Learning and Planning in Average-Reward Markov Decision Processes

Yi Wan [*1] **Abhishek Naik** [*1] **Richard S. Sutton** [1 2]

Abstract

We introduce learning and planning algorithms for average-reward MDPs, including 1) the first general proven-convergent off-policy model-free control algorithm without reference states, 2) the first proven-convergent off-policy model-free prediction algorithm, and 3) the first off-policy learning algorithm that converges to the actual value function rather than to the value function plus an offset. All of our algorithms are based on using the temporal-difference error rather than the conventional error when updating the estimate of the average reward. Our proof techniques are a slight generalization of those by Abounadi, Bertsekas, and Borkar (2001). In experiments with an Access-Control Queuing Task, we show some of the difficulties that can arise when using methods that rely on reference states and argue that our new algorithms can be significantly easier to use.

1. Average-Reward Learning and Planning

The average-reward formulation of Markov decision processes (MDPs) is arguably the most important for reinforcement learning and artificial intelligence (see, e.g., Sutton & Barto 2018 Chapter 10, Naik et al. 2019) yet has received much less attention than the episodic and discounted formulations. In the average-reward setting, experience is continuing (not broken up into episodes) and the agent seeks to maximize the average reward per step, or *reward rate*, with equal weight given to immediate and delayed rewards. In addition to this *control* problem, there is also the *prediction* problem of estimating the value function and the reward rate for a given *target policy*. Solution methods for these problems can be divided into those that are driven by experiential data, called *learning* algorithms, those that are driven by a model of the MDP, called *planning* algorithms, and combined methods that first learn a model and then plan with it. For learning and combined methods, both control and prediction problems can be further subdivided into *on-policy* versions, in which data is gathered using the target policy, and *off-policy* versions, in which data is gathered using a second policy, called the *behavior policy*. In general, both policies may be non-stationary. For example, in the control problem, the target policy should converge to a policy that maximizes the reward rate. Useful surveys of average-reward learning are given by Mahadevan (1996) and Dewanto et al. (2020).

On-policy problems are generally easier than off-policy problems and permit more capable algorithms with convergence guarantees. For example, on-policy *prediction* algorithms with function approximation and convergence guarantees include average-cost TD(λ) (Tsitsiklis & Van Roy 1999), LSTD(λ) (Konda 2002), and LSPE(λ) (Yu & Bertsekas 2009). On-policy *control* algorithms that have been proved to converge asymptotically or to achieve sublinear regret or to be probably approximately correct under various conditions include tabular learning algorithms (e.g., Wheeler & Narendra 1986, Abbasi-Yadkori et al. 2019a,b), tabular combined algorithms (e.g., Kearns & Singh 2002, Brafman & Tennenholtz 2002, Auer & Ortner 2006, Jaksch et al. 2010), and policy gradient algorithms (e.g., Sutton et al. 1999, Marbach & Tsitsiklis 2001, Kakade 2001, Konda 2002).

The *off-policy learning control* problem is particularly challenging, and theoretical results are available only for the tabular, discrete-state setting without function approximation. The most important prior algorithm is RVI Q-learning, introduced by Abounadi, Bertsekas, and Borkar (1998, 2001). The same paper also introduced *SSP Q-learning*, but SSP Q-learning was limited to MDPs with a special state that is recurrent under all stationary policies, whereas RVI Q-learning is convergent for more general MDPs. Ren and Krogh (2001) presented a tabular algorithm and proved its convergence, but their algorithm required knowledge of properties of the MDP which are not in general known. Gosavi (2004) also introduced an algorithm and proved its convergence, but it was limited in the same way as SSP Q-learning. Yang et al. (2016) presented an algorithm and claimed to prove its convergence, but their proof is not correct (as we detail in Appendix D). The earliest tabular average-reward off-policy learning control algorithms that

[*]Equal contribution. [1]University of Alberta and Alberta Machine Intelligence Institute (Amii), Edmonton, Canada. [2]DeepMind. Correspondence to: Yi Wan <wan6@ualberta.ca>, Abhishek Naik <abhishek.naik@ualberta.ca>.

Proceedings of the 38th International Conference on Machine Learning, PMLR 139, 2021. Copyright 2021 by the author(s).

we know of were those introduced (without convergence proofs) by Schwartz (1993) and Singh (1994). Bertsekas and Tsitsiklis (1996) and Das et al. (1999) introduced off-policy learning control algorithms with function approximation, but did not provide convergence proofs.

Abounadi et al.'s RVI Q-learning is actually a family of off-policy algorithms, a particular member of which is determined by specifying a function that references the estimated values of specific state–action pairs and produces an estimate of the reward rate. We call this function the *reference function*. Examples include a weighted average of the value estimates of all state–action pairs, or in the simplest case, the estimate of a single state–action pair's value. For best results, the referenced state–action pairs should be frequently visited; otherwise convergence can be unduly slow (as we illustrate in Section 3). However, if the behavior policy is linked to the target policy (as in ϵ-greedy behavior policies), then knowing which state–action pairs will be frequently visited may be to know a substantial part of the problem's solution. For example, in learning an optimal path through a maze from diverse starting points, the frequently visited state–action pairs are likely to be those on the shortest paths to the goal state. To know these would be tantamount to knowing a priori the best paths to the goal. This observation motivates the search for a general learning algorithm that does not require a reference function.

Our first contribution is to introduce such a learning control algorithm without a reference function. Our *Differential Q-learning* algorithm is convergent for general MDPs, which we prove by slightly generalizing the theory of RVI Q-learning (Abounadi et al. 2001). Unlike RVI Q-learning, Differential Q-learning does not involve reference states. Instead, it maintains an explicit estimate of the reward rate (as in Schwartz 1993, Singh 1994).

Our second contribution is *Differential TD-learning*, the first off-policy model-free prediction learning algorithm proved convergent to the reward rate and differential value function of the target policy. There are a number of algorithms that estimate the reward rate (e.g., Wen et al. 2020, Liu et al. 2018, Tang et al. 2019, Mousavi et al. 2020, Zhang et al. 2020a,b), but none that estimate the value function. These algorithms also differ from Differential TD-learning in that are not online algorithms; they operate on a fixed batch of data. Finally, they differ in that they estimate the ratio of the steady-state occupancy distributions under the target and behavior policies, whereas Differential TD-learning does not.

Planning algorithms for average-reward MDPs have been known at least since the setting was introduced by Howard in 1960. However, most of these, including value iteration (Bellman 1957), policy iteration (Howard 1960), and relative value iteration (RVI, White 1963), are ill-suited for use in reinforcement learning because they involve sub-steps whose complexity is order the number of states or more. Jalali and Ferguson (1989, 1990) were among the first to explore more incremental methods, though their algorithms are limited to special-case MDPs and require a reference state–action pair. In planning, as in learning, the state of the art appears to be RVI Q-learning, now applied as a planning algorithm to a stream of experience generated by the model. When our Differential Q-learning algorithm is applied in the same way, we call it *Differential Q-planning*; it improves over the RVI Q-learning's planner in that it omits reference states, with concomitant efficiencies just as in the learning case. In the prediction case we have *Differential TD-planning*. Both of these algorithms are fully incremental and well suited for use in reinforcement learning architectures (e.g., Dyna (Sutton 1990)).

All the aforementioned average-reward algorithms converge not to the actual value function, but to the value function plus an offset that depends on initial conditions or on a reference state or state–action pair. The offset is not necessarily a problem because only the relative values of states (or of state–action pairs) are used to determine policies. However, the actual value function of any policy is *centered*, meaning that the mean value of states encountered under the policy is zero. Although it is easy to center an estimated value function in the on-policy case, in the off-policy case it is not. Our final contribution is to extend our off-policy algorithms to centered versions that converge to the actual value function without an offset.

2. Learning and Planning for Control

We formalize an agent's interaction with its environment by a finite Markov decision process, defined by the tuple $\mathcal{M} \doteq (\mathcal{S}, \mathcal{A}, \mathcal{R}, p)$, where $\mathcal{S}$ is a set of states, $\mathcal{A}$ is a set of actions, $\mathcal{R}$ is a set of rewards, and $p : \mathcal{S}\times\mathcal{R}\times\mathcal{S}\times\mathcal{A} \to [0, 1]$ is the dynamics of the environment. At each of a sequence of discrete time steps $t = 0, 1, 2, \ldots$, the agent receives an indication of a state of the MDP $S_t \in \mathcal{S}$ and selects, using behavior policy $b : \mathcal{A} \times \mathcal{S} \to [0, 1]$, an action $A_t \in \mathcal{A}$, then receives from the environment a reward $R_{t+1} \in \mathcal{R}$ and the next state $S_{t+1} \in \mathcal{S}$, and so on. The transition dynamics are such that $p(s', r \mid s, a) \doteq \Pr(S_{t+1} = s', R_{t+1} = r \mid S_t = s, A_t = a)$ for all $s, s' \in \mathcal{S}, a \in \mathcal{A}$, and $r \in \mathcal{R}$. All policies we consider in the paper are in the set of stationary Markov policies Π.

Technically, for an unconstrained MDP, the best reward rate depends on the start state. For example, the MDP may have two disjoint sets of states with no policy that passes from one to the other; in this case there are effectively two MDPs, with unrelated rates of reward. A learning algorithm would have no difficulty with such cases—it would optimize for whichever sub-MDP it found itself in—but it is complex to state formally what is meant by an optimal policy. To remove this complexity, it is commonplace to rule out such cases by assuming that the MDP is communicating, which just means that there are no states from which it is impossible to get back to the others.

Communicating Assumption: For every pair of states, there exists a policy that transitions from one to the other in a finite number of steps with non-zero probability.

Under the communicating assumption, there exists a unique optimal reward rate r^* that does not depend on the start state. To define r^*, we will need the reward rate for an arbitrary policy π and a given start state s:

$$r(\pi, s) \doteq \lim_{n\to\infty} \frac{1}{n} \sum_{t=1}^{n} \mathbb{E}[R_t \mid S_0 = s, A_{0:t-1} \sim \pi]. \quad (1)$$

It turns out that the best reward rate from s, $\sup_\pi r(\pi, s)$, does not depend on s (see, e.g., Puterman 1994), and we define it as r^*. We seek a learning algorithm which achieves r^*.

Our *Differential Q-learning* algorithm updates a table of estimates $Q_t : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ as follows:

$$\begin{aligned} Q_{t+1}(S_t, A_t) &\doteq Q_t(S_t, A_t) + \alpha_t \delta_t, \quad (2)\\ Q_{t+1}(s, a) &\doteq Q_t(s, a),\ \forall s, a \neq S_t, A_t, \end{aligned}$$

where α_t is a step-size sequence, and δ_t, the temporal-difference (TD) error, is:

$$\delta_t \doteq R_{t+1} - \bar{R}_t + \max_a Q_t(S_{t+1}, a) - Q_t(S_t, A_t), \quad (3)$$

where $\bar{R}_t$ is a scalar estimate of r^*, updated by:

$$\bar{R}_{t+1} \doteq \bar{R}_t + \eta \alpha_t \delta_t, \quad (4)$$

and η is a positive constant.

The following theorem shows that $\bar{R}_t$ converges to r^* and Q_t converges to a solution of q in the Bellman equation:

$$q(s, a) = \sum_{s', r} p(s', r \mid s, a)(r - \bar{r} + \max_{a'} q(s', a')), \quad (5)$$

for all $s \in \mathcal{S}$ and $a \in \mathcal{A}$. The unique solution for $\bar{r}$ is r^*. To guarantee that Q_t converges to a unique point, we need to assume that the solution of q is unique up to a constant.

Theorem 1 (Informal). *If 1) the MDP is communicating, 2) the solution of q in* (5) *is unique up to a constant, 3) the step sizes, specific to each state–action pair, are decreased appropriately, 4) all the state–action pairs are updated an infinite number of times, and 5) the ratio of the update frequency of the most-updated state–action pair to the update frequency of the least-updated state–action pair is finite, then the Differential Q-learning algorithm* (2)–(4) *converges, almost surely, $\bar{R}_t$ to r^*, Q_t to a solution of q in* (5)*, and $r(\pi_t, s)$ to r^*, for all $s \in \mathcal{S}$, where π_t is any greedy policy w.r.t. Q_t.*

Proof. (Sketch; complete proof in Appendix B) Our proof comprises two steps. First, we combine our algorithm's two updates to obtain a single update that is similar to the RVI Q-learning's update. Second, we extend the family of RVI-learning algorithms so that the aforementioned single update is a member of the extended family and show convergence for the extended family.

Define $\Sigma_t \doteq \sum_{s,a} Q_t(s, a)$. At each time step, the increment to $\bar{R}_t$ is η times the increment to Q_t and hence to Σ_t. Therefore, the cumulative increment can be written as:

$$\begin{aligned} \bar{R}_t - \bar{R}_0 &= \sum_{i=0}^{t-1} \eta \alpha_i \delta_i = \eta\left(\Sigma_t - \Sigma_0\right) \\ \implies \bar{R}_t &= \eta\Sigma_t - c, \text{ where } c \doteq \eta\Sigma_0 - \bar{R}_0. \quad (6) \end{aligned}$$

Next, substitute $\bar{R}_t$ in (2) with (6):

$$\begin{aligned} Q_{t+1}(S_t, A_t) &= Q_t(S_t, A_t) + \\ \alpha_t\big(R_{t+1} &+ \max_a Q_t(S_{t+1}, a) - Q_t(S_t, A_t) - \eta\Sigma_t + c\big) \\ &= Q_t(S_t, A_t) + \\ \alpha_t\big(\tilde{R}_{t+1} &+ \max_a Q_t(S_{t+1}, a) - Q_t(S_t, A_t) - \eta\Sigma_t\big), \quad (7) \end{aligned}$$

where $\tilde{R}_{t+1} \doteq R_{t+1} + c$. Now (7) is in the same form as RVI Q-learning's update:

$$\begin{aligned} Q_{t+1}(S_t, A_t) &= Q_t(S_t, A_t) + \\ \alpha_t\big(R_{t+1} &+ \max_a Q_t(S_{t+1}, a) - Q_t(S_t, A_t) - f(Q_t)\big), \end{aligned} \quad (8)$$

with $f(Q_t) = \eta\Sigma_t$ for a slightly different MDP $\tilde{\mathcal{M}}$ whose rewards are all shifted by c.

Note that the convergence of Q_t in (7) cannot be obtained using the convergence theorem of RVI Q-learning because $\eta\Sigma_t = \eta\sum_{s,a} Q_t(s, a)$ in general does not satisfy conditions on f allowed by Assumption 2.2 of Abounadi et al. (2001). However, by extending the family of RVI Q-learning algorithms to cover the case of $f(Q_t) = \eta\sum_{s,a} Q_t(s, a)\ \forall\eta \in \mathbb{R}$, we show that the convergence of Q_t in (7) holds. In particular, we show that Q_t converges almost surely to a solution, denoted as q_∞, which is the unique solution for q in (5) under MDP $\tilde{\mathcal{M}}$ and $\eta\sum_{s,a} q(s, a) = r_* + c$. It can be shown that q_∞ is also a solution for q in (5) in $\mathcal{M}$. Additionally, because $\eta\Sigma_t = \eta\sum_{s,a} Q_t(s, a)$ converges to $\eta\sum_{s,a} q_\infty(s, a) = r_* + c$, we have $\bar{R}_t = \eta\Sigma_t - c$ converges to r_* almost surely. The almost-sure convergence of $r(\pi_t, s)$ to r_*, $\forall s$ then follows from a variant of Theorem 8.5.5 by Puterman (1994), the continuous mapping theorem, and the convergence of Q_t. □

Remark: Interestingly, RVI Q-learning and Differential Q-learning make the same updates to Q_t in special cases.

For RVI Q-learning, the special case is when the reference function is the mean of all state–action pairs' values. For Differential Q-learning, the special case is when $\eta = \frac{1}{|\mathcal{S}||\mathcal{A}|}$. These special cases are not particularly good for either algorithm, and therefore their special-case equivalence tells us little about the relationship between the algorithms in practice. In RVI Q-learning, it is generally better for the reference function to emphasize state–action pairs that are frequently visited rather than to weight all state–action pairs equally (an example of this is shown and discussed in Section 3). In Differential Q-learning, the special-case setting of $\eta = \frac{1}{|\mathcal{S}||\mathcal{A}|}$ would often be much too small on problems with large state and action spaces.

If Differential Q-learning is applied to simulated experience generated from a model of the environment, then it becomes a planning algorithm, which we call *Differential Q-planning*. Formally, the model is a function $\hat{p} : \mathcal{S} \times \mathcal{R} \times \mathcal{S} \times \mathcal{A} \to [0, 1]$, analogous to p, that, like p, sums to 1: $\sum_{s',r} \hat{p}(s', r \mid s, a) = 1$ for all s, a. A model MDP can be thus constructed using $\hat{p}$ and $\mathcal{S}, \mathcal{A}, \mathcal{R}$. If the model MDP is communicating, then there is a unique optimal reward rate $\hat{r}_*$. The simulated transitions are generated as follows: at each planning step n, the agent arbitrarily chooses a state S_n and an action A_n, and applies $\hat{p}$ to generate a simulated resulting state and reward $S'_n, R_n \sim \hat{p}(\cdot, \cdot \mid S_n, A_n)$.

Like Differential Q-learning, Differential Q-planning maintains a table of action-value estimates $Q_n : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ and a reward-rate estimate $\bar{R}_n$. At each planning step n, these estimates are updated by (2)–(4), just as in Differential Q-learning, except now using S_n, A_n, R_n, S'_n instead of $S_t, A_t, R_{t+1}, S_{t+1}$.

Theorem 2 (Informal)**.** *Under the same assumptions made in Theorem 1 (except now for the model MDP corresponding to $\hat{p}$ rather than p) the Differential Q-planning algorithm converges, almost surely, $\bar{R}_n$ to $\hat{r}_*$ and Q_n to a solution of q in the Bellman equation (cf.* (5)*) for the model MDP.*

Proof. Essentially as in Theorem 1. Full proof in Appendix B. □

3. Empirical Results for Control

In this section we present empirical results with both Differential Q-learning and RVI Q-learning algorithms on the Access-Control Queuing task (Sutton & Barto 2018). This task involves customers queuing up to access to one of 10 servers. The customers have differing priorities (1, 2, 4, or 8), which are also the rewards received if and when their service is complete. At each step, the customer at the head of queue is either accepted and allocated a free server (if any) or is rejected (in which case a reward of 0 is received). This decision is made based on the priority of the customer and the number of currently free servers, which together constitute the state of this average-reward MDP. The rest of the details of this test problem are exactly as described by Sutton and Barto (2018).

We applied RVI Q-learning and Differential Q-learning (pseudocodes for both algorithms are in Appendix A) to this task, each for 30 runs of 80,000 steps, and each for a range of step sizes α. Differential Q-learning was run with a range of η values, and RVI Q-learning was run with three kinds of reference functions suggested by Abounadi et al. (2001): (1) the value of a single reference state–action pair, for which we considered all possible 88 state–action pairs, (2) the maximum value of the action-value estimates, and (3) the mean of the action-value estimates. Both algorithms used an ϵ-greedy behavior policy with $\epsilon = 0.1$. The rest of the experimental details are in Appendix C.

A typical learning curve is shown in Figure 1. While this learning curve is for Differential Q-learning, the learning curves for both algorithms typically started at around 2.2 and plateaued at around 2.6, with different parameter settings leading to different rates of learning. A reward rate of 2.2 corresponds to a policy that accepts every customer irrespective of their priority or the number of free servers—with positive rewards for every accept action, such a policy is learned rapidly in the first few timesteps starting from a zero initialization of value estimates (i.e., a random policy). The optimal performance was close to 2.7 (note both algorithms use an ϵ-greedy policy without annealing ϵ).

Figure 2 shows parameter studies for each algorithm. Plotted is the reward rate averaged over all 80,000 steps, reflecting their rates of learning. The error bars denote one standard error.

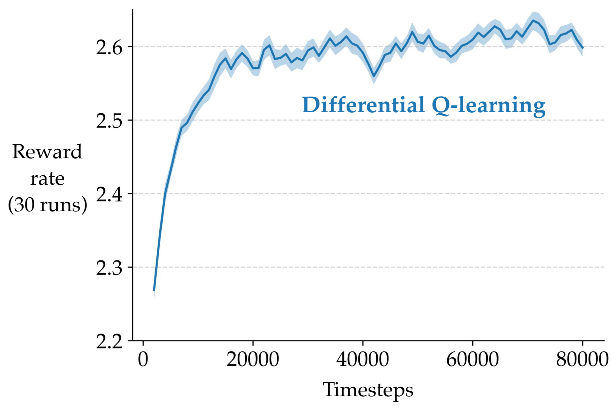

Figure 1: A typical learning curve for the Access-Control Queuing task. A point on the solid line denotes reward rate over the last 2000 timesteps, and the shaded region indicates one standard error.

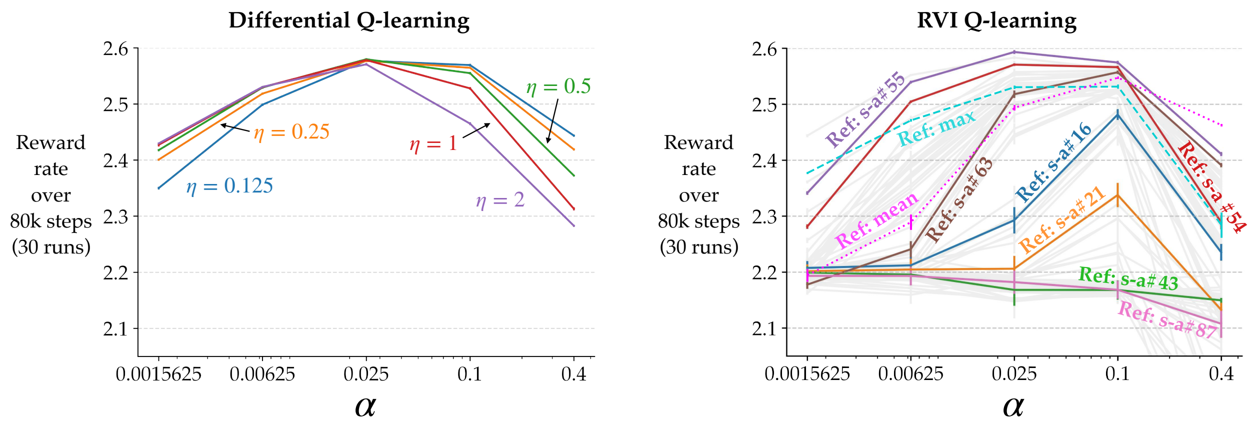

Figure 2: Parameter studies showing the sensitivity of the two algorithms' performance to their parameters. The error bars indicate one standard error, which at times is less than the width of the solid lines. *Left:* Differential Q-learning's rate of learning varied little over a broad range of its parameters. *Right:* RVI Q-learning's rate of learning depended strongly on the choice of the reference function. The solid greyed-out lines mark the performance for each of the 88 state–action pairs considered individually as the single reference pair, with a few representative ones highlighted (labelled as 'Ref: s-a'). The dotted lines correspond to the reference function being the mean or the max of all the action-value estimates.

We saw that Differential Q-learning performed well on this task for a wide range of parameter values (left panel). Its two parameters did not interact strongly; the best value of α was independent of the choice of η. Moreover, the best performance for different η values was roughly the same.

RVI Q-learning also performed well on this task for the best choice of the reference state–action pair, but its performance varied significantly for the various choices of the reference function and state–action pairs (right panel).

A closer look at the data revealed a correlation between the performance of a particular reference state–action pair and how frequently it occurs under an optimal policy. For example, state–action pairs 55 and 54 occurred frequently and also resulted in good performance. They correspond to states when only two servers are free and the customer at the front of the queue has priority 8 and 4 respectively, and the action is to accept. This is the optimal action in this state. On the other hand, the performance was poor with state–action pairs 43 and 87, which occurred rarely. They correspond to states when all 10 servers are free, a condition that rarely occurs in this problem. Finally, the mean of value estimates of all state–action pairs performs moderately well as a reference function. These observations lead us to a conjecture: *an important factor determining the performance of RVI Q-learning with a single reference state–action pair is how often that pair occurs under an optimal policy*. This is problematic because knowing which state–action pairs will occur frequently under an optimal policy is tantamount to knowing the solution of the problem we set out to solve.

The conjecture might lead us to think that the reference function that is the max over all action-value estimates would always lead to good performance because the corresponding state–action pair would occur most frequently under an optimal policy, but this is not true in general. For example, consider an MDP with a state that rarely occurs under any policy. Let all rewards in the MDP be zero except a positive reward from that state. Then the highest action value among all state–action pairs is corresponding to this rarely-occurring state.

To conclude, our experiments with the Access-Control Queuing task show that the performance of RVI Q-learning can vary significantly over the range of reference functions and state–action pairs. On the other hand, Differential Q-learning does not use a reference function and can be significantly easier to use.

4. Learning and Planning for Prediction

In this section, we define the problem setting for the prediction problem and then present our new algorithms for learning and planning.

In the prediction problem, we deal with Markov chains induced by the target and the behavior policies when applied to an MDP. The MDP interactions are the same as described earlier (Section 2).

As before, it is convenient to rule out the possibility of the reward rate of the target policy depending on the start state. In particular, we assume that under the target policy there is only one possible limiting distribution for the resulting

Markov chain, independent of the start state. This is known as the Markov chain being *unichain*. The reward rate of the target policy then does not depend on the start state. We denote it as $r(\pi)$, where π is the target policy:

$$r(\pi) \doteq \lim_{n\to\infty} \frac{1}{n} \sum_{t=1}^{n} \mathbb{E}[R_t \mid A_{0:t-1} \sim \pi]. \tag{9}$$

The *differential state-value function* (also called bias; see, e.g., Puterman 1994) $v_\pi : \mathcal{S} \to \mathbb{R}$ for a policy π is:

$$v_\pi(s) \doteq \lim_{n\to\infty} \frac{1}{n} \sum_{k=1}^{n} \sum_{t=1}^{k} \mathbb{E}\left[R_t - r(\pi) \mid S_0 = s, A_{0:t-1} \sim \pi\right],$$

for all $s \in \mathcal{S}$. As usual, the differential state-value function satisfies a recursive Bellman equation:

$$v(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r \mid s, a)\big(r - \bar{r} + v(s')\big), \tag{10}$$

for all $s \in \mathcal{S}$. The unique solution for $\bar{r}$ is $r(\pi)$ and the solutions for $v : \mathcal{S} \to \mathbb{R}$ are unique up to an additive constant.

As usual in off-policy prediction learning, we need an assumption of *coverage*. In this case we assume that every state–action pair for which $\pi(a|s) > 0$ occurs an infinite number of times under the behavior policy.

Our *Differential TD-learning* algorithm updates a table of estimates $V_t : \mathcal{S} \to \mathbb{R}$ as follows:

$$V_{t+1}(S_t) \doteq V_t(S_t) + \alpha_t \rho_t \delta_t, \tag{11}$$
$$V_{t+1}(s) \doteq V_t(s),\ \forall s \neq S_t,$$

where α_t is a step-size sequence, $\rho_t \doteq \pi(A_t|S_t)/b(A_t|S_t)$ is the importance-sampling ratio, and δ_t is the TD error:

$$\delta_t \doteq R_{t+1} - \bar{R}_t + V_t(S_{t+1}) - V_t(S_t), \tag{12}$$

where $\bar{R}_t$ is a scalar estimate of $r(\pi)$, updated by:

$$\bar{R}_{t+1} \doteq \bar{R}_t + \eta \alpha_t \rho_t \delta_t, \tag{13}$$

and η is a positive constant.

The following theorem shows that $\bar{R}_t$ converges to $r(\pi)$ and V_t converges to a solution of v in (10).

Theorem 3 (Informal). *If 1) the Markov chain induced by the target policy π is unichain, 2) every state–action pair for which $\pi(a|s) > 0$ occurs an infinite number of times under the behavior policy, 3) the step sizes, specific to each state, are decreased appropriately, and 4) the ratio of the update frequency of the most-updated state to the update frequency of the least-updated state is finite, then the Differential TD-learning algorithm* (11)–(13) *converges, almost surely, $\bar{R}_t$ to $r(\pi)$ and V_t to a solution of v in the Bellman equation* (10).

Proof. Essentially as in Theorem 1. Full proof in Appendix B. □

Note that this result applies to both on-policy and off-policy problems. In off-policy problems, Differential TD-learning is the first model-free average-reward algorithm proved to converge to the true reward rate.

The planning version of Differential TD-learning, called *Differential TD-planning*, uses simulated transitions generated just as in Differential Q-planning, except that Differential TD-planning chooses actions according to policy b and not arbitrarily. Differential TD-planning maintains a table of value estimate $V_n : \mathcal{S} \to \mathbb{R}$ and a reward rate estimate $\bar{R}_n$ and updates them just as in Differential TD-learning (11)–(13) using S_n, A_n, R_n, S'_n instead of $S_t, A_t, R_{t+1}, S_{t+1}$.

Theorem 4 (Informal). *Under the same assumptions made in Theorem 3 (except now for the model MDP corresponding to $\hat{p}$ rather than p) the Differential TD-planning algorithm converges, almost surely, $\bar{R}_n$ to $\hat{r}(\pi)$ and V_n to a solution of v in the state-value Bellman equation (cf.* (10)*) for the model MDP.*

Proof. Essentially as in Theorem 1. Full proof in Appendix B. □

5. Empirical Results for Prediction

In this section we present empirical results with average-reward prediction learning algorithms using the Two Loop task shown in the upper right of Figure 3 (cf. Mahadevan 1996, Naik et al. 2019). This is a continuing MDP with only one action in every state except state 0. Action `left` in state 0 gives an immediate reward of +1 and action `right` leads to a delayed reward of +2 after five steps. The optimal policy is to take the action `right` in state 0 to obtain a reward rate of 0.4 per step. The easier-to-find sub-optimal policy of going `left` results in a reward rate of 0.2.

We performed two prediction experiments: on-policy and off-policy. For the first on-policy experiment, the policy π to be evaluated was the one that randomly picks `left` or `right` in state 0 with probability 0.5. The reward rate corresponding to this policy is 0.3. In addition to the on-policy version of Differential TD-learning, we ran Tsitsiklis and Van Roy's (1999) Average Cost TD-learning. It is an on-policy algorithm with the following updates:

$$V_{t+1}(S_t) \doteq V_t(S_t) + \alpha_t \delta_t,$$
$$\bar{R}_{t+1} \doteq \bar{R}_t + \eta \alpha_t \big(R_{t+1} - \bar{R}_t\big), \tag{14}$$

where δ_t is the TD error as in (12). Both algorithms have the same two step-size parameters. For each parameter setting, 30 runs of 10,000 steps each were performed.

We evaluated the accuracy of the estimated value function as well as the estimated reward rate of the target policy. The top-left panel in Figure 3 shows the learning curves of the two algorithms (blue and orange) in terms of root-mean-squared value error (RMSVE) w.r.t. timesteps. We used Tsitsiklis and Van Roy's (1999) variant of RMSVE which measures the distance of the estimated values to the nearest solution that satisfies the state-value Bellman equation (10). We denote this metric by 'RMSVE (TVR)'. Details on how it was computed are provided in Appendix C along with the complete experimental details. We saw that the RMSVE (TVR) went to zero in a few thousand steps for both on-policy Differential TD-learning and Average Cost TD-learning. The top-right panel shows the learning curves of the two algorithms (blue and orange) in terms of squared error in the estimate of the reward rate w.r.t. the true reward rate of the target policy ($(r(\pi) - \bar{R}_t)^2$, denoted as reward rate error or 'RRE'), which also went to zero for both algorithms.

The plots in the bottom row indicate the sensitivity of the performance of these two algorithms to the two step-size parameters α and η. The average RMSVE (TVR) over all the 10k timesteps was equal or lower for Differential TD-learning than Average Cost TD-learning across the range of parameters tested. In addition, on-policy Differential TD-learning was less sensitive to the values of both α and η than Average Cost TD-learning. This was also the case with RRE, the plots for which are reported in Appendix C.

The green learning curves in the top row of Figure 3 correspond to the off-policy version of Differential TD-learning. This was used in the second off-policy experiment: the same policy as in the on-policy experiment was evaluated (i.e., target policy takes either action in state 0 with probability 0.5), now using data collected with a behavior policy that picks the `left` and `right` actions with probabilities 0.9 and 0.1 respectively. Both RMSVE (TVR) and RRE went to zero for off-policy Differential TD-learning within a reason-

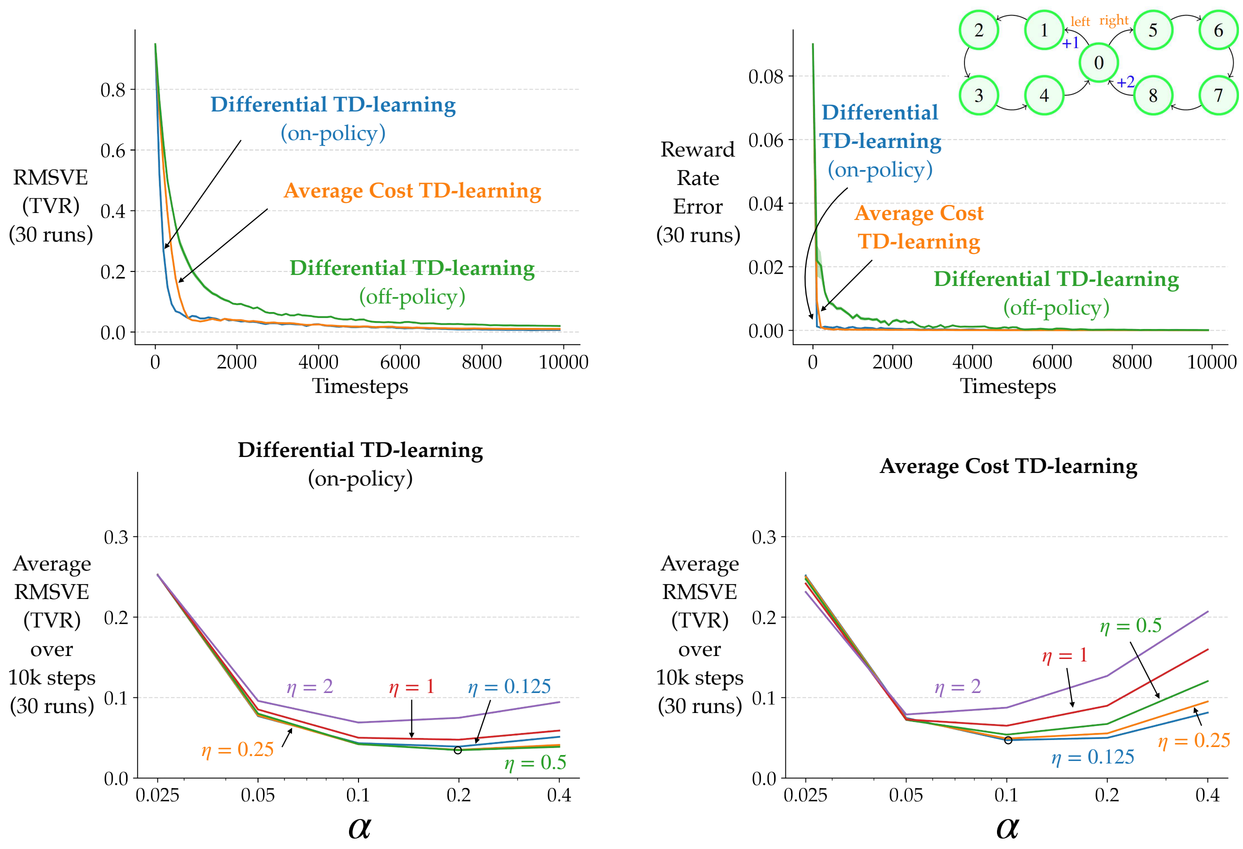

Figure 3: Learning curves and parameter studies for Differential TD-learning and Average Cost TD-learning on the Two Loop task (inset top-right). The standard errors are thinner than width of the solid lines. *Top:* Exemplary learning curves showing all three algorithms tend to zero errors in terms of RMSVE (TVR) and RRE. *Bottom:* Parameter studies showing the performance of Differential TD-learning in terms of average RMSVE (TVR) is less sensitive to the choice of parameters α and η than Average Cost TD-learning. The black circles in the bottom row denote the parameter configurations for which the learning curves in the top row are shown.

able amount of time. Its parameter studies for both RMSVE (TVR) and RRE are presented in Appendix C along with additional experimental details.[1]

Our experiments show that our on- and off-policy Differential TD-learning algorithms can accurately estimate the value function and the reward rate of a given target policy, as expected from Theorem 3. In addition, on-policy Differential TD-learning can be easier to use than Average Cost TD-learning.

6. Estimating the Actual Differential Value Function

All average-reward algorithms, including the ones we proposed, converge to an *uncentered* differential value function, in other words, the actual differential value function plus some unknown offset that depends on the algorithm itself and design choices such as initial values or reference states.

We now introduce a simple technique to compute the offset in the value estimates for both on- and off-policy learning and planning. Once the offset is computed, it can simply be subtracted from the value estimates to obtain the estimate of the actual (centered) differential value function.

We demonstrate how the offset can be eliminated in Differential TD-learning; the other cases (Differential TD-planning, Differential Q-learning and Differential Q-planning) are shown in Appendix B. For this purpose, we introduce, in addition to the first estimator (11)–(13), a second estimator for which the rewards are the value estimates of the first estimator. The second estimator maintains an estimate of the scalar offset $\bar{V}_t$, an auxiliary table of estimates $F_t(s), \forall s \in \mathcal{S}$, and uses the following update rules:

$$\begin{aligned} F_{t+1}(S_t) &\doteq F_t(S_t) + \beta_t \rho_t \Delta_t, \\ F_{t+1}(s) &\doteq F_t(s),\ \forall s \neq S_t, \end{aligned} \tag{15}$$

where β_t is a step-size sequence, Δ_t is the TD error of the second estimator:

$$\Delta_t \doteq V_t(S_t) - \bar{V}_t + F_t(S_{t+1}) - F_t(S_t), \tag{16}$$

where:

$$\bar{V}_{t+1} \doteq \bar{V}_t + \kappa \beta_t \rho_t \Delta_t, \tag{17}$$

and κ is a positive constant. We call (11)–(13) with (15)–(17) *Centered Differential TD-learning*. Before presenting the convergence theorem, we briefly give some intuition on why this technique can successfully compute the offset. By Theorem 3, $\bar{R}_t$ converges to $r(\pi)$ and V_t converges to some v_∞ almost surely, where $v_\infty(s) = v_\pi(s) + c, \forall s \in \mathcal{S}$ for some offset $c \in \mathbb{R}$. In Appendix B, we show $\sum_s d_\pi(s) v_\pi(s) = 0$, where d_π is the limiting state distribution following policy π, which implies $\sum_s d_\pi(s) v_\infty(s) = c$. As V_t converges to v_∞, $\sum_s d_\pi(s) V_t(s)$ converges to c. Now note that $\sum_s d_\pi(s) V_t(s)$ and $r(\pi) = \sum_s d_\pi(s) r_\pi(s)$ are of the same form. Therefore $\sum_s d_\pi(s) V_t(s)$ can be estimated similar to how $r(\pi)$ is estimated, using V_t as the reward. This leads to the second estimator: (15)–(17).

Theorem 5 (Informal)**.** *If the assumptions in Theorem 3 hold, and the step sizes, specific to each state, are decreased appropriately, then Centered Differential TD-learning* (15)–(17) *converges, almost surely,* $V_t(s) - \bar{V}_t$ *to* $v_\pi(s)$ *for all* s *and* $\bar{R}_t$ *to* $r(\pi)$*.*

The proof is presented in Appendix B. We also demonstrate how this technique can be used to learn the actual differential value function with an experiment in Appendix C (with full pseudocode in Appendix A).

7. Discussion and Future Work

We have presented several new learning and planning algorithms for average-reward MDPs. Our algorithms differ from previous work in that they do not involve reference functions, they apply in off-policy settings for both prediction and control, and they find centered value functions. In our opinion, these changes make the average-reward formulation more appealing for use in reinforcement learning.

The most important way in which our work is limited is that it treats only the tabular case, whereas some form of function approximation is necessary for large-scale applications and the larger ambitions of artificial intelligence. Indeed, the need for function approximation is a large part of the motivation for studying the average-reward setting. We present some ideas for extending our algorithms to linear function approximation in Appendix E. However, the theory and practice are both more challenging in the function approximation setting, and much future research is needed.

Our work is also limited in ways that are unrelated to function approximation. One is that we treat only one-step methods and not n-step, λ-return, or sophisticated eligibility-trace methods (van Seijen et al. 2016, Sutton & Barto 2018). Another important direction for future work is to extend these algorithms to semi-Markov decision processes so that they can be used with temporal abstractions like options (Sutton, Precup, & Singh 1999).

Acknowledgements

The authors were supported by DeepMind, Amii, NSERC, and CIFAR. The authors wish to thank Vivek Borkar, Huizhen Yu, Martha White, Csaba Szepesvári, Dale Schu-

[1]Average Cost TD-learning cannot be extended to the off-policy setting due to the use of a sample average of the observed rewards to estimate the reward rate (14). For more details, please refer to Appendix D.

urmans, and Benjamin Van Roy for their valuable feedback during various stages of the work. Computing resources were provided by Compute Canada.

References

Abounadi, J., Bertsekas, D., Borkar, V. S. (1998). *Stochastic Approximation for Nonexpansive Maps: Application to Q-Learning*, Report LIDS-P-2433, Laboratory for Information and Decision Systems, MIT.

Abounadi, J., Bertsekas, D., Borkar, V. S. (2001). Learning algorithms for Markov decision processes with average cost. *SIAM Journal on Control and Optimization, 40*(3):681–698.

Abbasi-Yadkori, Y., Bartlett, P., Bhatia, K., Lazic, N., Szepesvari, C., Weisz, G. (2019a). POLITEX: Regret bounds for policy iteration using expert prediction. In *Proceedings of the International Conference on Machine Learning*, pp. 3692–3702.

Abbasi-Yadkori, Y., Lazic, N., Szepesvari, C., Weisz, G. (2019b). Exploration-enhanced POLITEX. *ArXiv:1908.10479.*

Auer, R., Ortner, P. (2006). Logarithmic online regret bounds for undiscounted reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 49–56.

Bellman, R. E. (1957). *Dynamic Programming*. Princeton University Press.

Bertsekas, D. P., Tsitsiklis, J. N. (1996). *Neuro-dynamic Programming*. Athena Scientific.

Borkar, V. S. (1998). Asynchronous stochastic approximations. *SIAM Journal on Control and Optimization, 36*(3):840–851.

Borkar, V. S. (2009). *Stochastic Approximation: A Dynamical Systems Viewpoint*. Springer.

Brafman, R. I., Tennenholtz, M. (2002). R-MAX — a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research, 3*(10):213–231.

Das, T. K., Gosavi, A., Mahadevan, S. Marchalleck, N. (1999). Solving semi-Markov decision problems using average reward reinforcement learning. *Management Science, 45*(4):560–574.

Dewanto, V., Dunn, G., Eshragh, A., Gallagher, M., Roosta, F. (2020). Average-reward model-free reinforcement learning: a systematic review and literature mapping. *ArXiv:2010.08920.*

Gosavi, A. (2004). Reinforcement learning for long-run average cost. *European Journal of Operational Research, 155*(3):654–674.

Howard, R. A. (1960). *Dynamic Programming and Markov Processes*. MIT Press.

Jalali, A., Ferguson, M. J. (1989). Computationally efficient adaptive control algorithms for Markov chains. In *Proceedings of the IEEE Conference on Decision and Control*, pp. 1283–1288.

Jalali, A., Ferguson, M. J. (1990). A distributed asynchronous algorithm for expected average cost dynamic programming. In *Proceedings of the IEEE Conference on Decision and Control*, pp. 1394–1395.

Jaksch, T., Ortner, R., Auer, P. (2010). Near-optimal Regret Bounds for Reinforcement Learning. *Journal of Machine Learning Research, 11*(4):1563–1600.

Kakade, S. M. (2001). A natural policy gradient. In *Advances in Neural Information Processing Systems*, pp. 1531–1538.

Kearns, M., Singh, S. (2002). Near-optimal reinforcement learning in polynomial time. *Machine Learning, 49*(2):209–232.

Konda, V. R., (2002). *Actor-critic algorithms*. Ph.D. dissertation, MIT.

Liu, Q., Li, L., Tang, Z., Zhou, D. (2018). Breaking the curse of horizon: Infinite-horizon off-policy estimation. In *Advances in Neural Information Processing Systems*, pp. 5356–5366.

Mahadevan, S. (1996). Average reward reinforcement learning: Foundations, algorithms, and empirical results. *Machine Learning, 22*(1–3):159–195.

Marbach, P., Tsitsiklis, J. N. (2001). Simulation-based optimization of Markov reward processes. *IEEE Transactions on Automatic Control, 46*(2):191–209.

Mousavi, A., Li, L., Liu, Q., Zhou, D. (2020). Black-box off-policy estimation for infinite-horizon reinforcement learning. *ArXiv:2003.11126.*

Naik, A., Shariff, R., Yasui, N., Sutton, R. S. (2019). Discounted reinforcement learning is not an optimization problem. *Optimization Foundations for Reinforcement Learning Workshop at the Conference on Neural Information Processing Systems.* Also arXiv:1910.02140.

Puterman, M. L. (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming.* John Wiley & Sons.

Ren, Z., Krogh, B. H. (2001). Adaptive control of Markov chains with average cost. *IEEE Transactions on Automatic Control, 46*(4):613–617.

Schwartz, A. (1993). A reinforcement learning method for maximizing undiscounted rewards. In *Proceedings* *of the International Conference on Machine Learning,* pp. 298–305.

Schweitzer, P. J., & Federgruen, A. (1978). The Functional Equations of Undiscounted Markov Renewal Programming. *Mathematics of Operations Research, 3*(4), pp. 308–321.

Singh, S. P. (1994). Reinforcement learning algorithms for average-payoff Markovian decision processes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 700–705.

Sutton, R. S. (1990). Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings of the International Conference on Machine Learning*, pp. 216–224.

Sutton, R. S., McAllester, D. A., Singh, S. P., Mansour, Y. (1999). Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, pp. 1057–1063.

Sutton, R. S., Barto, A. G. (2018). *Reinforcement Learning: An Introduction.* MIT Press.

Tang, Z., Feng, Y., Li, L., Zhou, D., Liu, Q. (2019). Doubly robust bias reduction in infinite horizon off-policy estimation. *ArXiv:1910.07186.*

Tsitsiklis, J. N., Van Roy, B. (1999). Average cost temporal-difference learning. *Automatica, 35*(11):1799–1808.

van Seijen, H., Mahmood, A. R., Pilarski, P. M., Machado, M. C., Sutton, R. S. (2016). True online temporal-difference learning. *Journal of Machine Learning Research, 17*(145):1–40.

Wen, J., Dai, B., Li, L., Schuurmans, D. (2020). Batch Stationary Distribution Estimation. In *Proceedings of the International Conference on Machine Learning*, pp. 10203–10213.

Wheeler, R., Narendra, K. (1986). Decentralized learning in finite Markov chains. *IEEE Transactions on Automatic Control, 31*(6):519–526.

White, D. J. (1963). Dynamic programming, Markov chains, and the method of successive approximations. *Journal of Mathematical Analysis and Applications, 6*(3):373–376.

Yang, S., Gao, Y., An, B., Wang, H., Chen, X. (2016). Efficient average reward reinforcement learning using constant shifting values. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 2258–2264.

Yu, H., & Bertsekas, D. P. (2009). Convergence results for some temporal difference methods based on least squares. *IEEE Transactions on Automatic Control, 54*(7):1515–1531.

Zhang, R., Dai, B., Li, L., Schuurmans, D. (2020a). GenDICE: Generalized offline estimation of stationary values. *ArXiv:2002.09072.*

Zhang, S., Liu, B., Whiteson, S. (2020b). GradientDICE: Rethinking generalized offline estimation of stationary values. In *Proceedings of the International Conference on Machine Learning*, pp. 11194–11203.