Proceedings of the 2020 Winter Simulation Conference
K.-H. Bae, B. Feng, S. Kim, S. Lazarova-Molnar, Z. Zheng, T. Roeder, and R. Thiesing, eds.

THE ACTOR-CRITIC ALGORITHM FOR INFINITE HORIZON DISCOUNTED COST REVISITED

Abhijit Gosavi

Department of Engineering Management and Systems Engineering
Missouri University of Science and Technology
Rolla, MO 65409, USA

ABSTRACT

Reinforcement Learning (RL) is a methodology used to solve Markov decision processes (MDPs) within simulators. In the classical Actor-Critic (AC), a popular RL algorithm, the values of the so-called actor become unbounded. A recently introduced variant of the AC keeps the actor's values naturally bounded. However, the algorithm's convergence properties have not been established mathematically in the literature. Numerically, the bounded AC was studied under the Boltzmann action-selection strategy, but not under the more popular ϵ-greedy strategy in which the probability of selecting any non-greedy action converges to zero in the limit. The paper revisits the AC framework. A short review of the existing literature in the growing field of ACs is first presented. Thereafter, the algorithm is investigated for its convergence properties, under ϵ-greedy action selection, numerically on a small-scale MDP, as well as mathematically via the ordinary differential equation framework.

1 INTRODUCTION

Reinforcement Learning (Bertsekas and Tsitsiklis 1996; Sutton and Barto 1998) is a methodology used to solve Markov Decision Processes (MDPs) when the transition probabilities are unknown because of the complexity or large-scale of the problem. It can be used in real-time, which is common in the field of artificial intelligence and robotics (Szepesvári 2010) or within discrete-event simulators, which is more common in the operations research and management science community (Gosavi 2015; Chang et al. 2013). The Actor-Critic (AC) is the oldest algorithm in the Reinforcement Learning (RL) family (Barto et al. 1983) and predates Q-Learning (Watkins 1989). Perhaps due to technical difficulties with its applicability to large-scale problems, it was not as popular. However, that has changed, especially after its convergence properties were studied mathematically, and currently there is much interest in this algorithmic framework.

RL algorithms are studied under (a) the infinite time horizon and (b) the finite time horizon. Under each of these time horizons, one can use the so-called *net* discounted reward (or cost in case one is minimizing instead of maximizing) metric and the net undiscounted/average reward metric. The prefix "net" will be dropped in the rest of this paper. See the classical texts of Puterman (1994) and Bertsekas (2007) for a very thorough treatment of this vast subject. In this paper, the focus is on the infinite horizon discounted reward metric.

This paper revisits the AC framework, and in particular, a recently developed bounded version of the AC for discounted reward is investigated for its convergence properties. This bounded AC was proposed under the so-called Boltzmann action-selection strategy. In such a strategy, a function composed of exponential terms is used to select a stochastic policy during the learning process; the final values of the actor are used to determine the deterministic (stationary) policy delivered by the algorithm. A popular action-selection strategy in literature is the so-called ϵ-greedy strategy in which the non-greedy (exploratory) action(s) is selected with high probability at the start; this probability is decayed to zero in the limit (Sutton and Barto

1998). The focus of this paper is on studying the bounded AC for its convergence under the ε-greedy strategy. First, its convergence is tested numerically in a simulator of a small MDP. Thereafter, the convergence is investigated mathematically via the ordinary differential equation (ODE) method.

The rest of this paper is structured as follows: The AC *framework* is first reviewed, along with with a discussion on what has stirred the recent interest in it and its advantages and disadvantages *vis-a-vis* Q-Learning (Section 2). Then, the recently developed bounded AC is considered for the discounted reward metric (Section 3). Thereafter, this algorithm is tested under ε-greedy action selection numerically (Section 4). Finally, convergence properties of the bounded AC are investigated mathematically via the ODE method (Section 5). Conclusions and future work are discussed in the final section (Section 6).

2 BACKGROUND AND LITERATURE REVIEW

This section provides the technical background for the AC framework, along with a short review of the existing literature. It is first necessary to provide the mathematical notation to be used in this paper and the objective function:

2.1 Preliminaries and Notation

Much of the basic notation used in this paper is first presented; some other notation needed for specific topics will be introduced as needed later.

- $\mathscr{S}$: the finite set of states in the system
- $\mathscr{A}(i)$: the finite set of actions permitted in state i
- $\mathscr{A}$: the union of all sets $\mathscr{A}(.)$, i.e., $\cup_{i\in\mathscr{S}}\mathscr{A}(i) \equiv \mathscr{A}$
- $\mu(i)$: the action chosen in state i when deterministic, stationary policy μ is pursued; d also used to denote a stationary, deterministic policy
- $p(i,a,j)$: the transition probability associated with the transition from state i to j under action a
- $r(i,a,j)$: the one-step immediate reward of transition from state i to j under action a
- $\bar{r}(i,a) = \sum_{j\in\mathscr{S}} p(i,a,j)r(i,a,j)$: the one-step expected immediate reward of a transition from i under action a
- λ: the discount factor
- k: the number of iterations in the algorithm
- $V^k(i)$: critic's value for state i in the kth iteration
- $P^k(i,a)$: actor's value for state-action pair (i,a) in the kth iteration
- s_t: state in the tth step of an infinitely long trajectory
- α: a learning rate or step-size rule; β is another learning rate

The objective function under consideration in this paper is the discounted reward metric on an infinite horizon, which is defined as follows:

$$\Lambda_i(\mu) = \lim_{k\to\infty} \mathsf{E}\left[\sum_{t=1}^{k} \lambda^{t-1} r(s_t, \mu(s_t), s_{t+1}) \middle| s_1 = i\right].$$

The goal in this context is to identify a policy in which the above metric, i.e., $\Lambda_i(.)$, is maximized for all values of $i \in \mathscr{S}$.

2.2 The Actor-Critic Framework

A broad structure of the AC framework is now provided. It should be noted that the AC framework relies on the Bellman equation for a given policy used in policy iteration (Howard 1960). This Bellman equation,

which has a unique solution for the discounted reward over an infinite time horizon, is given as:

$$h_\mu(i) = \left[\bar{r}(i,\mu(i)) + \lambda \sum_{j \in \mathscr{S}} p(i,\mu(i),j) h_\mu(j) \right] \quad \text{for all } i \in \mathscr{S}, \tag{1}$$

where the unknowns in the linear equation above are the elements of the value function for a given policy μ: $h_\mu(i)$ for $i \in \mathscr{S}$. The above equation can be solved easily as it is a linear system of equations, provided the transition probabilities are known. The following updating equation, used within the simulators, is based on the above for the so-called *critic* in the AC framework. It is also based on ideas in the Robbins-Monro stochastic approximation (Robbins and Monro 1951) and was first developed in Barto et al. (1983):

$$V(i) \leftarrow (1-\beta)V(i) + \beta \left[r(i,a,j) + \lambda V(j) \right],$$

where $V(i)$ is the proxy for h_μ in Equation (1) above and is updated when the state i is visited in the simulator. This forms the critic update in the AC framework, which can be written as:

$$V(i) \leftarrow V(i) + \beta \left[r(i,a,j) + \lambda V(j) - V(i) \right] \equiv V(i) + \beta \left[\textit{TD-error} \right],$$

where the *TD-error* is defined as $[r(i,a,j) + \lambda V(j) - V(i)]$. The phrase *TD-error* has origins in the RL community, where TD denotes temporal differences; *TD-error* can simply be viewed as the feedback obtained from the environment or the simulator. The so-called *actor* works in conjunction with the critic; the critic's job is to estimate the optimal value function, while the actor's job is to select actions using the so-called actor's values. There is one actor-value, $P(i,a)$, for each state-action pair, (i,a). In the classical AC (Barto et al. 1983), the actor's value is updated as follows: $P(i,a) \leftarrow P(i,a) + \alpha \left[\textit{TD-error} \right]$, where α is another learning rate (step size) and the *TD-error* is as defined above. As the system (or its simulator) transitions from one state to the next, feedback is gathered for updating the actor and the critic values. Ideally, when this training ends, the algorithm delivers the optimal solution (policy).

The above discussion was meant to provide the intuition under this framework, while a bare-bones structural format for the AC framework is presented in Figure 1. When the algorithm terminates, it ideally

AC Structure

- Inputs: Initialize input values; $k_{\max}$ will denote the maximum number of iterations for which the algorithm is run.
- Loop until $k = k_{\max}$
  - Let i be the current state. Select an action. Let j be the next state.
  - Actor's Update: Update the actor's values from the *TD-error* calculated from the state transition.
  - Critic's Update: Update the critic's values from the same *TD-error*. In case of average reward, the average-reward parameters are also updated, typically with the critic's values (Lawhead and Gosavi 2019).
  - Set $k \leftarrow k+1$. If $k = k_{max}$, exit loop; otherwise, set $i \leftarrow j$ and continue within loop.
- Outputs: Extract the optimal policy from the actor's values.

Figure 1: A Template for An Actor Critic

delivers the optimal policy, i.e., the critic values equate or approach the solution of the classical Bellman *optimality* equation that provides the foundation for solving the discounted reward MDP optimally. The following provides the key result associated to the optimality equation (Puterman 1994; Bertsekas 2007):

Theorem 1 The system of equations defined by:

$$h(i) = \max_{a \in \mathscr{A}(i)} \left[\bar{r}(i,a) + \lambda \sum_{j \in \mathscr{S}} p(i,a,j) h(j) \right] \quad \text{for all } i \in \mathscr{S}, \tag{2}$$

where $h(i) \in \Re$ for all $i \in \mathscr{S}$, has a unique solution, $h^*(i)$ for all $i \in \mathscr{S}$, that leads to an optimal solution of the MDP, i.e., if for all $i \in \mathscr{S}$,

$$\mu^*(i) \in \arg\max_{a \in \mathscr{A}(i)} \left[\bar{r}(i,a) + \lambda \sum_{j \in \mathscr{S}} p(i,a,j) h^*(j) \right],$$

then μ^* is an optimal (deterministic and stationary) policy of the MDP.

2.3 Literature Review

The goal of this subsection is to provide a brief review of the growing field of ACs. As stated above, the AC was first proposed in Barto et al. (1983); see also Witten (1977) for earlier research on this topic. Konda and Borkar (1999) studied the convergence properties of the AC. Their analysis showed that the actor's values can get unbounded, and they used a projection to keep these values bounded for facilitating convergence. The recently developed bounded AC in Lawhead and Gosavi (2019) seeks to update the actor's values in a way that does *not* need a projection, but keeps the values bounded regardless; their study also tested the bounded AC framework on a large-scale airline revenue management problem under the Boltzmann action-selection strategy. While a great deal of theoretical work has occurred in related areas, much of it is on analyzing convergence when combined with function approximation (Haarnoja et al. 2020; Fujimoto et al. 2020) and on the related policy gradient algorithm with parameterized policies (Agarwal et al. 2020). In contrast, the focus here is on a different AC algorithm that does *not* require the projection but is along the lines of the classical actor-critic (Barto et al. 1983; Konda and Borkar 1999); further a tabular (lookup-table-based) approach is used in this study. Clearly, convergence under function approximation is a topic worth pursuing *after* the tabular approach is shown to converge.

The use of Q-values for critics and the first direct connection to the policy-gradient framework (Baxter and Bartlett 2001) can be found in Konda and Tsitsiklis (2003). There is growing interest in the natural (gradient) AC, which was first developed in Peters and Schaal (2008). The interested reader is referred to Grondman et al. (2012), who provide a comprehensive survey of numerous aspects of the AC, especially in the context of function approximation, as well as the connection to the policy-gradient framework and the use of Q-values, instead of the value function, as critics.

ACs are known to have some advantages and disadvantages in comparison to Q-Learning. It has been known from the early days of RL that Q-Learning has proven to be difficult to blend with function approximation on large-scale problems (Baird 1995). The Q-Learning algorithm's update is based on the Bellman *optimality* equation and is hence *non-linear*, which is perhaps why it is difficult to approximate the equation via function approximators. In contrast, the AC is based on the fixed policy equation (see Equation (1)), which is linear that makes it relatively easier to approximate the value function. One disadvantage of the AC over Q-Learning is that it requires additional memory, i.e., storage for the actor's values in addition to the critic values. The other disadvantage is that convergence properties of ACs are not as well understood, but this is expected to change as ACs are investigated more thoroughly.

2.4 Nature of the Bellman Equation in ACs

It should be stressed that the Bellman equation used in the AC is *not* the optimality equation, but rather the linear equation associated to a fixed policy employed in the steps of the policy iteration algorithm of Howard (1960). This equation is also used in the *modified* policy iteration algorithm (see Chapter 6 of Puterman (1994) for additional details). Further, recent, successful work in large-scale function approximation in

RL is based on *approximate* policy iteration (Mnih et al. 2015; Silver et al. 2017), which also uses the Bellman equation for a fixed policy. The algorithm *Q-P*-Learning (Gosavi 2004a) also uses the Bellman equation for a fixed policy and is rooted in (modified) policy iteration. One notable difference between policy-iteration-based algorithms, such as approximate policy iteration (Bertsekas 2018) and *Q-P*-Learning (Gosavi 2015), and the AC is that in the former class of algorithms, the notion of ε-greedy exploration is not used, and rather a fixed policy is evaluated for a finite number of iterations within each policy-evaluation phase of the algorithm. In the AC with ε-greedy exploration, the algorithm updating mechanism uses the equation for a fixed policy, but gradually guides the algorithm toward the optimal policy. In other words, the policy continually changes in the AC as the algorithm makes progress, and in this respect, the AC with ε-greedy exploration is similar to *Q*-Learning with ε-greedy exploration, but the critical difference is that *Q*-Learning uses the Bellman *optimality* equation.

3 BOUNDED ACTOR-CRITIC

The bounded AC for infinite-horizon discounted-reward MDPs from Lawhead and Gosavi (2019) is presented next under ε-greedy action selection.

- Inputs: Set k, the number of iterations, to 0. Initialize all actor, $P^k(.,.)$, and critic, $V^k(.)$, values to zero. Let α^k and β^k denote the learning rates (step sizes) in the kth iteration. Set $k_{\max}$, the maximum number of iterations for which the algorithm is run, to a large number.
- Loop until $k = k_{\max}$
  - Let i be the current state. Using the ε-greedy approach for action selection (discussed in more detail below), select an action, a, Let j be the next state. Let $r(i,a,j)$ denote the immediate reward in the state transition.
  - Actor's update:

$$P^{k+1}(i,a) \leftarrow (1-\alpha^k)P^k(i,a) + \alpha^k \left[r(i,a,j) + \lambda V^k(j) - V^k(i) \right]. \tag{3}$$

  - Critic's update:

$$V^{k+1}(i) \leftarrow (1-\beta^k)V^k(i) + \beta^k \left[r(i,a,j) + \lambda V^k(j) \right]. \tag{4}$$

  - Set $k \leftarrow k+1$. If $k = k_{max}$, exit loop; otherwise, set $i \leftarrow j$ and continue within loop.
- Outputs: The policy, d, delivered by the algorithm, is computed as follows. The action $d(i)$ in state i is: $\arg\max_{b \in \mathscr{A}(i)} P^k(i,b)$.

It should be noted that the classical actor-critic used in much of the literature uses the following update for the actor, which unlike Equation (3), does not use the multiplier $(1-\alpha^k)$ for the old value of the actor ($P^k(i,a)$) and consequently makes the actor's values unbounded, requiring a projection onto a finite interval:

$$P^{k+1}(i,a) \leftarrow P^k(i,a) + \alpha^k \left[r(i,a,j) + \lambda V^k(j) - V^k(i) \right].$$

$\varepsilon-$Greedy Action-Selection: In this kind of an action-selection strategy, a distinction is made between a greedy action(s) and a non-greedy action(s). The greedy action in a current state (i) is one that maximizes the actor's value for that action in current state i in the current iteration (k). Mathematically, this means: the greedy action belongs to the set: $\arg\max_{b \in \mathscr{A}(i)} P^k(i,b)$. (Note that in case the algorithm is presented in terms of minimizing costs, rather than rewards, the set in question would be: $\arg\min_{b \in \mathscr{A}(i)} P^k(i,b)$.) Any action that does not belong to this set in the kth iteration is considered non-greedy (exploratory). In the ε-greedy action-selection strategy, theoretically, the non-greedy action is typically selected with some probability (ε) that is decayed as the algorithm makes progress, and this probability converges to 0 in the limit. There are multiple ways to implement this strategy in practice. Two such ways are discussed next.

- Algebraic Decay: With probability $(1-\frac{B^k}{k})$, select the greedy action and with probability $\frac{B^k}{k(|\mathscr{A}(i)|-1)}$, select any of the other (non-greedy) actions in $\mathscr{A}(i)$, such that

$$\lim_{k\to\infty}\frac{B^k}{k}=0.$$

  An example for the function B^k that satisfies the above conditions is $B^k=1/|\mathscr{A}(i)|$, but any function that satisfies the above limit should work.
- Geometric Rate of Decay: Another approach is to geometrically decay the rate of exploration. Mathematically, this means with probability $(1-B^k)$, select the greedy action and with probability $\frac{B^k}{(|\mathscr{A}(i)|-1)}$, select any of the other (non-greedy) actions in $\mathscr{A}(i)$, such that

$$B^k=B^{k-1}\tau \text{ and } \lim_{k\to\infty}B^k=0,$$

  where $\tau\in(0,1)$ should be close to 1. An example for the above is $B^0=1/|\mathscr{A}(i)|$ and $\tau=0.9999$.

Every learning rate in RL must typically converge to zero and the two learning rates in two timescale settings must satisfy the following condition:

$$\limsup_{k\to\infty}\frac{\beta^k}{\alpha^k}=0, \text{ where} \tag{5}$$

α^k and β^k are the learning rates in the kth iteration on the faster and slower timescale respectively.

4 NUMERICAL CONVERGENCE

The Bounded AC for infinite horizon discounted reward MDPs was tested on a small-scale problem with two states and two actions in each state. The discount factor, λ, was set to equal 0.8. The rest of the inputs for the MDP simulated were as follows:

$$\mathbf{TPM}_1=\begin{bmatrix}0.7 & 0.3\\ 0.4 & 0.6\end{bmatrix};\mathbf{TPM}_2=\begin{bmatrix}0.9 & 0.1\\ 0.2 & 0.8\end{bmatrix};\mathbf{TRM}_1=\begin{bmatrix}16 & 5\\ 75 & 120\end{bmatrix};\mathbf{TRM}_2=\begin{bmatrix}80 & 10\\ 6 & 1\end{bmatrix}.$$

In the above, $\mathbf{TPM}_a$ denotes the one-step transition probability *matrix* of action a and $\mathbf{TRM}_a$ denotes the one-step immediate reward *matrix* for action a. In order to synchronize this notation with the previously defined notation in subsection 2.1 on one-step transition probabilities and reward, note that $TPM_a(i,j)=p(i,a,j)$ and $TRM_a(i,j)=r(i,a,j)$, where $Z(i,j)$ denotes the element in the ith row and jth column of matrix, $\mathbf{Z}$. The action set for each state is $\mathscr{A}=\{1,2\}$. Table 1 shows the results of implementing the bounded AC using ε-greedy action-selection, via a geometric decay, and dynamic programming (DP) on the problem. In the geometric decay of exploration, the following tuning parameters were used: $B^0=0.5$ and $\tau=0.999$. The following learning rates were found to yield the best results:

$$\alpha^k=\frac{logk}{k};\ \ \beta^k=\frac{A}{k+B},$$

with $A=5$ and $B=10$. Using $A=1$ and $B=0$ leads to a popular rule, $1/k$, which did not perform in a satisfactory manner on this simple problem in terms of generating the value function; there is prior evidence of weak behavior of this rule in RL (Gosavi 2008). For problems with a larger dimension, higher values of A and B are generally required (Lawhead and Gosavi 2019). It is not difficult to show that these learning rates satisfy the ratio condition in Equation (5) as follows:

$$\lim_{k\to\infty}\frac{\beta^k}{\alpha^k}=\lim_{k\to\infty}\frac{A/(k+B)}{logk/k}=\lim_{k\to\infty}\frac{A}{(1+\frac{B}{k})logk}=0.$$

The bounded AC was run for $k_{\max} = 10,000$ iterations in a simulator written in MATLAB on a 64-bit, 2.5 GHz windows operating system. The program terminated in less than 0.1 second.

The outputs from the computer program are described next. $P^{k_{\max}}(i,a)$ denotes the actor's value for state-action pair, (i,a), delivered by the algorithm; and these values are: $P^{k_{\max}}(1,1) = -48.3020$; $P^{k_{\max}}(1,2) = 3.9944$; $P^{k_{\max}}(2,1) = 24.3354$; and $P^{k_{\max}}(2,2) = -31.1489$. These values imply that the optimal action in state 1 is 2 and that in state 2 is 1; this stems from the fact that $P^{k_{\max}}(1,2) > P^{k_{\max}}(1,1)$ and $P^{k_{\max}}(2,1) > P^{k_{\max}}(2,2)$. This policy coincides with the optimal policy delivered by the DP algorithm. Figures 2 and 3 depict the critic values generated for states 1 and 2 respectively, during the run-time of the algorithm. The optimal policy is determined within 10 iterations of running the algorithm; however, additional iterations are needed before the actor's and critic's values stabilize. While the critic's values and the value function of DP do not match perfectly, which is usually true of ε-greedy exploration (Gosavi 2004b), what is more significant here is whether the optimal (or near-optimal) *policy* is delivered by the algorithm. Further, ε-greedy techniques make the convergence properties easier to handle mathematically, as the equilibrium (limiting) solution can be shown to satisfy or approach the solution of the Bellman optimality equation (Equation (2)), which guarantees optimality, but is not used within the updating schemes of the AC. As mentioned above, the Bellman equation for a fixed policy (Equation (1)) is used within the AC, and due to its linear nature, it is known to have advantages from the perspective of function approximation. Nonetheless, guiding the critic's values into the vicinity of optimal values should be investigated in the future, potentially via *algebraic* decay of the exploration rate.

Table 1: Performance of the bounded AC and DP: $h^*(i)$ denotes the optimal value function obtained from DP for state i, while $V^{k_{\max}}(i)$ denotes the critic value, i.e., value function, for state i obtained from the AC.

$h^*(1)$	$V^{k_{\max}}(1)$	$h^*(2)$	$V^{k_{\max}}(2)$
384.3288	355.7634	432.6622	376.2387

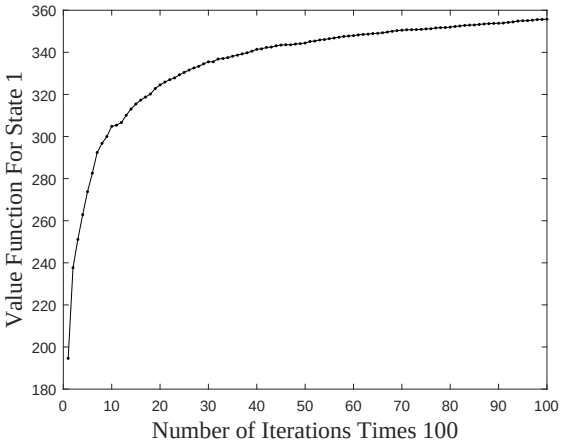

Figure 2: The behavior of the critic value for state 1, $V^k(1)$, as the algorithm makes progress.

5 CONVERGENCE PROPERTIES

The first result presented below is that related to stochastic approximation on two timescales (Borkar 2009). In such a setting, two sets of vectors are updated simultaneously, but different learning rates are used for each set; the learning rates are chosen in a manner such that the sequence defined by the slower learning rate divided by the faster learning rate converges to zero. Before the result is presented formally, some notation needs to be defined. Let $\vec{X}^k$ denote the vector of iterates on the faster time scale and $\vec{Y}^k$ the same on the

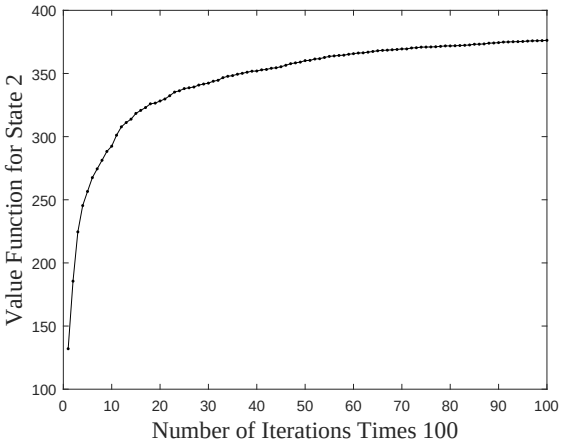

Figure 3: The behavior of the critic value for state 2, $V^k(2)$, as the algorithm makes progress.

slower time scale, assuming there are N_1 and N_2 iterates on the faster and slower timescales respectively. Further, the iterates generate the following sequence:$\{\Theta_1^1, \Theta_1^2, \ldots\}$ and $\{\Theta_2^1, \Theta_2^2, \ldots\}$, where Θ_1^k denotes the iterate from the faster time scale updated in the kth iteration while Θ_2^k denotes the iterate from the slower time scale updated in the kth iteration. Thus for $k = 1, 2, ...$: $\Theta_1^k \in \{1, 2, \ldots, N_1\}$ and $\Theta_2^k \in \{1, 2, \ldots, N_2\}$. The two-timescale algorithm under asynchronous updating can now be described as: For $l = 1, 2.., N_1$ and $l_2 = 1, 2, .., N_2$:

$$\begin{aligned} X^{k+1}(l_1) &= X^k(l_1) + \alpha^k(l_1)\left[G\left(\vec{X}^k, \vec{Y}^k\right)(l_1) + w_1^k(l_1)\right] I(l_1 = \Theta_1^k); \\ Y^{k+1}(l_2) &= Y^k(l_2) + \beta^k(l_2)\left[F\left(\vec{X}^k, \vec{Y}^k\right)(l_2) + w_2^k(l_2)\right] I(l_2 = \Theta_2^k); \end{aligned} \tag{6}$$

where

- $\alpha^k(.)$ and $\beta^k(.)$ are the learning rates on the faster and slower time-scale respectively in the kth iteration
- $G(.,.)$ and $F(.,.)$ denote the transformations driving the faster and slower time-scale updates respectively
- $\vec{w}_1^k$ and $\vec{w}_2^k$ denote the noise vectors in the kth iteration on the faster and slower time scales respectively

The following conditions will be assumed to be true of the stochastic approximation algorithm.
Condition 1. Boundedness of iterates: The iterates $\vec{X}^k$ and $\vec{Y}^k$ remain bounded with probability 1 ($wp1$).
Condition 2. $F(.,.)$ and $G(.,.)$ are Lipschitz continuous.
Condition 3. Conditions on noise: For $l_1 = 1, 2, \ldots, N_1$, $l_2 = 1, 2, \ldots, N_2$, and for every k, the following should be true about the noise terms:

$$\mathsf{E}\left[w_1^k(l_1)|\mathscr{F}^k\right] = 0; \quad \mathsf{E}\left[\left(w_1^k(l_1)\right)^2 \middle| \mathscr{F}^k\right] \leq z_1 + z_2||\vec{X}^k||^2 + z_3||\vec{Y}^k||^2;$$

$$\mathsf{E}\left[w_2^k(l_2)|\mathscr{F}^k\right] = 0; \quad \mathsf{E}\left[\left(w_2^k(l_2)\right)^2 \middle| \mathscr{F}^k\right] \leq z_1' + z_2'||\vec{X}^k||^2 + z_3'||\vec{Y}^k||^2;$$

where z_1, z_2, z_3, z_1', z_2', and z_3' are scalar constants and $||.||$ could be any norm.
Condition 4. Learning rate conditions of stochastic approximation: The learning rates satisfy the usual tapering size conditions for every $l_1 = 1, 2, \ldots, N_1$ and $l_2 = 1, 2, \ldots, N_2$:

$$\sum_{k=1}^{\infty} \alpha^k(l_1) = \infty; \quad \sum_{k=1}^{\infty} \left(\alpha^k(l_1)\right)^2 < \infty; \quad \sum_{k=1}^{\infty} \beta^k(l_2) = \infty; \quad \sum_{k=1}^{\infty} \left(\beta^k(l_2)\right)^2 < \infty.$$

In addition, the learning rates must satisfy the timescale ratio condition for every (l_1, l_2) pair defined in Equation (5). Further, the learning rates must also satisfy the evenly distributed and ideal tapering size assumptions defined in Borkar (2009).

Condition 5a. ODE condition: For any fixed value of $\vec{y} \in \Re^{N_2}$, the ODE

$$\frac{d\vec{x}}{dt} = G(\vec{x}, \vec{y}) \tag{7}$$

has a globally asymptotically stable equilibrium point which is a function of $\vec{y}$ and will be denoted by $\Omega(\vec{y})$, where $\Omega : \Re^{N_2} \rightarrow \Re^{N_1}$. Further, the function $\Omega(\vec{y})$ has to be Lipschitz continuous in $\vec{y}$.

Condition 5b. ODE condition: The following ODE has a globally asymptotically stable equilibrium $\vec{y}_*$

$$\frac{d\vec{y}}{dt} = F\left(\Omega(\vec{y}), \vec{y}\right). \tag{8}$$

Theorem 2 Consider the two-timescale asynchronous algorithm defined in Equation (6). Assume that Conditions 1 through 5 hold. Then, $wp1$, the sequence of iterates $\left\{\vec{X}^k, \vec{Y}^k\right\}_{k=1}^{\infty}$ converges to $(\Omega(\vec{y}_*), \vec{y}_*)$.

The above result will be exploited to establish convergence of the bounded actor-critic via the following result.

Theorem 3 Consider the AC algorithm defined via the updates in Equations (3) and (4). $Wp1$, the algorithm will deliver the Bellman-optimal policy in the limit.

Proof. (Sketch) Note that in the algorithm, $N_1 = |\mathscr{S}| \cdot |\mathscr{A}|$ and $N_2 = |\mathscr{S}|$. Further, the action chosen in state i in the kth iteration depends on the action-value matrix $\mathbf{P}^k$, which can be represented via a vector that will be denoted as $\vec{P}^k$. In the analysis, $\vec{X}^k \equiv \vec{P}^k$ and $\vec{Y}^k \equiv \vec{V}^k$; further $l_1 \equiv (i,a)$ and $l_2 \equiv i$. All the conditions are now inspected and shown to hold true under some assumptions. The following presents a sketch of the proof for verifying the conditions. First, the transformations and the noise terms have to be defined in the context of the algorithm under consideration:

Transformations for the Actor: The transformations $G(.,.)$ and $G'(.,.)$ associated to the actor update are defined as follows:

$$G\left(\vec{P}^k, \vec{V}^k\right)(i,a) = \sum_{j=1}^{|\mathscr{S}|} p(i,a,j)\left[r(i,a,j) + \lambda V^k(j) - V^k(i)\right] - P^k(i,a) \quad \forall (i,a); \tag{9}$$

$$G'\left(\vec{P}^k, \vec{V}^k\right)(i,a) = \sum_{j=1}^{|\mathscr{S}|} p(i,a,j)\left[r(i,a,j) + \lambda V^k(j)\right] - V^k(i) \quad \forall (i,a);$$

$$\text{which implies that } G\left(\vec{P}^k, \vec{V}^k\right)(i,a) = G'\left(\vec{P}^k, \vec{V}^k\right)(i,a) - P^k(i,a) \quad \forall (i,a). \tag{10}$$

Another transformation $g'(.,.)$ is defined as follows, noting that the transformation $g'(.,.)$ is over the sample of which $G'(.,.)$ computes the expectation:

$$g'\left(\vec{P}^k, \vec{V}^k\right)(i,a) = \left[r(i,a,\xi^k) + \lambda V(\xi^k)\right] - V^k(i),$$

where ξ^k is the random state reached when action $u^k(i)$ is chosen in state i in the kth iteration. Now, a noise term has to be defined as:

$$w_1^k(i,a) = g'\left(\vec{P}^k, \vec{V}^k\right)(i,a) - G'\left(\vec{P}^k, \vec{V}^k,\right)(i,a) \quad \forall (i,a).$$

Then, one can write the actor updating transformation in the algorithm (Equation (3)) as:

$$P^{k+1}(i,a) = P^k(i,a) + \alpha^k \left[g' \left(\vec{P}^k, \vec{V}^k \right) (i) - P^k(i,a) \right] \quad \forall (i,a).$$

Now, using the relation defined in Equation (10), one can re-write the above as:

$$P^{k+1}(i,a) = P^k(i,a) + \alpha^k \left[G \left(\vec{P}^k, \vec{V}^k \right) (i) + w_1^k(i,a) \right] \quad \forall (i,a). \tag{11}$$

Transformations for the Critic: The action selected in state i in the kth iteration will be denoted by $u^k(i)$ in this case. The transformations $F(.,.)$ and $F'(.,.)$ associated to the critic update are defined as follows :

$$F \left(\vec{P}^k, \vec{V}^k \right) (i) = \sum_{j=1}^{|\mathscr{S}|} p(i,u^k(i),j) \left[r(i,u^k(i),j) + \lambda V^k(j) \right] - V^k(i) \quad \forall i;$$

$$F' \left(\vec{P}^k, \vec{V}^k \right) (i) = \sum_{j=1}^{|\mathscr{S}|} p(i,u^k(i),j) \left[r(i,u^k(i),j) + \lambda V^k(j) \right] \quad \forall i;$$

$$\text{which implies that } F \left(\vec{P}^k, \vec{V}^k \right) (i) = F' \left(\vec{P}^k, \vec{V}^k \right) (i) - V^k(i) \quad \forall i. \tag{12}$$

Another transformation $f'(.)$ is defined as follows, analogous to $g'(.)$ above:

$$f' \left(\vec{P}^k, \vec{V}^k \right) (i) = \left[r(i,u^k(i),\xi^k) + \lambda V(\xi^k) \right] \quad \forall i,$$

where ξ^k is the random state reached when action $u^k(i)$ is chosen in state i in the kth iteration. Now, the noise term for the critic update can be defined as:

$$w_2^k(i) = f' \left(\vec{P}^k, \vec{V}^k \right) (i) - F' \left(\vec{P}^k, \vec{V}^k, \right) (i) \quad \forall i.$$

Then, one can write the updating transformation in the algorithm for the critic (Equation (4)) as:

$$V^{k+1}(i) = V^k(i) + \beta^k \left[f' \left(\vec{P}^k, \vec{V}^k \right) (i) - V^k(i) \right].$$

Now, using the relation defined in Equation (12), one can re-write the above:

$$V^{k+1}(i) = V^k(i) + \beta^k \left[F \left(\vec{P}^k, \vec{V}^k \right) (i) + w_2^k(i) \right]. \tag{13}$$

Conditions $1-4$**:** Condition 1 has been established in Lawhead and Gosavi (2019). Condition 2 is true since these transformations are linear functions. The conditional second moment of each noise term in Condition 3 remains bounded because the iterates themselves are bounded. Under ε-greedy action selection, the GLIE (Greedy in the Limit with Infinite Exploration) property holds, which implies that all the state-action pairs are tried infinitely often (Singh et al. 2000) and so the conditional mean of both noise terms in Condition 3 equals zero. The assumptions in Condition 4 can be shown to hold under suitable learning rates.

Condition 5: For a fixed set of values for the vector $\vec{V}^k$, the update defined in (11) behaves like a Robbin-Monro scheme seeking to estimate the expectation of a constant from samples and hence converges (Bertsekas and Tsitsiklis 1996), implying $G(\vec{P}^\infty, \vec{V}^\infty) = 0$. Details are omitted. Thus, the ODE in (7) has a unique globally asymptotically stable equilibrium. The equilibrium will be a linear function of $\vec{V}$ and will hence be Lipschitz continuous in $\vec{V}$. Transformation $F(.,.)$ is the classical dynamic programming operator (of value iteration) and is known to be contractive (Puterman 1994). Hence, the update defined in (13)

converges, implying $F(\vec{P}^{\infty}, \vec{V}^{\infty}) = 0$. Thus, the ODE in (8) has a unique globally asymptotically stable equilibrium. From Theorem 2, one now has convergence to a fixed point for each class of iterates $wp1$.

While the above shows that the algorithm converges, what remains to be shown is that it does so to the optimal solution. Since the action used in the critic update, $u(.)$, is greedy in the limit, for each $i \in \mathscr{S}$:

$$u^{\infty}(i) \in \arg\max_{b \in \mathscr{A}(i)} P^{\infty}(i,b) = \arg\max_{b \in \mathscr{A}(i)} \sum_{j=1}^{|\mathscr{S}|} p(i,b,j) \left[r(i,b,j) + \lambda V^{\infty}(j) - V^{\infty}(i) \right] \tag{14}$$

$$= \arg\max_{b \in \mathscr{A}(i)} \sum_{j=1}^{|\mathscr{S}|} p(i,b,j) \left[r(i,b,j) + \lambda V^{\infty}(j) \right]. \tag{15}$$

In the above, note that the relation in (14) follows from the fact that in the limit, the actor values converge, implying $G(.,.) = 0$, i.e., from Equation (9), one has that

$$P^{\infty}(i,b) = \sum_{j=1}^{|\mathscr{S}|} p(i,b,j) \left[r(i,b,j) + \lambda V^{\infty}(j) - V^{\infty}(i) \right] \quad \forall (i,b).$$

Note that (14) and (15) are equivalent because a constant is being subtracted. Finally, relationship (15) implies that the Bellman optimal action is selected in the limit for each $i \in \mathscr{S}$. □

6 CONCLUSIONS

The paper revisited the domain of actor-critic (AC) algorithms for discounted cost under the infinite time horizon. The classical algorithm (Barto et al. 1983) is known to produce unboundedness of the actor's values. However, the algorithm is growing in popularity (Grondman et al. 2012). This paper studied a bounded version of the discounted reward AC in which the actor's values remain naturally bounded without a projection and with an ε-greedy action selection. The algorithm was shown to have satisfactory numerical convergence on a problem whose optimal solution is known. Further, a proof of theoretical convergence was sketched. Future work in this area will study three extensions: (i) experimenting with a real-world semi-MDP problem via the bounded AC, (ii) testing a suitable function-approximation scheme for the algorithm, and (iii) fully developing the convergence proof sketched here.

REFERENCES

Agarwal, A., S. Kakade, J. Lee, and G. Mahajan. 2020. "Optimality and Approximation with Policy Gradient Methods in Markov Decision Processes". https://arxiv.org/abs/1908.00261, accessed 8^{th} August.

Baird, L. 1995. "Residual Algorithms: Reinforcement Learning with Function Approximation". In *Proceedings of the Twelfth International Conference on Machine Learning*, 30–37: Morgan Kaufmann.

Barto, A., R. Sutton, and C. Anderson. 1983. "Neuronlike Elements that can Solve Difficult Learning Control Problems". *Institute of Electrical and Electronics Engineers Transactions on Systems, Man, and Cybernetics* 13:835–846.

Baxter, J., and P. Bartlett. 2001. "Infinite-Horizon Policy-Gradient Estimation". *Journal of Artificial Intelligence* 15:319–350.

Bertsekas, D. 2007. *Dynamic Programming and Optimal Control*. 3rd ed. Boston, MA: Athena.

Bertsekas, D. 2018. "Feature-Based Aggregation and Deep Reinforcement Learning: A Survey and Some New Implementations". *Institute of Electrical and Electronics Engineers/Chinese Association of Automation Journal of Automatica Sinica* 6(1):1–31.

Bertsekas, D., and J. Tsitsiklis. 1996. *Neuro-Dynamic Programming*. Boston, MA: Athena.

Borkar, V. 2009. *Stochastic Approximation: A Dynamical Systems Viewpoint*. New York, NY, USA: Springer.

Chang, H., M. Fu, J. Hu, and S. I. Marcus. 2013. *Simulation-Based Algorithms for Markov Decision Processes*. second ed. New York, NY: Springer.

Fujimoto, S., H. Van Hoof, and D. Meger. 2020. "Addressing Function Approximation Error in Actor-Critic Methods". https://arxiv.org/abs/1802.09477, accessed 20^{th} August.

Gosavi, A. 2004a. "A Reinforcement Learning Algorithm Based on Policy Iteration for Average Reward: Empirical Results with Yield Management and Convergence Analysis". *Machine Learning* 55:5–29.

Gosavi, A. 2004b. "Reinforcement Learning for Long-Run Average Cost". *European Journal of Operational Research* 155:654–674.

Gosavi, A. 2008. "On Step Sizes, Stochastic Shortest Paths, and Survival Probabilities in Reinforcement Learning". In *2008 Winter Simulation Conference*, edited by S. Mason, R. Hill, L. Moench, and O. Rose, 525–531. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Gosavi, A. 2015. *Simulation-Based Optimization: Parametric Optimization Techniques and Reinforcement Learning*. second ed. New York, NY: Springer.

Grondman, I., L. Busoniu, G. Lopes, and R. Babuska. 2012. "A Survey of Actor-Critic Reinforcement Learning: Standard and Natural Policy Gradients". *Institute of Electrical and Electronics Engineers Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 42(6):1291–1307.

Haarnoja, T., A. Zhou, P. Abbeel, and S. Levine. 2020. "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with A Stochastic Actor". https://arxiv.org/abs/1801.01290, accessed 20^{th} August.

Howard, R. 1960. *Dynamic Programming and Markov Processes*. Cambridge, MA: MIT Press.

Konda, V., and V. S. Borkar. 1999. "Actor-Critic Type Learning Algorithms for Markov Decision Processes". *Society for Industrial and Applied Mathematics Journal on Control and Optimization* 38(1):94–123.

Konda, V., and J. Tsitsiklis. 2003. "On Actor-Critic Algorithms". *Society for Industrial and Applied Mathematics Journal on Control and Optimization* 42(4):1143–1166.

Lawhead, R., and A. Gosavi. 2019. "A Bounded Actor-Critic Reinforcement Learning Algorithm Applied to Airline Revenue Management". *Engineering Applications of Artificial Intelligence* 82:252–262.

Mnih, V., K. Kavukcuoglu, D. Silver, A. Rusu, J. Veness, M. Bellemare, A. Graves, M. Riedmiller, A. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and H. D.. 2015. "Human-Level Control Through Deep Reinforcement Learning". *Nature* 518(7540):529–533.

Peters, J., and S. Schaal. 2008. "Natural Actor-Critic". *Neurocomputing* 71(7-9):1180–1190.

Puterman, M. L. 1994. *Markov Decision Processes*. New York, NY: Wiley Interscience.

Robbins, H., and S. Monro. 1951. "A Stochastic Approximation Method". *Annals of Mathematical Statistics.* 22:400–407.

Silver, D., J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis. 2017. "Mastering the Game of Go Without Human Knowledge". *Nature* 550(7676):354–359.

Singh, S., T. Jaakkola, M. Littman, and C. Szepesvári. 2000. "Convergence Results for Single-Step On-Policy Reinforcement-Learning Algorithms". *Machine Learning* 38(3):287–308.

Sutton, R., and A. G. Barto. 1998. *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press.

Szepesvári, C. 2010. *Algorithms for Reinforcement Learning*. San Rafael, CA: Morgan Claypool.

Watkins, C. 1989. *Learning from Delayed Rewards*. Ph. D. thesis, Kings College, Cambridge, England.

Witten, I. 1977. "An Adaptive Optimal Controller for Discrete Time Markov Environments". *Information and Control* 34:286–295.

AUTHOR BIOGRAPHY

ABHIJIT GOSAVI is an associate professor in the Department of Engineering Management and Systems Engineering at Missouri University of Science and Technology, Rolla, MO, USA. He holds a PhD in industrial engineering from the University of South Florida. His research interests are in simulation-based optimization, Markov decision processes, and lean manufacturing, and he is currently working in the areas of disaster preparedness and health-care sciences. His email address is gosavia@mst.edu.