s') = P(s'|s,a)$ is the transition function, and
- $\mathcal{R}(s,a,s') = E(r|s,a,s')$, is the reward function.

The goal of the agent is to find an optimal policy $\pi^* = P(a|s)$, which maximizes the expected discounted return:

$$R_t = r_{t+1} + \gamma\, r_{t+2} + \gamma^2\, r_{t+3} + ... = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \quad (1)$$

where γ is a discount factor with $0 \leq \gamma \leq 1$.

All TD algorithms are based on estimating *value functions*. The *state-value function* $V^\pi(s)$ gives the expected return when the agent is in state s and follows policy π. The *action-value function* $Q^\pi(s,a)$ gives the expected return when the agent takes action a in state s and follows policy π thereafter. These two functions are related through

$$V^\pi(s) = \sum_a \pi(s,a)\, Q^\pi(s,a) \quad (2)$$

TD methods seek the optimal action-value function $Q^*(s,a)$, from which an optimal policy π^* can easily be deduced. $Q^*(s,a)$ can be found by iteratively updating the estimate $Q(s,a)$.

The off-policy method Q-learning updates its Q values using the update rule

$$\begin{aligned} Q(s_t,a_t) \quad \leftarrow \quad & Q(s_t,a_t) + \alpha\,[r_{t+1} + \\ & \gamma\, max_a\, Q(s_{t+1},a) - Q(s_t,a_t)] \end{aligned} \quad (3)$$

The *max* operator causes the estimation policy to be greedy, which guarantees the Q values converge to $Q^*(s,a)$. The behavior policy of Q-learning is usually exploratory and based on $Q(s,a)$.

For Sarsa the behavior policy and the estimation policy are equal. The update rule of Sarsa is

$$\begin{aligned} Q(s_t,a_t) \quad \leftarrow \quad & Q(s_t,a_t) + \alpha\,[r_{t+1} + \\ & \gamma\, Q(s_{t+1},a_{t+1}) - Q(s_t,a_t)] \end{aligned} \quad (4)$$

Because Sarsa is on-policy, it will not converge to optimal Q values as long as exploration occurs. However, by annealing exploration over time, Sarsa can converge to optimal Q values, just like Q-learning.

III. Expected Sarsa

Since Sarsa's convergence guarantee requires that every state be visited infinitely often, the behavior and therefore also the estimation policy is typically stochastic so as to ensure sufficient exploration. As a result, there can be substantial variance in Sarsa updates, since a_{t+1} is not selected deterministically.

Of course, variance can occur in updates for any TD method because the environment can introduce stochasticity through $\mathcal{T}$ and $\mathcal{R}$. Since TD methods are typically used when a model of the environment is not available, there is little the agent can do about this stochasticity except employ a suitably low α. However, the additional variance introduced by Sarsa stems from the policy stochasticity, which is known to the agent.

Expected Sarsa is a variation of Sarsa which exploits this knowledge to prevent stochasticity in the policy from further increasing variance. It does so by basing the update, not on $Q(s_{t+1},a_{t+1})$, but on its expected value $E\{Q(s_{t+1},a_{t+1})\}$. The resulting update rule is:

$$\begin{aligned} Q(s_t,a_t) \quad \leftarrow \quad & Q(s_t,a_t) + \alpha\,[r_{t+1} + \\ & \gamma \sum_a \pi(s_{t+1},a)\, Q(s_{t+1},a) - Q(s_t,a_t)] \end{aligned} \quad (5)$$

Using this expectation reduces the variance in the update, as we show formally in Section V. Lower variance means that in practice α can often be increased in order to speed learning, as we demonstrate empirically in Section VII. In fact, when the environment is deterministic, Expected Sarsa can employ $\alpha = 1$, while Sarsa still requires $\alpha < 1$ to cope with policy stochasticity.

Algorithm 1 shows the complete Expected Sarsa algorithm. Because the update rule of Expected Sarsa, unlike Sarsa, does not make use of the action taken in s_{t+1}, action selection can occur *after* the update. Doing so can be advantageous in problems containing states with returning actions, i.e. $P(s_{t+1} = s_t) > 0$. When $s_{t+1} = s_t$, performing an update of $Q(s_t,a_t)$, will also update $Q(s_{t+1},a_t)$, yielding a better estimate before action selection occurs.

Algorithm 1 Expected Sarsa

```
1: Initialize $Q(s,a)$ arbitrarily for all $s$,$a$
2: **loop** {over episodes}
3:   Initialize s
4:   **repeat** {for each step in the episode}
5:     choose $a$ from $s$ using policy $\pi$ derived from Q
6:     take action $a$, observe $r$ and $s'$
7:     $V_{s'} = \sum_a \pi(s',a) \cdot Q(s',a)$
8:     $Q(s,a) \leftarrow Q(s,a) + \alpha\,[r + \gamma\, V_{s'} - Q(s,a)]$
9:     $s \leftarrow s'$
10:  **until** $s$ is terminal
11: **end loop**
```

Expected Sarsa can also be viewed, not as a lower-variance version of Sarsa, but as an on-policy version of Q-learning. Note the similarity between the expectation value $E\{Q(s_{t+1},a_{t+1})\}$ used by Expected Sarsa and (2) relating $V^\pi(s)$ to $Q^\pi(s,a)$. Since $Q(s,a)$ is an estimate of $Q^\pi(s,a)$, its expectation value can be seen as the estimate $V(s)$ for $V^\pi(s)$ using the relation:

$$V(s) = \sum_a \pi(s,a)\, Q(s,a) \quad (6)$$

If the policy π is greedy, $\pi(s,a) = 0$ for all a except for the action for which Q has its maximal value. Therefore, in the case of a greedy policy, (6) simplifies to

$$V(s) = \max_a Q(s,a) \tag{7}$$

Thus, Q-learning's update rule (3) is just a special case of Expected Sarsa's update rule (5) for the case when the estimation policy is greedy. Nonetheless, the complete Expected Sarsa algorithm is different from that of Q-learning because the former is on-policy and the latter is off-policy.

IV. Convergence

In this section, we prove that Expected Sarsa converges to the optimal policy under some straightforward conditions given below. We make use of the following Lemma, which was also used to prove convergence of Sarsa [12]:

Lemma 1: Consider a stochastic process (ζ_t, Δ_t, F_t), where $\zeta_t, \Delta_t, F_t : X \rightarrow \mathbb{R}$ satisfy the equations

$$\Delta_{t+1}(x_t) = (1-\zeta_t(x_t))\Delta_t(x_t) + \zeta_t(x_t)F_t(x_t) \ ,$$

where $x_t \in X$ and $t = 0, 1, 2, \ldots$. Let P_t be a sequence of increasing σ-fields such that ζ_0 and Δ_0 are P_0-measurable and ζ_t, Δ_t and F_{t-1} are P_t-measurable, $t \geq 1$. Assume that the following hold:

1) the set X is finite,
2) $\zeta_t(x_t) \in [0,1]$, $\sum_t \zeta_t(x_t) = \infty$, $\sum_t (\zeta_t(x_t))^2 < \infty$ w.p.1 and $\forall x \neq x_t : \zeta_t(x) = 0$,
3) $||E\{F_t|P_t\}|| \leq \kappa||\Delta_t|| + c_t$, where $\kappa \in [0,1)$ and c_t converges to zero w.p.1,
4) $\text{Var}\{F_t(x_t)|P_t\} \leq K(1+\kappa||\Delta_t||)^2$, where K is some constant,

where $||\cdot||$ denotes a maximum norm. Then Δ_t converges to zero with probability one.

The idea is to apply Lemma 1 with $X = S \times A$, $P_t = \{Q_0, s_0, a_0, r_0, \alpha_0, s_1, a_1, \ldots, s_t, a_t\}$, $x_t = (s_t, a_t)$, $\zeta_t(x_t) = \alpha_t(s_t, a_t)$ and $\Delta_t(x_t) = Q_t(s_t, a_t) - Q^*(s_t, a_t)$. If we can then prove that Δ_t converges to zero with probability one, we have convergence of the Q values to the optimal values. The maximum norm specified in the lemma can then be understood as satisfying the following equation:

$$||\Delta_t|| = \max_s \max_a |Q_t(s,a) - Q^*(s,a)| \tag{8}$$

Theorem 1: Expected Sarsa as defined by update (5) converges to the optimal value function whenever the following assumptions hold:

1) S and A are finite,
2) $\alpha_t(s_t, a_t) \in [0,1]$, $\sum_t \alpha_t(s_t, a_t) = \infty$, $\sum_t (\alpha_t(s_t, a_t))^2 < \infty$ w.p.1 and $\forall (s,a) \neq (s_t, a_t) : \alpha_t(s,a) = 0$,
3) The policy is greedy in the limit with infinite exploration,
4) The reward function is bounded.

Proof: To prove this theorem, we simply check that all the conditions of Lemma 1 are fulfilled. The first, second and fourth conditions of this lemma correspond to the first, second and fourth assumptions of the theorem. Below, we will show the third condition of the lemma holds.

We can derive the value of F_t as follows:

$$\begin{aligned} F_t &= \frac{1}{\alpha_t}\Big(\Delta_{t+1} - (1-\alpha_t)\Delta_t\Big) \ , \\ &= r_t + \gamma \sum_a \pi_t(s_{t+1}, a) Q_t(s_{t+1}, a) - Q^*(s_t, a_t) \ , \end{aligned}$$

where all the values are taken over the state action pair (s_t, a_t), except when specified differently.

If we can show that $||E\{F_t\}|| \leq \kappa||\Delta_t|| + c_t$, where $\kappa \in [0,1)$ and c_t converges to zero, all the conditions of the lemma can be fulfilled and we have convergence of Δ_t to zero and therefore convergence of Q_t to Q^*. We derive this as follows:

$$\begin{aligned} & ||E\{F_t\}|| \\ = \ & ||E\{r_t + \gamma \sum_a \pi_t(s_{t+1}, a) Q_t(s_{t+1}, a) - Q^*(s_t, a_t)\}|| \\ \leq \ & ||E\{r_t + \gamma \max_a Q_t(s_{t+1}, a) - Q^*(s_t, a_t)\}|| + \\ & \gamma ||E\{\sum_a \pi_t(s_{t+1}, a) Q_t(s_{t+1}, a) - \max_a Q_t(s_{t+1}, a)\}|| \\ \leq \ & \gamma \max_s \Big| \max_a Q_t(s,a) - \max_a Q^*(s,a) \Big| + \\ & \gamma \max_s \Big| \sum_a \pi_t(s,a) Q_t(s,a) - \max_a Q_t(s,a) \Big| \\ \leq \ & \gamma ||\Delta_t|| + \\ & \gamma \max_s \Big| \sum_a \pi_t(s,a) Q_t(s,a) - \max_a Q_t(s,a) \Big| \ , \end{aligned}$$

where the second inequality results from the definition of Q^* and the fact that the maximal difference in value over all states is always at least as large as a difference between values corresponding to a state s_{t+1}. The third inequality follows directly from (8). The other (in)equalities are based on algebraic rewriting or definitions.

We identify $c_t = \gamma \max_s |\sum_a \pi_t(s,a) Q_t(s,a) - \max_a Q_t(s,a)|$ and $\kappa = \gamma$. Clearly, c_t converges to zero for policies that are greedy in the limit. Therefore, if $\gamma < 1$, all of the conditions of Lemma 1 follow from the assumptions in the present theorem and we can apply the lemma to prove convergence of Q_t to Q^*. ∎

V. Variance Analysis

Section IV shows that Expected-Sarsa converges to the optimal policy under the same conditions as Sarsa. In this section, we further analyze the behavior of the two methods to show theoretically under what conditions Expected-Sarsa will in some sense perform better. Specifically, we show that both algorithms have the same bias and that the variance of Expected-Sarsa is lower. Finally, we describe which factors affect this difference in variance. In this section, we use $v_t = r_t + \gamma \sum_a \pi_t(s_{t+1}, a) Q_t(s_{t+1}, a)$ and $\hat{v}_t = r_t + \gamma Q_t(s_{t+1}, a_{t+1})$ to denote the target of Expected-Sarsa and Sarsa, respectively.

The bias of the updates of both algorithms under a certain policy π is given by the following equation:

$$Bias(s,a) = Q^\pi(s,a) - E\{X_t\} \tag{9}$$

where X_t is either v_t or $\hat{v}_t$. Both algorithms have the same bias, since $E\{v_t\} = E\{\hat{v}_t\}$. The variance is then given by:

$$Var(s,a) = E\{(X_t)^2\} - (E\{X_t\})^2 \tag{10}$$

We first calculate this variance for Sarsa:

$$\begin{aligned} Var(s,a) &= \sum_{s'} T_{sa}^{s'} \Big(\gamma^2 \sum_{a'} \pi_{s'a'} (Q_t(s',a'))^2 + (R_{sa}^{s'})^2 \\ &\quad + 2\gamma R_{sa}^{s'} \sum_{a'} \pi_{s'a'} Q_t(s',a') \Big) - (E\{\hat{v}_t\})^2 \ . \end{aligned}$$

Similarly, for Expected-Sarsa we get:

$$\begin{aligned} Var(s,a) &= \sum_{s'} T_{sa}^{s'} \Big(\gamma^2 (\sum_{a'} \pi_{s'a'} Q_t(s',a'))^2 + (R_{sa}^{s'})^2 \\ &\quad + 2\gamma R_{sa}^{s'} \sum_{a'} \pi_{s'a'} Q_t(s',a') \Big) - (E\{\hat{v}_t\})^2 \ . \end{aligned}$$

Since $E\{v_t\} = E\{\hat{v}_t\}$, the difference between the two variances simplifies to the following:

$$\gamma^2 \sum_{s'} T_{sa}^{s'} \Big(\sum_{a'} \pi_{s'a'} (Q_t(s',a'))^2 - (\sum_{a'} \pi_{s'a'} Q_t(s',a'))^2 \Big) \ .$$

The inner term is of the form:

$$\sum_i w_i x_i^2 - (\sum_i w_i x_i)^2 \ , \tag{11}$$

where the w and x correspond to the π and Q values. When $w_i \geq 0$ for all i and $\sum_i w_i = 1$, we can give an unbiased estimate of the variance of the weighed values $w_i x_i$ as follows:

$$\frac{\sum_i w_i (x_i - \bar{x})^2}{1 - \sum_i w_i^2} \ , \tag{12}$$

where $\bar{x}$ is the weighted mean $\sum_i w_i x_i$. Taking the numerator of this fraction and rewriting this gives us:

$$\begin{aligned} \sum_i w_i (x_i - \bar{x})^2 &= \sum_i w_i x_i^2 - 2 \sum_i w_i x_i \bar{x} + \sum_i w_i \bar{x}^2 \\ &= \sum_i w_i x_i^2 - 2\bar{x}^2 + \bar{x}^2 \\ &= \sum_i w_i x_i^2 - \bar{x}^2 \ , \end{aligned}$$

which is exactly the same quantity as given in (11). This shows that this quantity is closely related to the weighted variance of the $w_i x_i$. Therefore, the more the x_i deviate from the weighted mean $\sum_i w_i x_i$, the larger this quantity will be. In our context this occurs in settings where there is a large difference between the Q values of different actions and there is much exploration. In case of a greedy policy or when all Q values have the same value, this quantity is 0.

VI. Hypotheses

In this section, we formulate specific hypotheses about when Expected Sarsa will outperform Q-learning and Sarsa. These hypotheses are based on the central differences between Expected Sarsa and these two alternatives: 1) unlike Q-learning, Expected Sarsa is on-policy and 2) Expected Sarsa has lower variance than Sarsa.

For simplicity, we restrict our attention to the case where exploration is performed using ϵ-soft behavior policies, i.e., the agent takes a random action with probability ϵ and uses the estimation policy otherwise. Using such exploration, off-policy methods can sometimes perform quite differently than on-policy methods. For example, in the cliff-walking task (detailed in Section VII), some actions can have disastrous consequences in certain states, e.g., when near a cliff. Off-policy methods try to estimate the optimal way to behave *without exploration* and then merely employ an ϵ-soft version of the resulting policy. Consequently, they may never learn to avoid such catastrophic actions. By contrast, on-policy methods try to estimate the optimal way to behave *given the exploration that is occurring*. Therefore, they can learn policies that are qualitatively different from the optimal policy without exploration but that avoid catastrophic actions in the presence of exploration, e.g., by staying further away from the cliff. Based on this difference we can define two different types of problems:

1) Problems where the optimal ϵ-soft policy is better than the ϵ-soft policy based on $Q^*(s,a)$.
2) Problems where the optimal ϵ-soft policy is equal to the ϵ-soft policy based on $Q^*(s,a)$.

Because Expected Sarsa is on-policy and Q-learning is off-policy, we we state the following hypothesis:

Hypothesis 1: Expected Sarsa will outperform Q-learning for problems of Type 1.

Section V demonstrated that the variance in the update target for Sarsa is larger than for Expected Sarsa, especially when the policy stochasticity is large and when there is a large spread in Q values of the actions of a state. Based on these facts, we can formulate a second hypothesis, one about the performance difference between Expected Sarsa and Sarsa.

Hypothesis 2: Expected Sarsa will outperform Sarsa on problems of both Type 1 and Type 2. The size of the performance difference depends primarily on two factors:

1) When environment stochasticity is high, performance difference will be small.
2) When policy stochasticity is high, performance difference will be large.

VII. Results and Discussion

In this section we present a series of experiments to compare the online performance of Expected Sarsa to that of Sarsa and Q-learning in order to test the hypotheses described in the previous section. We start with the cliff walking problem. This is an example of a problem where an exploration policy based on the optimal action values $Q^*(s,a)$ is not equal to the optimal policy with exploration added. Sutton and Barto showed that Sarsa outperforms Q-learning on this problem [2]. We show that Expected Sarsa outperforms Q-learning as well as Sarsa, confirming Hypothesis 1 and providing some evidence for Hypothesis 2.

We then present results on two versions of the windy grid world problem, one with a deterministic environment and one with a stochastic environment. We do so in order to evaluate the influence of environment stochasticity on the performance difference between Expected Sarsa and Sarsa and confirm the first part of Hypothesis 2. We then present results for different amounts of policy stochasticity to confirm the second part of Hypothesis 2. For completeness, we also show the performance of Q-learning on this problem. Finally, we present results in other domains verifying the advantages of Expected Sarsa in a broader setting. All results presented below are averaged over numerous independent trials such that the standard error becomes negligible.

A. *Cliff Walking*

We begin by testing Hypothesis 1 using the cliff walking task, an undiscounted, episodic navigation task in which the agent has to find its way from start to goal in a deterministic grid world. Along the edge of the grid world is a cliff (see Figure 1). The agent can take any of four movement actions: *up*, *down*, *left* and *right*, each of which moves the agent one square in the corresponding direction. Each step results in a reward of -1, except when the agent steps into the cliff area, which results in a reward of -100 and an immediate return to the start state. The episode ends upon reaching the goal state.

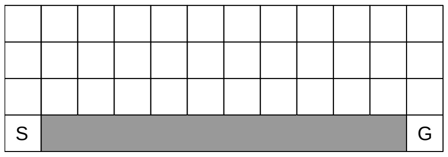

Fig. 1. The cliff walking task. The agent has to move from the start [S] to the goal [G], while avoiding stepping into the cliff (grey area).

We evaluated the performance over the first n episodes as a function of the learning rate α using an ϵ-greedy policy with ϵ = 0.1. Figure 2 shows the result for $n = 100$ and $n = 100,000$. We averaged the results over 50,000 runs and 10 runs, respectively.

Discussion. Expected Sarsa outperforms Q-learning and Sarsa for all learning rate values, confirming Hypothesis 1 and providing some evidence for Hypothesis 2. The optimal α value of Expected Sarsa for $n = 100$ is 1, while for Sarsa it is lower, as expected for a deterministic problem. That the optimal value of Q-learning is also lower than 1 is surprising, since Q-learning also has no stochasticity in its updates in a deterministic environment. Our explanation is that Q-learning first learns policies that are sub-optimal in the greedy sense, i.e. walking towards the goal with a detour further from the cliff. Q-learning iteratively optimizes these early policies, resulting in a path more closely along the cliff. However, although this path is better in the off-line sense, in terms of on-line performance it is worse. A large value of α ensures the goal is reached quickly, but a value somewhat lower than 1 ensures that the agent does not try to walk right on the edge of the cliff immediately, resulting in a slightly better on-line performance.

For $n = 100,000$, the average return is equal for all α values in case of Expected Sarsa and Q-learning. This indicates that the algorithms have converged long before the end of the run for all α values, since we do not see any effect of the initial learning phase. For Sarsa the performance comes close to the performance of Expected Sarsa only for $\alpha = 0.1$, while for large α, the performance for $n = 100,000$ even drops below the performance for $n = 100$. The reason is that for large values of α the Q values of Sarsa diverge. Although the policy is still improved over the initial random policy during the early stages of learning, divergence causes the policy to get worse in the long run.

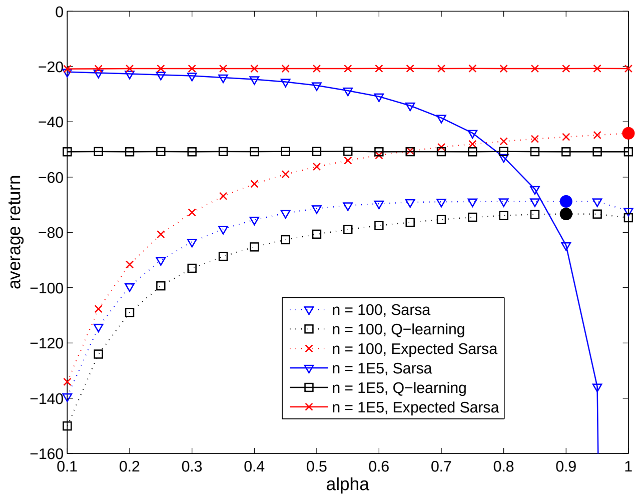

Fig. 2. Average return on the cliff walking task over the first n episodes for $n = 100$ and $n = 100,000$ using an ϵ-greedy policy with ϵ = 0.1. The big dots indicate the maximal values.

B. *Windy Grid World*

We turn to the windy grid world task to further test Hypothesis 2. The windy grid world task is another navigation task, where the agent has to find its way from start to goal. The grid has a height of 7 and a width of 10 squares. There is a wind blowing in the 'up' direction in the middle part of the grid, with a strength of 1 or 2 depending on the column. Figure 3 shows the grid world with a number below each column indicating the wind strength. Again, the agent can choose between four movement actions: *up*, *down*, *left* and *right*, each resulting in a reward of -1. The result of an action is a movement of 1 square in the corresponding direction plus an additional movement in the 'up' direction, corresponding with the wind strength. For example, when the agent is in the square right of the goal and takes a 'left' action, it ends up in the square just above the goal.

1) Deterministic Environment: We first consider a deterministic environment. As in the cliff walking task, we use an ϵ-greedy policy with $\epsilon = 0.1$. Figure 4 shows the performance as a function of the learning rate α over the first n episodes for $n = 100$ and $n = 100,000$. For $n = 100$

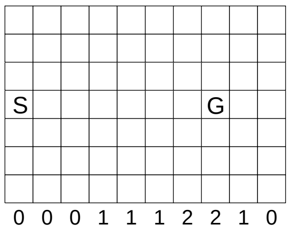

Fig. 3. The windy grid world task. The agent has to move from start [S] to goal [G]. The numbers under the grid indicate the wind strength in the column above.

the results are averaged over $10,000$ independent runs, for $n = 100,000$ over 10 independent runs.

Discussion. For the deterministic windy grid world task the performance of Q-learning and Expected Sarsa is essentially equal. The fact that for $n = 100,000$ the average return is equal indicates that the behavior policies of Expected Sarsa and Q-learning are equal in the limit for this task, i.e., the optimal policy among the ϵ-greedy policies (Expected Sarsa) is equal to the policy that is ϵ-greedy with respect to $Q^*(s,a)$ (Q-learning). The optimal α is 1 for Expected Sarsa as well as Q-learning. Sarsa again has a lower optimal α. As in the cliff walking task we observed divergence of Q values for high α values in the case of Sarsa. The performance difference for $n = 100$ between Expected Sarsa and Sarsa at their optimal values is $(-45.0) - (-58.3) = 13.3$ in favor of Expected Sarsa.

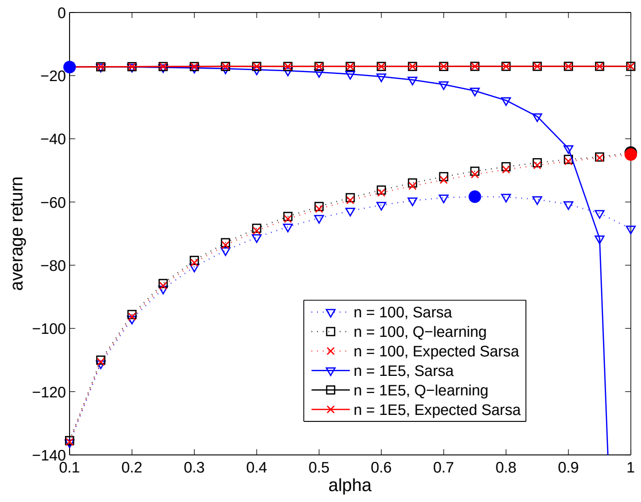

Fig. 4. Average return on the windy grid world task over the first n episodes for $n = 100$ and $n = 100,000$ and an ϵ-greedy policy with $\epsilon = 0.1$ in a deterministic environment. The big dots indicate maximal values.

2) Environment Stochasticity: We also consider a stochastic variation of the windy grid world problem and compare results to the performance difference in the deterministic case in order to evaluate the first part of Hypothesis 2. We added stochasticity to the environment by moving the agent with a probability of 20% in a random direction instead of the direction corresponding to the action. The performance as function of the learning rate is presented in Figure 5 for $n = 100$ and $n = 100,000$. Again, we averaged the results over 10,000 runs and 10 runs respectively.

Discussion. As expected, the optimal α for Expected Sarsa and Q-learning in case of $n = 100$ drops considerably in comparison to the deterministic case, to a value of 0.6. The optimal α value of Sarsa also decreases, to 0.55. From the $n = 100,000$ case, we can see that the policy no longer converges for Expected Sarsa and Q-learning for all α values. Although not stable for high α values, the average policy is better for Expected Sarsa than for Q-learning, which is likely due to the on-policy nature of Expected Sarsa. On the other hand, For $n = 100$, Q-learning slightly outperforms Expected Sarsa because it benefits more from *optimistic initialization*, i.e., initially overestimating the Q values to increase exploration during early learning. Since Q-learning uses the maximal Q value of the next state in its update, it takes longer for the Q values to decrease.

The performance difference for $n = 100$ between Expected Sarsa and Sarsa at their optimal values is $(-93.7) - (-98.3) = 4.6$ in favor of Expected Sarsa. The performance difference is less than half that of the deterministic case, confirming the first part of Hypothesis 2.

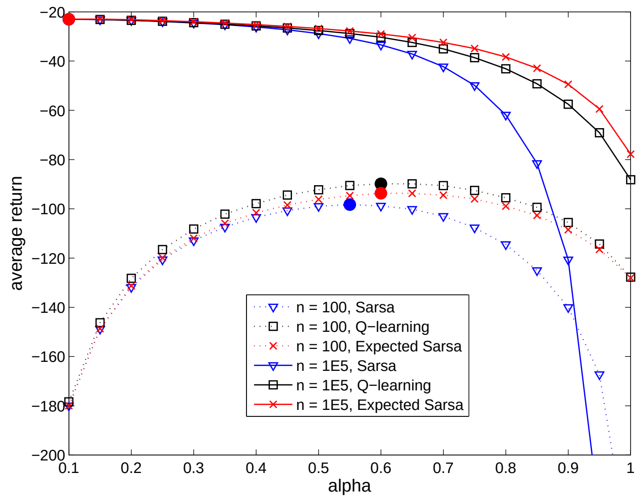

Fig. 5. Average return on the windy grid world task over the first n episodes for $n = 100$ and $n = 100,000$ using a ϵ-greedy policy with $\epsilon = 0.1$ in a stochastic environment. The big dots indicate maximal values.

3) Policy Stochasticity: To confirm the second part of Hypothesis 2, we repeat the stochastic windy grid world experiment but with higher policy stochasticity, using an ϵ of 0.3 instead of 0.1. Figure 6 shows the results.

Discussion For $n = 100$ the optimal α for Sarsa drops from 0.55 to 0.45 and the optimal α for Q-learning decreases slightly, though for Expected Sarsa it stays the same. Furthermore, the performance difference between Q-learning and Expected Sarsa increases. The performance difference between Sarsa and Expected Sarsa also increases for $n = 100$ and is now $(-121.0) - (-136.4) = 15.4$, confirming the second part of Hypothesis 2. Other experiments, not shown in this paper, confirmed that also the opposite is true: when policy stochasticity is low, i.e. using an ϵ-greedy policy with

$\epsilon = 0.01$ there is practically no performance difference between Sarsa and Expected Sarsa.

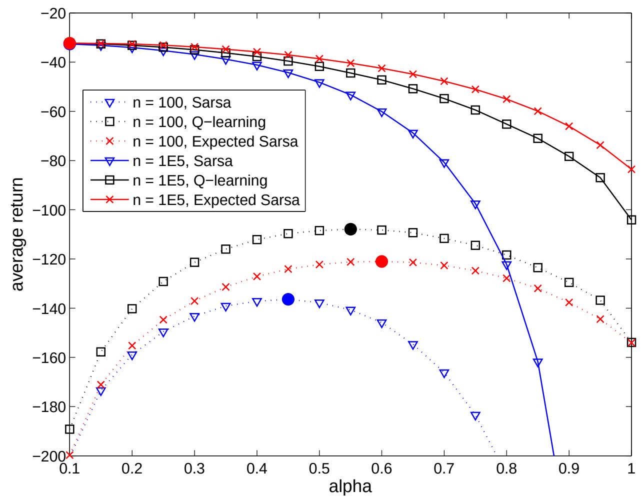

Fig. 6. Average return on the windy grid world task over the first n episodes for $n = 100$ and $n = 100,000$ using a ϵ-greedy policy with ϵ = 0.3 in a stochastic environment. The big dots indicate maximal values.

C. Other Domains

To demonstrate that the advantage of Expected Sarsa holds more generally, we also tested in other domains.

1) Maze: We compared Expected Sarsa to Sarsa and Q-learning on the maze problem shown in Figure 7. The goal of the agent is to find a path from start to goal, while avoiding hitting the walls. The reward for arriving at the goal is 100. When the agent bumps into a wall or border of the environment it stays at the same position, but receives a reward of -2. For all other steps a reward of -0.1 is received. The environment is stochastic and moves the agent with a probability of 10% in a random direction instead of the direction corresponding to the action. The discount factor γ is set to 0.997. A trial is finished after the agent reaches the goal or 10,000 actions have been performed. An ϵ-greedy behavior policy is used with $\epsilon = 0.05$ and we initialized the Q values to 0.

We optimized α for each method such that the average reward over the first $2 * 10^6$ timesteps is maximized. The optimal values were 0.24, 0.28 and 0.27 for Sarsa, Q-learning and Expected Sarsa respectively. We then plotted the reward as function of the number of timesteps for these optimal α values to get a more detailed look at performance. Figure 8 shows the results, which are averaged over 100 trials.

Discussion. Although Expected Sarsa and Q-learning perform equally, Sarsa's performance is lower and not monotonically increasing. It shows a drop in performance after $0.2 * 10^6$ timesteps, before it slowly increases again. This drop occurs in all one hundred runs.

Although this is a clear demonstration of the possibility that Sarsa can be unstable in certain cases, we have not observed this phenomenon in previous research, and it is remarkable because the value function is represented in a table, without the complications of function approximation. We explain this temporary performance drop of Sarsa as follows: since in our implementation we initialized all Q values to 0, while their real value is higher, all values start to increase in the beginning. However, the values of the best actions increase faster because they have a shorter propagation path to the final reward of 100. Therefore, initially Sarsa learns well. However, because of the high discount factor of 0.997, all action-values in a state start to get very close to each other. This makes it possible that after a bad exploration step, some values are updated in a way that makes the policy worse. After a while Sarsa finds a policy that is not optimal, but that is robust against such value updates. The same drop in performance also happens when using a learning rate of 0.04 for Sarsa, although initial learning performance was slower and the drop occurred later. The update targets of Expected Sarsa and Q-learning are not effected by the action selected in the next state and are therefore more robust towards performance drops.

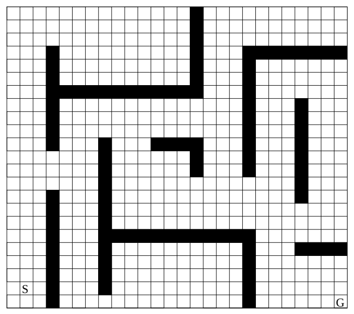

Fig. 7. The maze problem. The starting position is indicated by [S] and the goal position is indicated by [G].

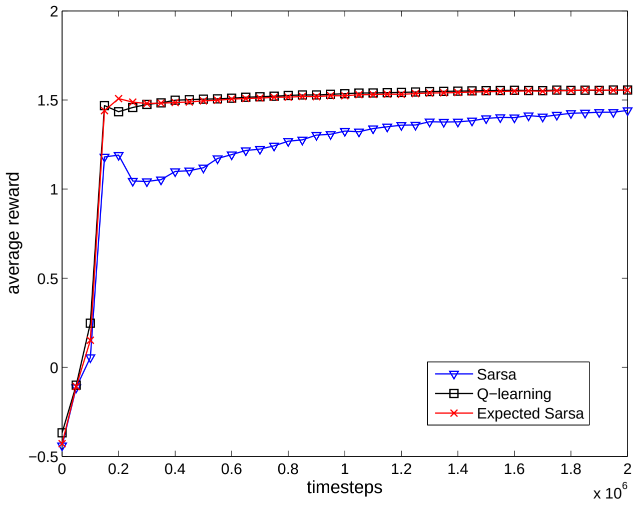

Fig. 8. The on-line performance of the different methods on the maze problem. The results are averaged over 100 runs.

2) Cart Pole: As a final comparison, we test the on-line performance of Expected Sarsa, Sarsa and Q-learning on a cart-pole task. The goal was to balance a 1 m long pole, weighing 0.1 kg, on a cart that weighs 1.0 kg. The possible actions were all integer amounts between -10 and 10 Newton, where positive and negative forces correspond to pushing the cart right and left, respectively. An action was performed every 0.02 s. If the cart was pushed further than 2.4 m from the center of the track or if the pole drops further than 12 degrees to either side, the algorithm would receive a -1 reward and the cart would be reset to the center with the pole at a random angle between -3 and 3 degrees. A neural network with 15 sigmoidal hidden units was used to approximate the Q values. The input vector consisted of the position and velocity of the cart and the angle and angular velocity of the pole, all normalized to [-1,1]. The value of ϵ was 0.05 and γ was 0.95. Figure 9 shows the average reward during learning at optimized α values of 0.12, 0.16 and 0.16 for Sarsa, Q-learning and Expected Sarsa respectively.

Discussion. We see again that Expected Sarsa and Q-learning perform similar, while Sarsa is less stable and shows lower performance. This demonstrates that the results extend to the case of function approximation.

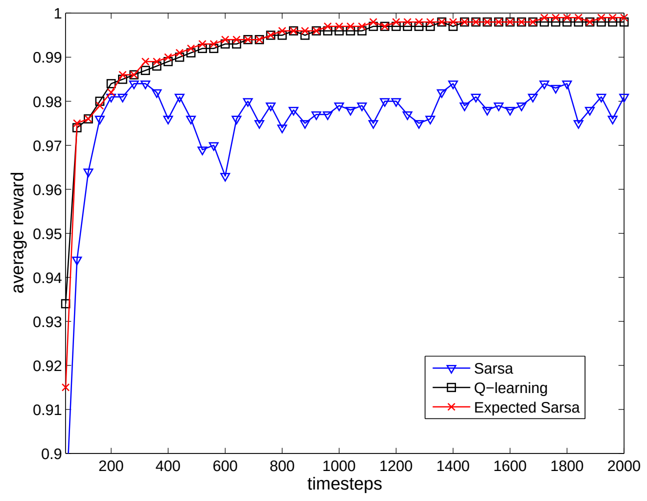

Fig. 9. The learning performance of the different methods on the cart pole. The results are averaged over 200 simulations.

VIII. Conclusion

In this paper we examined Expected Sarsa, a variation on the Sarsa algorithm intended to decrease the variance in the update rule, and compared it to the Sarsa and the Q-learning algorithm.

We proved that Expected Sarsa converges under the same conditions as Sarsa. We also proved that the variance in the update rule of Expected Sarsa is smaller than the variance for Sarsa and that the difference in variance is largest when there is a high amount of exploration and a large spread in Q values of the actions of a specific state. Based on this theoretical analysis, we hypothesized that the on-line performance of Expected Sarsa will be higher than for Sarsa and that the difference in performance will be relatively large when there is a lot of policy exploration and small when the environment is very stochastic. We also formulated a second hypothesis based on the on-policy nature of Expected Sarsa that states that Expected Sarsa will outperform Q-learning for problems where an ϵ-soft behavior policy based on $Q^*(s, a)$ is not equal to the optimal ϵ-soft policy. We confirmed these hypotheses using experiments on the cliff walking task and the windy grid world task. Finally, we presented results on two additional problems to verify the advantages of Expected Sarsa in a broader setting.

References

[1] L. P. Kaelbling, M. L. Littman, and A. P. Moore, "Reinforcement learning: A survey," *Journal of Artificial Intelligence Research*, vol. 4, pp. 237–285, 1996.

[2] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, Massachussets: MIT Press, 1998.

[3] R. E. Bellman, "A Markov decision process," *Journal of Mathematical Mechanics*, vol. 6, pp. 679–684, 1957.

[4] R. E. Bellman, *Dynamic Programming*. Princeton, NJ.: Princeton University Press, 1957.

[5] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Machine Learning*, vol. 3, pp. 9–44, 1988.

[6] C. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8(3-4), pp. 9–44, 1992.

[7] J. A. Boyan and A. W. Moore, "Generalization in reinforcement learning: Safely approximating the value function," in *Advances in Neural Information Processing Systems 7*, G. Tesauro, D. S. Touretzky, and T. K. Leen, Eds. MIT Press, Cambridge MA, 1995, pp. 369–376.

[8] G. Gordon, "Stable function approximation in dynamic programming," in *Machine Learning: Proceedings of the Twelfth International Conference*, A. Prieditis and S. Russell, Eds. Morgan Kaufmann Publishers, San Francisco, CA, 1995, pp. 261–268.

[9] L. Baird, "Residual algorithms: Reinforcement learning with function approximation," in *Machine Learning: Proceedings of the Twelfth International Conference*, A. Prieditis and S. Russell, Eds. Morgan Kaufmann Publishers, San Francisco, CA, 1995, pp. 30–37.

[10] G. Rummery and M. Niranjan, "On-line Q-learning using connectionist systems," Cambridge University, Tech. Rep. CUED/F-INFENG/TR 166, 1994.

[11] R. Sutton, "Generalization in reinforcement learning: Successful examples using sparse coarse coding," in *Advances in Neural Information Processing Systems 8*, 1996, pp. 1038–1044.

[12] S. Singh, T. Jaakkola, M. L. Littman, and C. Szepesvári, "Convergence results for single-step on-policy reinforcement-learning algorithms," *Machine Learning*, vol. 38, no. 3, pp. 287–308, 2000.