A Bounded Actor-Critic Reinforcement Learning Algorithm Applied to Airline Revenue Management

Ryan J. Lawhead and *Abhijit Gosavi*
Department of Engineering Management and Systems Engineering
Missouri University of Science and Technology
Rolla, MO 65409

Abstract

Reinforcement Learning (RL) is an artificial intelligence technique used to solve Markov and semi-Markov decision processes. Actor critics form a major class of RL algorithms that suffer from a critical deficiency, which is that the values of the so-called *actor* in these algorithms can become very large causing computer overflow. In practice, hence, one has to artificially constrain these values, via a projection, and at times further use temperature-reduction tuning parameters in the popular Boltzmann action-selection schemes to make the algorithm deliver *acceptable* results. This artificial bounding and temperature reduction, however, do *not* allow for full exploration of the state space, which often leads to sub-optimal solutions on large-scale problems. We propose a new actor-critic algorithm in which (i) the actor's values remain bounded *without any projection* and (ii) no temperature-reduction tuning parameter is needed. The algorithm also represents a significant improvement over a recent version in the literature, where although the values remain bounded they usually become very large in magnitude, necessitating the use of a temperature-reduction parameter. Our new algorithm is tested on an important problem in an area of management science known as airline revenue management, where the state-space is very large. The algorithm delivers encouraging computational behavior, outperforming a well-known industrial heuristic called EMSR-b on industrial data.

Keywords: reinforcement learning; actor critics; airline revenue management

1. Introduction

Reinforcement Learning or RL (see Bertsekas and Tsitsiklis [1], Sutton and Barto [2], Gosavi [3] and Szepesvári [4]) is used to solve problems in which an agent selects optimal actions in an unknown stochastic environment via repeated trials and errors. In every trial, the agent gathers feedback from the environment and uses the feedback to update its knowledge base. Typically, after a large number of trials and errors in its interactions, the agent learns to select an *optimal* action in every state. The Markov Decision Process or Problem (MDP) and the semi-MDP (SMDP) [1] have been extensively used as the underlying models in the above-described RL domain. Essentially, in the MDP model, the underlying system dynamics and behaviour are governed by Markov chains. Further, in an MDP, the time taken in a one-step transition from any state to any other is assumed to be the same. The SMDP is a *more general* model in which this transition time is assumed to be a random variable whose distribution is known.

In this paper, we study a class of RL algorithms known as *actor critics.* The classical actor-critic algorithm [5] predates the more popular RL algorithm, namely, Q-Learning, discovered by Watkins [6]. The classical actor-critic algorithm, however, has the following critical deficiency: the values of one class of iterates of the algorithm, called the actor's values, become unbounded, i.e., become very large in magnitude. This causes the computer to overflow. One approach to alleviate this difficulty is to use a mathematical projection that artificially bounds the actor values [7]. However, this artificial bounding curtails the amount of exploration the algorithm can perform, leading to poor solutions on large-scale problems at times. Kulkarni et al. [8] studied a version of this projection-bounded actor-critic algorithm on a problem from airline revenue management, but the best results from these algorithms were obtained from employing numerous replications (re-runs of the simulations with new sets of random numbers). In other words, at times, the algorithm did *not* explore sufficiently [9]. To alleviate this difficulty, Gosavi [10] proposed a variant of the classical algorithm in which the actor's values were *naturally* bounded; however, experimentation with this algorithm also showed that the magnitude of the actor's values still often become *quite* large. When the magnitude of the values becomes large, one needs to use a temperature-tuning parameter in the Boltzmann action-selection strategy, which unfortunately adds a whole layer to the computational exercise involved in the algorithm [11]. Also, different temperatures lead to different depths of exploration, and hence one must then search for a good temperature for the best exploration. Further, even with this temperature tuning, the algorithm in Gosavi [10] may still explore *insufficiently* on large-scale problems, leading the user to sub-optimal solutions. In other words, overall, the artificial projection as well as the temperature-tuning make it harder to use the algorithm in practice.

Contributions of this paper: In this paper, we present a new algorithm that provides a significant improvement in its performance over that of the above-described past work in the existing literature in the following ways: (i) the actor's values remain naturally bounded, thereby eliminating the need for any artificial projection, and (ii) the actor's values also remain *small in magnitude*, thereby eliminating the need for any temperature tuning with the Boltzmann action-selection. The algorithm is first tested on small-scale problems in this paper to demonstrate both of these features. But, the true test of strength for any RL algorithm is on large-scale problems, where unless the algorithm explores sufficiently, it cannot generate satisfactory performance. We hence tested our algorithm on a large-scale problem from airline revenue management with real-world data, where it outperformed a well-known industrial heuristic called EMSR-b [12]. We further note that while the λ-SMART algorithm [13] and the projection-bounded actor-critic algorithm in [8] are other examples of RL algorithms that have been applied to the airline revenue management problem in the past, the λ-SMART algorithm is based on a finite trajectory, which may not be applicable to all RL settings, and the projection-bounded actor-critic algorithms of the past do not always return optimal solutions in practice.

The reader interested in exploring the connection of actor-critics to policy gradients [14] is referred to an excellent review paper by Grondman et al. [15]. Actor-critics have also been studied via a control-theoretic viewpoint [16, 17, 18, 19]. Finally, RL algorithms based on Q-Learning [6] and SARSA [20] have been used widely in industrial problems, ranging from preventive maintenance [21, 22] to supply chain management [23, 24, 25] and robotics [26], but industrial-scale applications of actor-critics are not as common as those of traditional Q-Learning-based algorithms.

The rest of this article is organized as follows. Section 2 provides the background on the MDP and SMDP, as well as that of the airline revenue management problem. Section 3 presents the new algorithm, as well as a review of the past work. Section 4 discusses numerical results with using the algorithm. Concluding remarks along with comments on future work are provided in Section 5.

2. Background

This section is divided into two parts: the first subsection is devoted to presenting the mathematical framework underlying MDPs and SMDPs, as well the motivation for using RL, while the second is devoted to a description of the airline revenue management problem.

2.1. MDPs, SMDPs, and RL

As mentioned above, MDPs and SMDPs are useful in modeling interactions in stochastic environments. In such models, in every state visited by the system, a decision must be selected from a set of permitted actions in that state. The objective considered in this paper is to maximize the so-called *average reward*, i.e., the expected reward per unit time over an infinitely long time horizon; in such an objective function, the assumption is that the system settles down into a steady state after a long period of time. In MDPs, the time taken for any transition from one state to another is the same for every transition and is considered to be one unit. In an SMDP, the time of transition is any given random variable, whose distribution is known, and the time is explicitly modeled into the objective function. Since the MDP is a special case of the SMDP in which the time of transition always equals one, we present details of the SMDP. We first present some notation:

- $\mathcal{S}$: the finite set of states in the system

- $\mathcal{A}(i)$: the finite set of actions permitted in state i
- $\mathcal{A}$: the union of all sets $\mathcal{A}(.)$, i.e., $\cup_{i \in \mathcal{S}} \mathcal{A}(i) \equiv \mathcal{A}$
- $\pi(i)$: the action chosen in state i when policy π is pursued
- $p(i, a, j)$: the 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
- $t(i, a, j)$: the time spent in one transition from state i to j under action a

Also, note that: $\bar{r}(i, a) = \sum_{j \in \mathcal{S}} p(i, a, j) r(i, a, j)$ will denote the expected immediate reward in state i when action a is chosen in state i, while $\bar{t}(i, a) = \sum_{j \in \mathcal{S}} p(i, a, j) t(i, a, j)$ will denote the expected immediate transition time out of state i when action a is chosen in state i.

We now define the so-called *average reward* for a policy π in an SMDP. Let x_s denote the state of the system before the sth transition of the system, where it is important to note that in an infinite horizon problem, s will go from 1 to infinity. Then, the following scalar, in which $x_1 = i$, is called the average reward of the policy π if the system starts its transitions from state i:

$$\rho_i(\pi) = \lim_{k \to \infty} \frac{\mathsf{E}\left[\sum_{s=1}^{k} r(x_s, \pi(x_s), x_{s+1}) | x_1 = i\right]}{\mathsf{E}\left[\sum_{s=1}^{k} t(x_s, \pi(x_s), x_{s+1}) | x_1 = i\right]}. \tag{1}$$

It can be shown that when the policy π is *regular* [27], the average reward does *not* depend on the starting state i and can hence be denoted as $\rho(\pi)$, which essentially means that the average reward is the same regardless of which state the system starts at. In this paper, we will assume that all policies in our SMDP are regular. The goal in solving the SMDP is then to identify the policy that maximizes the average reward, i.e., identify a policy π^*, whose average reward equals ρ^*:

$$\text{Maximize}_{\pi} \quad \rho(\pi) \equiv \rho^*.$$

In an MDP, which, as we stated above is a special case of the SMDP, the transition time $t(., ., .) = 1$ for all instances of $t(., ., .)$. The average reward for the MDP can then be analogously obtained from Equation (1).

The average reward is necessary in the large-scale tests of our new algorithm. However, in order to demonstrate key properties of the algorithm, such as boundedness, we also used the so-called discounted-reward metric in tests on small-scale problems. The discounted reward metric for MDPs is defined as follows:

$$\Theta_i(\pi) = \lim_{k \to \infty} \mathsf{E}\left[\sum_{s=1}^{k} \lambda^{s-1} r(x_s, \pi(x_s), x_{s+1}) | x_1 = i\right], \tag{2}$$

where λ is the discount factor. The goal in this context is to identify a policy, π^*, that maximizes $\Theta_i(.)$ for every value of $i \in \mathcal{S}$.

A classical method for solving MDPs and SMDPs is dynamic programming (DP) [28]. DP seeks to solve these problems using the so-called transition probabilities (TPs) of the Markov chains underlying the system transitions; the TP is the probability of transitioning from one state to another under a given action and has been defined above in the notation. Because TPs are *required* in DP, the latter tends to break down when the number of state-action pairs exceeds a few thousands. This is because the TP model then becomes too large to store in the computers. To be more specific, a system with N states and M actions would yield a TP matrix (TPM) of size $N \times N$ for each of the M actions. As a result, when N is large, it is difficult to store and process all the elements of the TPMs, and then the so-called *curse of dimensionality* sets in. This is typical of large-scale problems in the real world. RL was invented to deal with the curse of dimensionality; RL avoids the TPs, but requires either a simulator of the real-world system or has to be implemented in the real-world system itself to work. RL thus allows us to bypass the construction of the TPMs, thereby avoiding the curses of dimensionality but still producing near-optimal solutions.

2.2. Airline Revenue Management

Deregulation in 1978 gave airlines in the U.S. the flexibility to determine their own schedules, routes, and fares, as long as they followed the guidelines formulated by the Federal Aviation Administration. The science of airline revenue management started gaining attention from then onwards. More recently with the progress of DP and simulation, the problem of airline revenue management has been studied via near-optimal or optimal techniques. The main decision to be made in the context of revenue management is to decide whether to accept or reject customers as they arrive, via a website [29], for the price they are willing to pay. Essentially, the airline offers different fares for differing privileges for a given origin-destination itinerary, and customers reserve tickets accordingly. But as time passes and as seats at different fares are sold, the airline must close down certain fares and open new fares—in order to maximize their profits. This is the central theme underlying the dynamics of the airline revenue management problem.

Durham [30] estimates that a reservation system may need to handle up to five thousand potential bookings per second, which should highlight the importance and scale of this problem. The customer in the main cabin of the economy class is generally offered a set of several different fares for a given origin-destination plan. Internally, for the airline, each fare is associated to a fare class. Different fare classes do not imply that the seats are located in different sections of the plane; typically, all seats are available to all fare classes within the cabin. Generally, the lower fare classes are among the first ones to be closed down by the airline. This is because, in general, the lower fare classes have the greatest demand and are hence sold first; however, it should be noted that the higher fare classes may offer advantages, and hence some passengers arriving early in the booking horizon may actually buy higher fares, even when lower fares are still available. Customers who choose to pay a higher fare, even when lower ones are available, generally, receive better benefits such as a lower cancellation penalty or the ability to board the flight sooner. In general though, customers arriving earlier in the booking horizon are more likely to buy a lower-priced ticket. Each airline typically updates its price offerings regularly based on the time remaining until departure, preferences of the customer, and many other factors. Prices have to be adjusted in a suitable manner in order for the continued success of an airline company.

Mathematically, the problem of setting prices is really one of determining the number of seats to be allocated to each fare class in a way that leads to the maximum profits. If too many seats are allocated to the lower fare classes, there would be few empty seats at the time of departure, but low profits would also result at the same time. On the other hand, if too many seats are allocated to the higher fare classes, one will end up with many empty seats at the time of departure, which will also lead to diminished profits. Thus, airlines seek a compromise between these two extreme scenarios to strike a balance.

Another aspect of this problem is the number of empty seats in the plane when it takes off. Airline seats are a perishable commodity, meaning that as soon as a flight departs, any empty seat signifies a loss of *potential* revenue. Needless to add, airline companies strive to reduce the probability of empty seats; of course, this is a world of cut-throat competition, and every opportunity to make revenues is seized upon by competitors, making it essential for every airline to ensure that it loses no opportunity to make revenues in a legal manner. These cancellations and no shows are accounted for by *overbooking* a flight. This implies that the airline company sells more seats than the total number of seats available on the plane. If the number of passengers who show up exceeds the capacity, the airline company must pay an overbooking penalty to the passengers who could not get a seat (a compensation fee) and also find a new flight for them.

Taken together, for any given origin-destination, the demand for different fares, the probability of cancellations, and overbooking make the airline revenue management problem a challenging one in which it is necessary to get the arithmetic right in terms of how many seats are sold at each fare, prior to flight departure. This requires data collection on the actual demand for each fare, the cancellation probabilities, the cancellation penalties, and the overbooking penalties. When this data is available, one can either use a heuristic, or if the problem has a small dimension, a more advanced DP technique. DP techniques break down on large-scale problems encountered in industry; though heuristics work on large-scale problems, they are always questionable in terms of how close to optimality their solutions are. In this paper, we will use the SMDP model underlying DP, but use an RL technique that can be employed on large-scale problems for solution purposes. RL can generate near-optimal solutions on MDPs/SMDPs when DP breaks down on them due to their large dimensionality, making it impossible to compute the TPs underlying the respective Markov chains. We will also show that the RL-based approach outperforms the heuristic approach in our numerical experiments.

We now enumerate two assumptions made in our model. 1) We assume a binary choice to the customer: either the customer accepts the offer of the current fare or rejects it. This assumption is justified by the nature of the data we use for our simulation model, where the arrival process is assumed to be Poisson and hence the arrival of each customer class is an independent Poisson process (see Chapter 2 of [31]). See also Talluri and van Ryzin [32] for a model in which this assumption is relaxed, but, their approach is not simulation-based; rather used in conjunction with the heuristic EMSR-b. 2) The input data for every leg in the network is known and can be used independently for each leg. The decomposition of the network problem into independent legs is possible via the Displacement Adjusted Virtual Revenue approach developed in the revenue management community (see [12]) for networks. The decomposition allows the airline to focus on the key legs that generate the most revenues and use the very tractable single-leg approach in each of the legs studied. A network model that considers multiple legs becomes too unwieldy and acquires a black-box nature that practicing managers are not attracted to. Also, from a technical standpoint, network problems become intractable for simulation-based settings and typically require either multi-stage stochastic programming [33] or a recently developed fluid model [34]. As such, the single-leg approach continues to be popular in the airline industry, as indicated to us by industry practitioners in the 2016 Annual Conference of the Institute of Operations Research and Management Science (INFORMS) during our presentation.

SMDP model. Before presenting the SMDP model, i.e., the state and action space, for the revenue management problem, we present additional notation that is required:

- s_i: the number of seats sold in the ith fare class
- n: the number of fare classes
- c: class of the current customer
- t: the time remaining for the departure of the plane
- Λ: Poisson rate of arrival of all customers

The action space for this problem contains two actions, which are *(Accept,Reject)*, and the state space is as follows: $(c, t, s_1, s_2, \ldots, s_n, \psi_1, \psi_2, \ldots, \psi_n)$, where ψ_i is a vector of size s_i that contains the times of arrival (in the booking horizon) of the passengers in the ith fare class. Clearly, the state space here is too large. Even if the continuous component, t, is ignored, the size of the state space equals several billion. Naturally, developing TPs for this model is also ruled out, thus making this a suitable case study for an RL algorithm.

Even for RL, the state space cannot be used as is. After significant experimentation for the airline case study (the details of which can be found in Section 4.2), it was found that keeping t and the vectors ψ did not improve the results from the RL algorithm, and hence t and ψ were dropped from the state space. Such approximation of state-space is common in the RL community

[2]. The following function from [35] was used to reduce its dimensionality to a manageable number:

$$\phi = \frac{\sum_{i=1}^{n}(f_i \times s_i)}{\theta},$$

where f_i is the fare for the ith class and θ is a hand-coded, but user-defined, scaling value used in the encoding needed to produce an aggregation of the state space. The value of θ must be determined through experimentation (trial and error) for each case of input parameters, and its value will hence be case dependent. We further note that ϕ is often referred to as a feature in the literature [1]. The equation above actually produces a continuous state space, which can be problematic; but by rounding the value of the right-hand side of the equation down to the nearest integer, we have a discrete integer value for ϕ and thus a suitable feature space that can be used in our experimentation. As a result of the above transformation, the altered state (feature) space can now be defined in discrete terms as (c, ϕ).

EMSR-b. A widely used heuristic in the airline industry for solving the single-leg version of the problem is called EMSR-b [12]. We will use this heuristic to benchmark the computational results from using our RL algorithm on the airline case study. We now present details of how the heuristic works.

As noted above, we will use f_i to denote the fare in dollars for the ith class, where $f_1 < f_2 < f_3 < \cdots < f_n$, and Y_i to denote the demand (i.e., projected number of customers) in the ith class. The heuristic first computes two quantities based on the fares and the projected demands: (i) the so-called aggregate demand, $\hat{Y}_i$, for the ith fare class and (ii) the so-called aggregate revenue, $\overline{f}_i$, for the ith fare class. For $i = 1, 2, \ldots, n$,

$$\hat{Y}_i = \sum_{j=i}^{n} Y_j.$$

Thus, $\hat{Y}_i$ denotes the sum of the demands for the ith fare class and that for all classes with fares exceeding f_i. Also, for $i = 1, 2, \ldots, n$,

$$\overline{f}_i = \frac{\sum_{j=i}^{n} f_j \mathsf{E}[Y_j]}{\sum_{j=i}^{n} \mathsf{E}[Y_j]}.$$

Next, the heuristic solves the following equation, also called Littlewood's equation [36]: For $i = 1, 2, \ldots, n-1$,

$$f_i = \overline{f}_{i+1} \mathsf{Pr}[\hat{Y}_{i+1} > P_{i+1}],$$

where P_{i+1} is the so-called protection level for the ith class and is one of the $(n-1)$ unknown variables, whose value needs to be determined from solving the equation above; the protection level is the number of seats to be protected for a given class from the lower fare classes. Thus, P_i is the number of seats to be protected for class $(i-1)$ from classes $i, i+1, \ldots, n$. There is no protection level for class 1, as it is the lowest fare class from which no protection is needed. For solving the Littlewood's equation, one needs the distribution of each of the random variables, $\overline{Y}_i$, for all values of i; these distributions can be determined from the input data available to airlines, and oftentimes, the underlying distribution is normal, which can be approximated by the Poisson distribution, which allows us to use the exponential distribution for the time between successive arrivals.

Finally, in the last step of the heuristic, the booking limit for the ith class is calculated as follows: $BL_n = C$ and for $i = 1, 2, \ldots, n-1$,

$$BL_i = \max\{C - P_{i+1}, 0\},$$

where C denotes the capacity of the plane. The tangible meaning of the booking limit in the airline reservation system is that if there are BL_i seats booked in class i already, no further customers are allowed in that class. If overbooking is considered, one heuristically replaces C in the above by $C/(1 + cp)$, where cp denotes the *average* cancellation probability over all fare classes, in order to accommodate for an artificially increased capacity only during the booking process.

3. New Algorithm

In this section, we first present the background theory of actor critics, along with mathematical reasons for difficulties encountered with the algorithm in the literature, and finally propose a new algorithm: first a discounted-reward version and then the average-reward version; the latter is suitable for the airline case study. This section is organized as follows. Subsection 3.1 discusses the background material focussing on the classical actor critic. Subsection 3.2 presents the discounted-reward version on the MDP model. Subsection 3.3 is devoted to presenting a step-by-step description of the new algorithm on the average-reward SMDP.

3.1. Classical Actor Critics

The key underlying problem in the RL setting is to discover the optimal action in each state. The so-called policy is a collection of actions for each state. The optimal policy is hence one that delivers the best value for the performance metric. In the actor-critic setting, an actor is the agent that selects a policy and a critic is the other agent that computes the so-called value function of dynamic programming [28] for each policy. As a result of its interactions with the environment, both the actor and the critic update their iterates on the basis of feedback produced by the environment. We now provide mathematical notation needed for the actor-critic algorithm.

- $P(i, a)$: The value of the actor's iterate associated to state i and action a
- $V(i)$: The value of the critic's iterate for state i; equivalently the current value function of state i

- $q(i,a)$: The probability with which the algorithm selects action a in state i
- η: A tunable contraction factor, set in the range $(0,1)$, close to 1
- λ: The discount factor in discounted-reward MDPs
- α: The learning rate or step-size for the actor
- β: The learning rate or step-size for the critic
- γ: The learning rate or step-size for the average reward

Steps in Projection-Bounded Actor-Critic Algorithm for Discounted-Reward MDPs:

The main steps in the discounted-reward traditional actor-critic MDP algorithm that uses the projection to bound its actor's values are as follows.

- Inputs: Initialize all actor, $P(.,.)$, and critic, $V(.)$, values to zero. Let $\overline{P}$ be a large positive number, such that $e^{\overline{P}}$ can be stored in the computer without overflow. Set k, the number of iterations, to 0. Let $k_{\max}$ denote the maximum number of iterations for which the algorithm is run.
- Loop until $k = k_{\max}$
  - Let i be the current state. Select action a with probability of
    $$q(i,a) = \frac{e^{P(i,a)}}{\sum_{b \in \mathcal{A}(i)} e^{P(i,b)}}.$$
    The above is called Boltzmann action selection. Simulate action a, and let the next state be j. Let $r(i,a,j)$ be the immediate reward in the state transition.
  - Actor's update:
    $$P(i,a) \leftarrow P(i,a) + \alpha \left[r(i,a,j) + \lambda V(j) - V(i) \right]. \quad (3)$$
  - Projection: If $P(i,a) > \overline{P}$, set $P(i,a) = \overline{P}$. If $P(i,a) < -\overline{P}$, set $P(i,a) = -\overline{P}$.
  - Critic's update:
    $$V(i) \leftarrow (1-\beta)V(i) + \beta \left[r(i,a,j) + \lambda V(j) \right]. \quad (4)$$
  - 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: $\arg\max_{b \in \mathcal{A}(i)} P(i,b)$.

As discussed above, the artificial bound does not allow proper exploration. One heuristic way to use a large value for $\overline{P}$ and still compute the exponential term is to use a so-called temperature, U, where $U \in (0, 1)$, in the Boltzmann action-selection, modifying it to the following:

$$q(i, a) = \frac{e^{P(i,a)\times U}}{\sum_{b\in\mathcal{A}(i)} e^{P(i,b)\times U}}.$$

As a result of the above, since U is positive but much smaller than 1, the product $P(i, a) \times U$ becomes small, even if $P(, .,)$ is large, thereby allowing us to compute $e^{P(i,a)\times U}$. Unfortunately, there are three difficulties associated with this: (i) technically the convergence proof requires that U is equal to 1, (ii) this heuristic approach still limits the exploration, and (iii) using the temperature does not resolve the problem of computer overflow with the value of the actor itself, i.e., when $P(, .,)$ itself becomes too large to be stored in the computer.

3.2. Bounded Actor-Critic for Discounted-Reward MDPs

The algorithm in Gosavi [10] that we now present in brief seeks to alleviate the above-mentioned difficulties; the aim is to produce boundedness in the actor's iterates without any projection—by using a convex combination in its update. The algorithm's steps would be the same as shown for the projection-bounded algorithm with the following exceptions. Of course, the projection step would be eliminated and the update in Equation (3) would be replaced by:

$$P(i, a) \leftarrow (1 - \alpha)P(i, a) + \alpha \left[r(i, a, j) + \lambda V(j)\right]. \tag{5}$$

Note that a key difference between the update above and that in the projection-bounded algorithm is that we multiply the first $P(i, a)$ term on the right hand side by $(1 - \alpha)$, which makes the update a convex combination. Of course, one still needs to prove mathematically that the iterates will remain bounded with the update defined in (5); see [10] for a mathematical proof. The other difference with the update in Equation (3) is that the term within the square brackets in the right-hand side of Equation (5) does not contain the subtracted term $V(i)$.

As discussed in the introductory section, despite the mathematical bound, unfortunately, the values of the actor using the update in Equation (5) still become quite large in magnitude in practice, which poses problems for the exploration. We will demonstrate this issue numerically in Section 4.

We now propose a *different* refinement of the projection-bounded algorithm in which we do *not* erase the subtracted term $V(i)$ from the original algorithm, while simultaneously using the notion of convex combination. This algorithm and its extension to the average-reward SMDP, discussed in the next subsection, are the main contributions of this paper. The main update for the actor in the new algorithm for the discounted-reward MDP, which will henceforth be referred to as **Algorithm 1**, would be:

$$P(i, a) \leftarrow (1 - \alpha)P(i, a) + \alpha \left[r(i, a, j) + \lambda V(j) - V(i)\right]. \tag{6}$$

This algorithm should follow all the steps shown for the projection-bounded algorithm with two exceptions: (i) the projection step should be skipped and (ii) Equation (6) should replace Equation (3). Our new algorithm is shown to have the following nice properties: (i) boundedness, (ii) the actor's iterates, i.e., the $P(.,.)$ terms, end up with values that have a *small* magnitude, and (iii) as a result of the previous behaviours, the algorithm can explore *fully.* We will prove mathematically the first property, i.e., the boundedness of the actor's and critic's iterates, in the Appendix (A1). The other two properties will be demonstrated in the section on numerical results.

3.3. Bounded Actor-Critic for Average-Reward SMDPs

For the average reward SMDP, the algorithm needs an additional update for computing the average reward. Further, the Bellman equation is different, and we present the main result associated to it.

Theorem 1. *For an average-reward SMDP in which all Markov chains are regular, there exists a vector* $V \equiv \{V(1), V(2), \ldots, V(|\mathcal{S}|)\}$ *and a scalar* ρ *that solve the following system of equations:*

$$V(i) = \max_{a \in \mathcal{A}(i)} \left[\bar{r}(i,a) - \rho \bar{t}(i,a) + \sum_{j=1}^{|\mathcal{S}|} p(i,a,j) V(j) \right] \text{ for all } i \in \mathcal{S}. \tag{7}$$

Further ρ *equals* ρ^**, the optimal average reward of the SMDP.*

Equation (7) is the so-called Bellman optimality equation for SMDPs. The above result leads us to the optimal solution of the average reward SMDP, since it implies that if one can find a solution to the vector V and the scalar ρ^*, then the following policy d is optimal, where

$$d(i) \in \arg\max_{a \in \mathcal{A}(i)} \left[\bar{r}(i,a) - \rho^* \bar{t}(i,a) + \sum_{j=1}^{|\mathcal{S}|} p(i,a,j) V(j) \right] \text{ for all } i \in \mathcal{S}.$$

In order to use the equation in our RL framework, we will need a slight modification of the above equation, which is as follows:

$$V(i) = \max_{a \in \mathcal{A}(i)} \left[\bar{r}(i,a) - \rho^* \bar{t}(i,a) + \eta \sum_{j=1}^{|\mathcal{S}|} p(i,a,j) V(j) \right], \tag{8}$$

where $\eta \in (0,1)$ is a constant. The uniqueness of the solution of the above equation follows directly from the theory of discounted reward MDPs [28]. The use of η, it has been observed empirically, makes actor-critic algorithms behave better in practice [8], i.e., makes it easier to approach the optimal solution, and just as importantly also ensures that the values of the actor remain bounded; (we will present a mathematical proof of boundedness in the Appendix (A2).) Our algorithm will hence use the above equation, Equation (8), as its foundation.

As η tends to 1, the above equation tends to the Bellman equation for SMDPs, i.e., Equation (7). In practice, if the value of η is set close to 1, Equation (8) behaves just like (resembles) the Bellman optimality equation for SMDPs. Forcing a unique solution for the average reward Bellman equation is an idea that we have borrowed from the literature: this idea has been used in the context of policy gradients [14] to obtain superior algorithmic behavior. Further, past work in RL for average reward SMDPs has also used this concept [37, 8]. Because of the use of η in Equation (8), the equation is mathematically identical to the Bellman optimality equation of a discounted reward MDP, which is known to be a contraction map and to consequently carry a unique solution (see Prop. 1.4.1 in Vol II of [28]). This related discounted reward MDP would have an average immediate reward function defined as $\overline{w}(i,a) = \bar{r}(i,a) - \rho^* \bar{t}(i,a)$ for all (i,a) and a discount factor λ that equals η.

The main steps in the new algorithm, which will henceforth be called **Algorithm 2**, are as follows.

Steps in the New Actor-Critic Algorithm for Average-Reward SMDPs:

- Inputs: Initialize all actor, $P(.,.)$, and critic, $V(.)$, values to zero. Set k, the number of iterations, to 0. Set the scalars, R, T, and ρ, to zero. Set η to a positive value very close to 1 but strictly less than 1. Let $k_{\max}$ denote the maximum number of iterations for which the algorithm is run.
- Loop until $k = k_{\max}$
  - Let i be the current state. Select action a with probability of
    $$q(i,a) = \frac{e^{P(i,a)}}{\sum_{b \in \mathcal{A}(i)} e^{P(i,b)}}. \tag{9}$$
    Let j be the next state. Let $r(i,a,j)$ denote the immediate reward in the state transition and $t(i,a,j)$ denote the time taken in the transition.
  - Actor's update:
    $$P(i,a) \leftarrow (1-\alpha)P(i,a) + \alpha\left[r(i,a,j) - \rho t(i,a,j) + \eta V(j) - V(i)\right]. \tag{10}$$
  - Critic's update:
    $$V(i) \leftarrow (1-\beta)V(i) + \beta\left[r(i,a,j) - \rho t(i,a,j) + \eta V(j)\right]. \tag{11}$$
  - Average Reward update: Update R, T, and ρ as follows:
    $$R \leftarrow R + r(i,a,j); \quad T \leftarrow T + t(i,a,j); \quad \rho = (1-\gamma)\rho + \gamma[R/T]. \tag{12}$$
  - 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: $\arg\max_{b\in\mathcal{A}(i)} P(i,b)$.

We note that the Boltzmann action-selection scheme employed above and shown via Equation (9) does not require the temperature, U, and, as will be shown later in the next section, works effectively in the algorithm; in other words, no temperature reduction is needed, nor is any artificial bounding required in our algorithm.

4. Numerical Results

This section is divided into two subsections. The first subsection is devoted to results on small MDPs to demonstrate important properties of our new algorithm, while the second provides results via the airline case study.

4.1. Small MDPs

Algorithm 1, i.e., the new bounded actor-critic algorithm whose actor update is defined in Equation (6), was run for 4 different discounted reward MDPs consisting of two states each and two actions allowed in each state. Cases have been taken from [10]. The data for each case is as follows, where TPM_a denotes the TPM for action a and TRM_a denotes the TRM for action a. Note that the element in the ith row and jth column of TPM_a equals $p(i,a,j)$. Similarly, the element in the ith row and jth column of TRM_a equals $r(i,a,j)$.
Case 1:

$$\mathbf{TPM}_1 = \begin{bmatrix} 0.7 & 0.3 \\ 0.4 & 0.6 \end{bmatrix}; \mathbf{TPM}_2 = \begin{bmatrix} 0.9 & 0.1 \\ 0.2 & 0.8 \end{bmatrix};$$

$$\mathbf{TRM}_1 = \begin{bmatrix} 6 & -5 \\ 7 & 12 \end{bmatrix}; \mathbf{TRM}_2 = \begin{bmatrix} 10 & 17 \\ -14 & 13 \end{bmatrix}.$$

For the remaining cases, only those inputs where the problem differs from Case 1 are listed. **Case 2:** $r(1,1,2) = 5$; $r(2,2,1) = 14$; **Case 3:** $r(1,2,1) = 12$; **Case 4:** $r(1,1,1) = 16$. Also, $\lambda = 0.8$ for all cases.

The algorithm was run for a maximum of 10,000 iterations with the following learning rates: $\alpha = (log(k+1))/(k+1)$, $\beta = 150/(300+k)$. The optimal policy for each case was obtained using Q-value iteration [3] and is denoted as $< a_1, a_2 >$, where a_1 denotes the optimal action in state 1 and a_2 denotes the optimal action in state 2. Table 1 shows the optimal policy, d^*, and the optimal value function, $V^*(.)$—both obtained from value iteration. The table also shows the value function, $V(.)$, obtained from the actor-critic algorithm for discounted reward MDPs, where the actor-update defined in Equation (6) is employed. It can easily be seen that the value functions produced by the actor-critic are very close to the optimal values produced from value iteration.

Table 2 shows the actor's values from the actor-critic with the update in Equation (6), as well as the policy produced by the algorithm, d; this policy can be derived by examining the actor's values and finding the action that produces the largest values for each state. For example, in Case 1, $P(1,1) = -8.389$ and

$P(1,2) = -0.004$ implying that action 2 is better in state 1 because $P(1,2) > P(1,1)$. Similarly, in state 2, $P(2,1) = -0.020 > P(2,2) = -3.497$ meaning that action 1 is better in state 2, which produces a policy of $< 2,1 >$; this matches the optimal policy produced from value iteration (See Table 1). Thus, it is clear from the table that the actor-critic algorithm produces the optimal policy in all 4 cases. Very importantly, the actor's values, $P(i,a)$, are of a significantly *small* magnitude in all cases.

In contrast, to the small magnitude actor's values produced above, see Table 3 for results from using on the same cases where the actor's update followed Equation (5), which is from [10]; the table also shows the policy generated by the algorithm, which matches with the optimal one in each case. What is more interesting is that the actor's values in Table 3 have significantly *larger absolute values* than those in Table 2, which are from the proposed new algorithm; the *maximum* absolute value for the actor in Table 2 is 10.73, while *minimum* absolute value in Table 3 for the actor is 38.17. It is interesting to note that though this algorithm, based on Equation (5), also generated the optimal policy, our simulations showed that a large magnitude of the actor's values did not permit the thorough exploration of the state space that was observed with the previous algorithm (Algorithm 1).

Table 1: Value function obtained from actor critic and from DP, as well as the optimal policy

Case	d^*	$V(1)$	$V(2)$	$V^*(1)$	$V^*(2)$
1	$< 2,1 >$	52.93	51.67	53.03	51.86
2	$< 2,2 >$	55.38	61.16	55.77	61.45
3	$< 2,1 >$	60.80	56.59	60.83	56.66
4	$< 1,1 >$	49.90	49.35	48.97	49.36

Table 2: The actor's values using actor update in Equation (6) and the resulting policy

Case	d	$P(1,1)$	$P(1,2)$	$P(2,1)$	$P(2,2)$
1	$< 2,1 >$	−8.389	−0.004	−0.020	−3.497
2	$< 2,2, >$	−3.887	−0.029	−2.045	−0.036
3	$< 2,1 >$	−10.730	−0.001	−0.001	−3.552
4	$< 1,1 >$	0.070	−7.744	0.022	−1.605

4.2. Airline Revenue Management

We now present numerical results from an elaborate experimentation on a large-scale airline system, using the average reward SMDP actor-critic algorithm, i.e., Algorithm 2, proposed in this paper. Much of this data employed here is from an airline industry, but it has been masked and slightly modified without changing the basic structure to avoid identification.

Table 3: Actor's values from the actor update in Equation (5) and the resulting policy

Case	d	$P(1,1)$	$P(1,2)$	$P(2,1)$	$P(2,2)$
1	$< 2,1 >$	43.95	53.00	52.07	39.48
2	$< 2,2 >$	50.11	55.81	48.40	61.71
3	$< 2,1 >$	49.32	60.81	56.70	43.08
4	$< 1,1 >$	48.87	38.17	49.03	38.75

Input Parameters: The fare structure for each case is given by

$$FS = (f_1, f_2, f_3, \dots, f_n, b),$$

where f_i is the fare of the ith fare class and b is the bumping cost. As stated before, a lower value of i stands for a lower revenue fare class. Two sets of systems (cases) were created for the experimentation: systems with four fare classes and systems with six fare classes. Part of this dataset was obtained from a real airline company, where it was made available at the 2017 INFORMS conference, but the dataset was masked to hide the identity of the airline, i.e., some numbers were modified without altering the basic structure of the dataset. In every case, the booking horizon was assumed to be 100 days long, and, for the arrivals, a homogeneous Poisson process with a rate of $\Lambda = 1.4$ passengers per day was used; the plane was assumed to have a total capacity of 100 seats. The Poisson process for each fare class will hence be an independent process, whose rate should be equal to $\Lambda Pr(i)$, where $Pr(i)$ denotes the probability that the arrival belongs to the ith class. The so-called cancellation probability for each fare class is essentially the probability with which a traveler in that given fare class cancels the ticket. When a cancelation occurs, it is scheduled using a uniform distribution between the time of arrival and the time of flight departure. Tables 4 and 5 provide much of the data for input parameters needed in our experimentation. Finally, the tuning parameters of the algorithm were determined as follows. The value of θ in the algorithm update had to be determined *separately* for each individual case, based on careful experimentation, to produce the best possible policy, and these values are presented in the table that shows the outputs from our experimentation. A value of 0.999999 was used for η in the actor-critic algorithm, which was also determined after suitable experimentation. After significant experimentation, the following step-sizes were found to be most suitable for the three updates in the algorithm:

$$\alpha = \frac{15000}{300000 + k}; \quad \beta = \frac{10000}{300000 + 3k}; \quad \gamma = \frac{10000}{300000 + 10k}.$$

Experimentation and Algorithm Performance: The performance for both the actor-critic algorithm and EMSR-b was measured in terms of the average reward, ρ, whose unit is dollars per day. The algorithm was tested on ten cases for each of the four-fare systems and for each of the six-fare systems. Booking limits were first computed from the EMSR-b heuristic, via a MATLAB

Table 4: Input parameters for the 4- and 6-fare systems

Case	Fares (4-Fare System)	Fares (6-Fare System)
1	75, 200, 400, 550	101, 127, 153, 179, 293, 419
2	80, 200, 400, 500	94, 112, 142, 160, 271, 395
3	75, 150, 300, 550	111, 131, 153, 185, 293, 426
4	80, 150, 400, 550	127, 143, 167, 199, 320, 462
5	70, 150, 350, 550	105, 135, 143, 179, 284, 411
6	125, 180, 225, 400	90, 105, 139, 156, 261, 388
7	100, 175, 250, 400	108, 127, 155, 191, 295, 431
8	100, 150, 200, 450	76, 98, 123, 162, 247, 400
9	119, 139, 239, 430	87, 115, 162, 185, 278, 410
10	145, 209, 280, 350	115, 134, 165, 184, 302, 430

Table 5: Other input parameters: Canc. denotes cancellation and Pen. denotes penalty

Parameter	4-Fare systems	6-Fare Systems
Arrival Probabilities	0.6, 0.25, 0.09, 0.06	0.3, 0.3, 0.13, 0.13, 0.09 0.06
Canc. Probabilities	0.1, 0.2, 0.2, 0.4	0.1, 0.1, 0.1, 0.2, 0.2, 0.4
Canc. Pen. (Cases 1:5)	70, 50, 30, 10	70, 50, 50, 30, 10, 0
Canc. Pen. (Cases 6:10)	100, 90, 60, 40	70, 50, 50, 30, 10, 0
Bumping Pen.	200	250

program. These limits were then used within the system simulator, again with 8 replications for each case, to evaluate the performance of the EMSR-b heuristic; the average value of average reward from these replications was denoted by ρ_{EMSR-b}. Tables 6 and 7 provide the EMSR-b booking limits returned for the 4-fare and 6-fare systems, respectively, where $BL(i)$ represents the booking limit of the ith fare class. Results of using the actor-critic algorithm, as well as EMSR-b, for the 4-fare systems and 6-fare systems, are shown in Tables 8 and 9 respectively. The learning phase of the actor-critic algorithm was run for approximately 1000 flights and took at most 130 seconds on a 64-bit, 2.5 GHz windows operating system in MATLAB. The learning phase helped determine the policy generated by the algorithm. Then the simulator was re-run with the fixed policy (also called frozen policy) for 8 replications with 200 flights per replication; the resulting average reward was shown as $\rho_{Actor-Critic}$ in Tables 8 and 9. Numerical improvement of the actor-critic algorithm over EMSR-b was defined as:

$$IMP = \frac{\rho_{Actor-Critic} - \rho_{EMSR-b}}{\rho_{EMSR-b}} \times 100\%.$$

This improvement was also shown in Tables 8 and 9. As can be seen from the tables, the actor-critic algorithm outperforms EMSR-b; a t-test was performed to determine if the results delivered from the actor critic differ from those of EMSR-b with 95% confidence in a statistical sense, and, in every case, a statistical difference was shown to exist. The improvement has ranged from 1.35% to 4.36%. It is to be noted that EMSR-b is widely used in the industry, where even a 1% improvement can lead to increased profits of millions of dollars in a single year. Figures 1:4 show the plots and the nature of the learning that occurs in some sample cases; each "iteration" shown on the x-axis of these figures actually equals 1000 iterations of the *algorithm.* Note that these figures display the so-called learning curves of reinforcement learning. Each learning curve can be unique, where the algorithm learns with trial and error. It is not uncommon for the algorithm to learn a policy that produces high rewards in the short run and yet dip to a lower reward after some time, but recover later to a better policy; Figure 2, which is for Case 7 of the 4-fare systems, shows such behavior. However, in the other three cases (see Figures 1, 3, and 4), the algorithm shows gradually improving or stable behavior in the limit.

5. Conclusion

While the actor-critic algorithm predates the more popular Q-Learning algorithm, one drawback of the actor critic that has perhaps prevented its applications in large-scale problems is the unboundedness of the actor's values. There are two significant difficulties associated to the unboundedness: (i) the values can become too large in magnitude causing a computer overflow and (ii) the large values usually cause insufficient exploration of the state space when used in conjunction with the popular Boltzmann action-selection scheme. Two mechanisms suggested in the literature to circumvent these difficulties are: (i) an

Table 6: EMSE-b booking limits for the 4-fare systems.

Case	BL(1)	BL(2)	BL(3)	BL(4)
1	68	107	122	129
2	69	108	123	129
3	68	107	122	129
4	69	106	122	129
5	69	106	122	129
6	74	109	121	129
7	72	109	122	129
8	73	108	120	129
9	75	107	121	129
10	75	110	123	129

Table 7: EMSR-b booking limits for the 6-fare systems.

Case	BL(1)	BL(2)	BL(3)	BL(4)	BL(5)	BL(6)
1	25	67	86	103	116	122
2	26	67	86	103	116	122
3	26	68	86	103	116	122
4	27	68	86	103	116	122
5	26	68	85	103	116	122
6	26	66	86	103	116	122
7	26	67	86	103	116	122
8	24	66	85	103	116	122
9	24	66	86	103	116	122
10	27	67	86	103	117	122

Table 8: Outputs with the actor critic and EMSR-b on the 4-fare systems.

Case	ρ_{EMSR-b}	$\rho_{Actor-Critic}$	θ	IMP (%)
1	163.79 ± 0.515	168.01 ± 1.454	1400	2.58
2	163.53 ± 0.365	167.52 ± 0.895	1200	2.44
3	138.56 ± 0.241	141.81 ± 1.195	1500	2.35
4	152.06 ± 0.417	157.83 ± 0.958	1900	3.8
5	140.24 ± 0.350	146.33 ± 0.438	1500	4.34
6	170.18 ± 0.403	173.81 ± 0.530	1800	2.13
7	154.68 ± 0.427	161.42 ± 0.489	1500	4.36
8	144.55 ± 0.651	149.20 ± 0.447	1000	3.22
9	157.01 ± 0.446	162.24 ± 0.253	1100	3.34
10	195.25 ± 0.421	199.16 ± 0.365	1700	2

Table 9: Outputs with the actor critic and EMSR-b on the 6-fare systems.

Case	ρ_{EMSR-b}	$\rho_{Actor-Critic}$	θ	IMP (%)
1	156.86 ± 0.260	160.28 ± 0.716	1200	2.18
2	141.16 ± 0.362	144.03 ± 0.692	1800	2.03
3	161.08 ± 0.360	164.68 ± 0.751	1600	2.23
4	177.89 ± 0.324	181.01 ± 0.665	1200	1.75
5	157.09 ± 0.231	161.06 ± 0.335	1000	2.53
6	135.55 ± 0.350	140.50 ± 0.514	1400	3.65
7	160.84 ± 0.295	163.01 ± 0.412	1600	1.35
8	128.02 ± 0.438	130.63 ± 0.395	1300	2.04
9	150.41 ± 0.426	152.84 ± 0.832	1800	1.62
10	166.01 ± 0.394	170.11 ± 0.585	1200	2.47

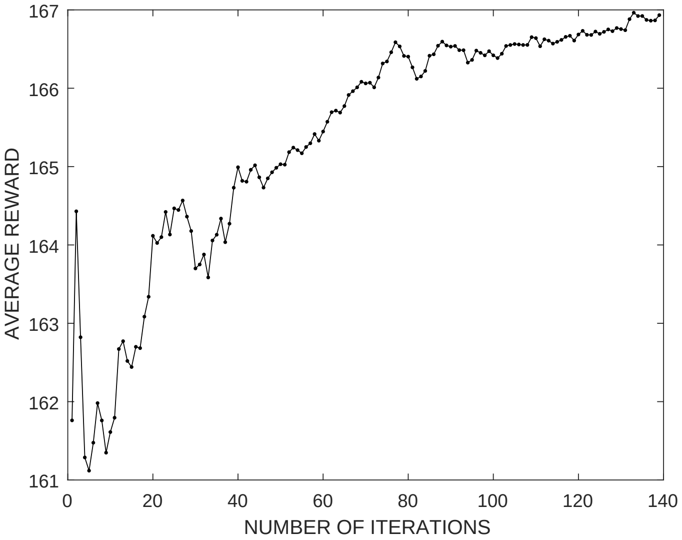

Figure 1: The figure shows the run-time behavior of the algorithm for Case 1 of 4-fare systems.

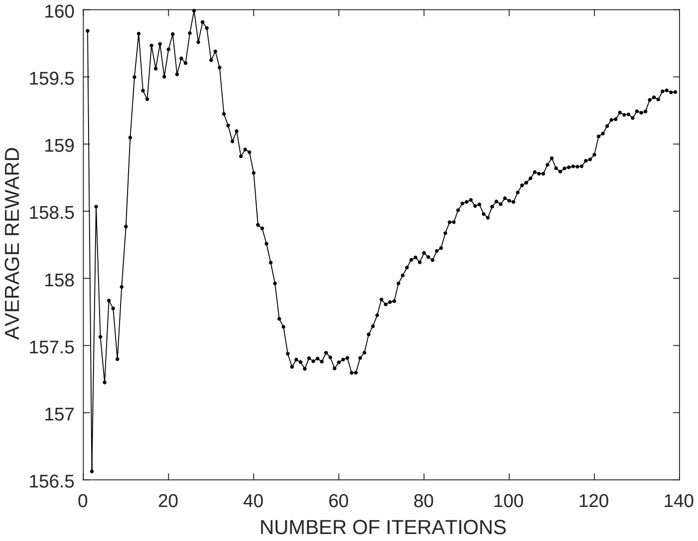

Figure 2: The figure shows the run-time behavior of the algorithm for Case 7 of 4-fare systems.

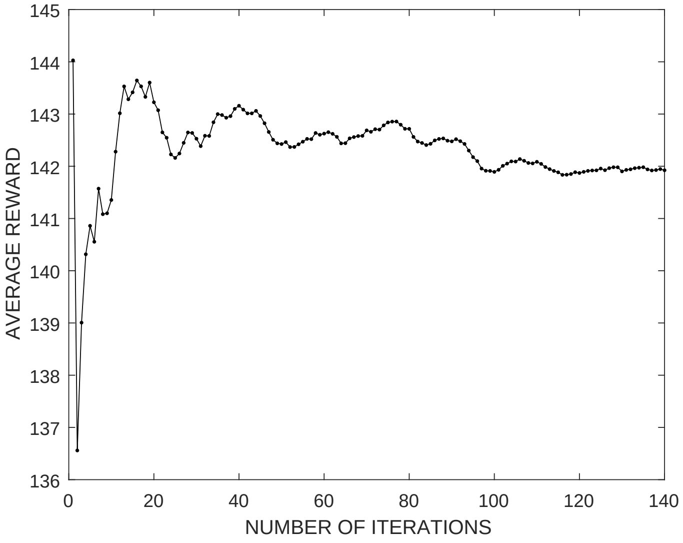

Figure 3: The figure shows the run-time behavior of the algorithm for Case 2 of 6-fare systems.

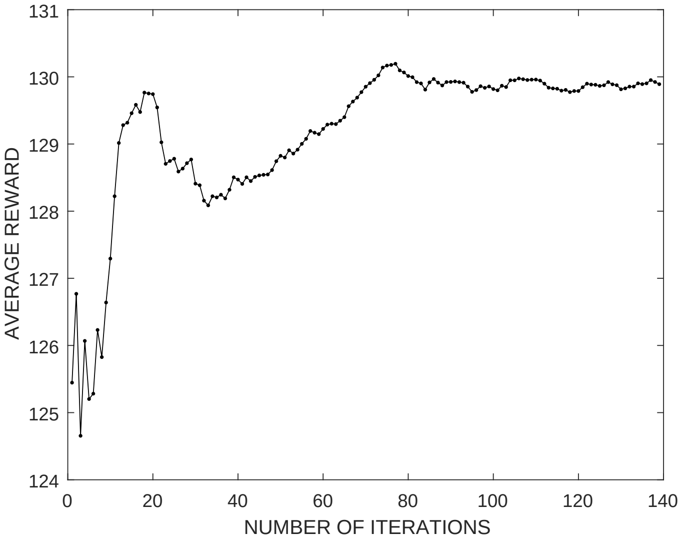

Figure 4: The figure shows the run-time behavior of the algorithm for Case 8 of 6-fare systems.

awkward projection that forces the values to be bounded and (ii) a temperature-reduction scheme; unfortunately, both of these mechanisms can still lead to poor solutions due to the insufficient exploration that they deliver. A key contribution of this paper was to develop a new update in which the actor's iterates were not only bounded, but also remained small in magnitude *without any artificial projection or temperature reduction*; further it should be noted that this led to a superior exploration of the state space.

We developed two algorithms for two different performance metrics: one for the discounted reward MDP and the second for the average reward SMDP. Both performance metrics are of interest in industry; the first is used widely in computer science, while the second is more popular in management science. Numerical tests were performed with both algorithms: the discounted reward MDP algorithm was tested on small instances, where the optimal policy was known, while the average reward SMDP was tested on a large-scale test-bed from the domain of airline revenue management with industrial data. In both types of tests, the algorithm showed encouraging empirical behavior, generating the optimal solution on the small MDPs in the discounted reward case and outperforming a well-known industrial heuristic in the large-scale tests with the average reward SMDPs. We also proved boundedness of the iterates *mathematically* for both algorithms, but a full-blown convergence analysis of the algorithms is beyond the scope of this paper. In future work, we will pursue such a convergence analysis for both algorithms developed here and develop an extension of the first algorithm to discounted reward SMDPs. We also seek to develop model-based reinforcement learning actor-critics (see [9] for an earlier attempt), which have the potential to be more robust than their model-free counterparts [38]. An additional future direction would be to test these algorithms on conjunction with deep learning architectures that are currently gaining significant interest in the field of artificial intelligence.

ACKNOWLEDGEMENTS: The authors would like to gratefully acknowledge the Intelligent Systems Cluster at Missouri University of Science and Technology for partially funding this research. We also wish to thank the anonymous reviewers of this article and the industry practitioners present at our presentation in the Annual Conference of Institute of Operations Research and Management Science (INFORMS) at Nashville in 2016.

APPENDIX

A1. Algorithm for Discounted Reward MDPs (Algorithm 1): we now analyze the boundedness of the iterates in Algorithm 1. The main idea underlying the proof follows from [10], but since the algorithm here is actually different, we present the complete proof. We first show that the critic's values remain bounded. The boundedness of the actor's values can then be shown from the boundedness of the critic's value.

Let V^k denote the vector of values computed by the critic and P^k denote the same by the actor in the kth iteration. Technically, $P(.,.)$ is a matrix, but we can map it into a one-dimensional vector.

Theorem 2. *The sequence $\{V^k, P^k\}_{k=1}^{\infty}$ in Algorithm 1 remains bounded.*

Proof The proof will be presented through two lemmas. The first lemma will prove that the vector of iterates V^k remain bounded, and it will be needed to prove the second lemma, which will show that the vector of iterates P^k will also remain bounded.

Lemma 1. *The sequence $\{V^k\}_{k=1}^{\infty}$ in Algorithm 1 remains bounded.*

Proof We claim that for every state i:

$$|V^k(i)| \leq M(1 + \lambda + \lambda^2 + \cdots + \lambda^k), \tag{13}$$

in which M is a positive finite number defined as follows:

$$M = \max\left\{ r_{\max}, \max_{i \in \mathcal{S}} |V^1(i)| \right\}, \tag{14}$$

$$\text{where } r_{\max} = \max_{i,j \in \mathcal{S}, a \in \mathcal{A}(i)} |r(i,a,j)|. \tag{15}$$

Since $r(.,.,)$ is bounded, $r_{\max}$ must be bounded. Since we start with finite values for V, then M too must be bounded. Then, from the above claim (13), boundedness follows since if $k \to \infty$,

$$\limsup_{k\to\infty} |V^k(i)| \leq M\frac{1}{1-\lambda}$$

for all $i \in \mathcal{S}$, since $0 \leq \lambda < 1$. We now prove our claim in (13) via induction.

The V-value of only one state is updated in a given iteration in asynchronous updating, while the other V-values remain un-updated. Hence, in the kth iteration of the asynchronous algorithm, the update for $V^k(i)$ is either according to Case 1 or Case 2.

Case 1: The state is updated in the kth iteration: $V^{k+1}(i) = (1-\beta)V^k(i) + \beta\left[r(i,a,j) + \lambda V^k(j)\right]$.

Case 2: The state is not updated in the kth iteration: $V^{k+1}(i) = V^k(i)$.

Now, if the update is carried out as in Case 1:

$$\begin{aligned}|V^2(i)| &\leq (1-\beta)|V^1(i)| + \beta|r(i,a,j) + \lambda V^1(j)| \\ &\leq (1-\beta)M + \beta M + \beta\lambda M \text{ (from (15) and (14))} \\ &\leq (1-\beta)M + \beta M + \lambda M = M(1+\lambda) \text{ (since } \beta \leq 1)\end{aligned}$$

Now, if the update is carried out as in Case 2: $|V^2(i)| = |V^1(i)| \leq M \leq M(1+\lambda)$. From the above, our claim in (13) is true for $k = 1$. Now assuming that the claim is true when $k = m$, we have that for all $i \in \mathcal{S}$.

$$|V^m(i)| \leq M(1 + \lambda + \lambda^2 + \cdots + \lambda^m). \tag{16}$$

Now, if the update is carried out as in Case 1:

$$\begin{aligned}|V^{m+1}(i)| &\leq (1-\beta)|V^m(i)| + \beta|r(i,a,j) + \lambda V^m(j)| \\ &\leq (1-\beta)M(1+\lambda+\lambda^2+\cdots+\lambda^m) + \beta M \\ &+ \beta\lambda M(1+\lambda+\lambda^2+\cdots+\lambda^m) \text{ (from (16))} \\ &= M(1+\lambda+\lambda^2+\cdots+\lambda^m) - \beta M(1+\lambda+\lambda^2+\cdots+\lambda^m) \\ &+ \beta M + \beta M(\lambda+\lambda^2+\cdots+\lambda^{m+1}) \\ &= M(1+\lambda+\lambda^2+\cdots+\lambda^m) + \beta M\lambda^{m+1} \\ &\leq M(1+\lambda+\lambda^2+\cdots+\lambda^m) + M\lambda^{m+1} \\ &= M(1+\lambda+\lambda^2+\cdots+\lambda^m+\lambda^{m+1}).\end{aligned}$$

Now, if the update is carried out as in Case 2:

$$\begin{aligned}|V^{m+1}(i)| &= |V^m(i)| \leq M(1+\lambda+\lambda^2+\cdots+\lambda^m) \\ &\leq M(1+\lambda+\lambda^2+\cdots+\lambda^m+\lambda^{m+1}).\end{aligned}$$

From the above, the claim in (13) is proved for $k = m+1$. □

Lemma 2. *The sequence* $\{P^k\}_{k=1}^{\infty}$ *in Algorithm 1 remains bounded.*

Proof We will now show that $P(.,.)$ is bounded by $\bar{\mathsf{P}}$ where

$$\bar{\mathsf{P}} = \max\left\{ r_{\max} + (\lambda+1)\frac{M}{1-\lambda}, \max_{i,a}|P^1(i,a)| \right\}.$$

Again, we will use induction. Our claim is $|P(i,a)| < \bar{\mathsf{P}}$ for all (i,a) pairs.

From the actor update, for $k=1$, for any (i,a) pair,

$$\begin{aligned} |P^2(i,a)| &\leq (1-\alpha^1)|P^1(i,a)| + \alpha^1|r(i,a,j) + \lambda V^1(j) - V(i)| \\ &\leq (1-\alpha^1)\bar{\mathsf{P}} + \alpha^1\left| r_{\max} + \lambda\frac{M}{1-\lambda} + \frac{M}{1-\lambda}\right| \\ &\leq (1-\alpha^1)\bar{\mathsf{P}} + \alpha^1\bar{\mathsf{P}} = \bar{\mathsf{P}}. \end{aligned}$$

Assuming the result is true for $k=m$, i.e., $P^m(i,a) \leq \bar{\mathsf{P}}$, we have for any (i,a),

$$\begin{aligned} |P^{m+1}(i,a)| &\leq (1-\alpha^m)|P^m(i,a)| + \alpha^m|r(i,a,j) + \lambda V^m(j) - V^m(i)| \\ &\leq (1-\alpha^m)\bar{\mathsf{P}} + \alpha^m\left| r_{\max} + \lambda\frac{M}{1-\lambda} + \frac{M}{1-\lambda}\right| \\ &\leq (1-\alpha^m)\bar{\mathsf{P}} + \alpha^m\bar{\mathsf{P}} = \bar{\mathsf{P}}. \end{aligned}$$

□

Lemmas 1 and 2 together prove the result. □

A2. Algorithm for Average Reward SMDPs (Algorithm 2): We now analyze the boundedness of iterates in the algorithm for average reward SMDPs.

Theorem 3. *The sequence $\{V^k, P^k, \rho^k\}_{k=1}^{\infty}$ in Algorithm 2 remains bounded.*

Proof We will first need to show boundedness of the iterate ρ. In what follows, we will use an enhanced notation for the iterates R, T and ρ in the actor-critic for average-reward SMDPs, where R will be replaced by R^k to indicate its value in the kth iteration. Similarly, T^k and ρ^k will be used to indicate the value of T and ρ respectively in the kth iteration. The proof follows from [8], but is presented here for the sake of completeness.

Lemma 3. *The sequence $\{\rho^k\}$ in Algorithm 2 remains bounded.*

Let

$$\frac{\max_{i,a,j}|r(i,a,j)|}{\min_{i,a,j} t(i,a,j)} = \overline{\rho} < \infty,$$

where we assume that $t(i,a,j) > 0$ always. We can show that $|\rho^k| \leq \overline{\rho}$ for all k. Since R and T are initialized to 0, $|R^k| < k\max_{i,a,j}|r(i,a,j)|$ and $|T^k| > k\min_{i,a,j} t(i,a,j)$. Since all terms in the two previous inequalities are positive, we have that

$$\left|\frac{R^k}{T^k}\right| = \frac{|R^k|}{|T^k|} < \frac{\max_{i,a,j}|r(i,a,j)|}{\min_{i,a,j} t(i,a,j)}.$$

Since $\rho^1 = 0$ by assumption, we will need to first show (induction argument) that $\rho^2 \leq \overline{\rho}$. Now, have from the update in Equation (12)

$$|\rho^2| \leq (1-\gamma^1)|\rho^1| + \gamma^1 \frac{\max_{i,a,j} |r(i,a,j)|}{\min_{i,a,j} t(i,a,j)} = \gamma^1 \overline{\rho} < \overline{\rho}.$$

Next, we show that the result holds for $k+1$:

$$\begin{aligned} |\rho^{k+1}| &\leq (1-\gamma^k)|\rho^k| + \gamma^k \left| \frac{R^k}{T^k} \right| \\ &\leq (1-\gamma^k)\overline{\rho} + \gamma^k \left(\frac{\max_{i,a,j} |r(i,a,j)|}{\min_{i,a,j} t(i,a,j)} \right) \\ &= (1-\gamma^k)\overline{\rho} + \gamma^k \overline{\rho} = \overline{\rho}. \end{aligned}$$

□

When $r_{\max}$ in Lemma 1 is redefined as follows:

$$\text{where } r_{\max} = \max_{i,j \in \mathcal{S}, a \in \mathcal{A}(i)} |r(i,a,j)| + \overline{\rho} \max_{i,j \in \mathcal{S}, a \in \mathcal{A}(i)} t(i,a,j),$$

the rest of the proof will be similar to that of Theorem 2 and is hence skipped.
□

References

[1] D.P. Bertsekas and J. Tsitsiklis. *Neuro-Dynamic Programming.* Athena Scientific, Belmont, MA, USA, 1996.

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

[3] A. Gosavi. *Simulation-Based Optimization: Parametric Optimization Techniques and Reinforcement Learning.* Springer, New York, second edition, 2014.

[4] C. Szepesvári. Algorithms for reinforcement learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning, Morgan Claypool Publishers*, 10:1–103, 2010.

[5] A.G. Barto, R.S. Sutton, and C.W. Anderson. Neuron-like elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, 13:835–846, 1983.

[6] C.J. Watkins. *Learning from Delayed Rewards.* PhD thesis, Kings College, Cambridge, England, May 1989.

[7] V.S. Borkar. *Stochastic Approximation: A Dynamical Systems Viewpoint.* Hindusthan Book Agency, New Delhi, India, 2008.

[8] K. Kulkarni, A. Gosavi, S. Murray, and K. Grantham. Semi-Markov adaptive critic heuristics with application to airline revenue management. *Journal of Control Theory and Applications*, 9(3):421–430, 2011.

[9] A. Gosavi, S. Murray, J. Hu, and S. Ghosh. Model-building adaptive critics for semi-Markov control. *Journal of Artificial Intelligence and Soft Computing Research*, 2(1), 2012.

[10] A. Gosavi. How to rein in the volatile actor: A new bounded perspective. In *Procedia Computer Science, Complex Adaptive Systems, Philadeplphia, PA, Volume 36*, pages 500–507. Elsevier, 2014.

[11] R.J. Lawhead, A. Gosavi, and S.L. Murray. A bounded actor-critic reinforcement learning algorithm with applications in airline revenue management. In *Annual Intelligence Systems (ISC) Graduate Research Symposium.* Missouri University of Science and Technology, Rolla, MO, 2017.

[12] K. Talluri and G. van Ryzin. *The Theory and Practice of Revenue Management.* Kluwer Academic, Boston, MA, 2004.

[13] A. Gosavi, N. Bandla, and T. K. Das. A reinforcement learning approach to a single leg airline revenue management problem with multiple fare classes and overbooking. *IIE Transactions*, 34(9):729–742, 2002.

[14] J. Baxter and P. Bartlett. Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence*, 15:319–350, 2001.

[15] I. Grondman, L. Busoniu, G. Lopes, and R. Babuska. A survey of actor-critic reinforcement learning: Standard and natural policy gradients. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(6):1291–1307, 2012.

[16] F.L. Lewis and D. Vrabie. Reinforcement learning and adaptive dynamic programming for feedback control. *IEEE Circuits and Systems Magazine*, 3rd Quarter:32–50, 2009.

[17] G. Venayagamoorthy, R. Harley, and D. Wunsch. Comparison of heuristic dynamic programming and dual heuristic programming adaptive critics for neuro-control of a turbo-generator. *IEEE Transactions on Neural Networks*, 13 (3):764–773, 2002.

[18] P. J. Werbös. Building and understanding adaptive systems: A statistical/numerical approach to factory automation and brain research. *IEEE Transactions on Systems, Man., and Cybernetics*, 17:7–20, 1987.

[19] D. Liu, X. Xiong, and Y. Zhang. Action-dependent adaptive critic designs. In *Proc. INNS-IEEE Int. Joint Conf. Neural Networks, Washington, DC*, pages 990–995. IEEE, 2001.

[20] G.A. Rummery and M. Niranjan. On-line Q-learning using connectionist systems. Technical Report CUED/F-INFENG/TR 166. Engineering Department, Cambridge University, 1994.

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

[22] N. Aissani, B. Beldjilali, and D. Trentesaux. Dynamic scheduling of maintenance tasks in the petroleum industry: A reinforcement approach. *Engineering Applications of Artificial Intelligence*, 22(7):1089–1103, 2009.

[23] P. Pontrandolfo, A. Gosavi, O.G. Okogbaa, and T.K. Das. Global supply chain management: A reinforcement learning approach. *International Journal of Production Research*, 40(6):1299–1317, 2002.

[24] A. Mortazavi, A.A. Khamseh, and P. Azimi. Designing of an intelligent self-adaptive model for supply chain ordering management system. *Engineering Applications of Artificial Intelligence*, 37:207 – 220, 2015.

[25] S.K. Chaharsooghi, J. Heydari, and S.H. Zegordi. A reinforcement learning model for supply chain ordering management: An application to the beer game. *Decision Support Systems*, 45:949959, 2008.

[26] J. Kober, J.A. Bagnell, and J. Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

[27] S. Ross. *Applied Probability Models with Optimization Applications.* Dover, New York, USA, 1992.

[28] D.P. Bertsekas. *Dynamic Programming and Optimal Control.* Athena Scientific, Belmont, MA, USA, 3rd edition, 2007.

[29] J.I. McGill and G. J. van Ryzin. Revenue management: Research overview and prospects. *Transportation Science*, 33(2):233–256, 1999.

[30] M. J. Durham. The future of SABRE. In D. Jenkins, editor, *The Handbook of Airline Economics*, pages 485–491. 1995.

[31] Edward PC Kao. *An Introduction to Stochastic Processes.* Duxbury Press, 1996.

[32] K. Talluri and G. van Ryzin. Revenue management under a general discrete choice model of consumer behavior. *Management Science*, 50(1):15–33, 2004.

[33] L. Chen and T. Homem-de Mello. Re-solving stochastic programming models for airline revenue management. *Annals of Operations Research*, 177: 91–114, 2010.

[34] J. Dai, A. Kleywegt, and Y. Xiao. Network revenue management with cancellations and no-shows. *Production and Operations Management (in press)*, 2018.

[35] A. Gosavi. A reinforcement learning algorithm based on policy iteration for average reward: Empirical results with yield management and convergence analysis. *Machine Learning*, 55:5–29, 2004.

[36] K. Littlewood. Forecasting and control of passenger bookings. *In Proceedings of the 12th AGIFORS (Airline Group of the International Federation of Operational Research Societies Symposium)*, pages 95–117, 1972.

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

[38] M. A. Wiering, R. P. Salustowicz, and J. Schmidhuber. Model-based reinforcement learning for evolving soccer strategies. In *Computational Intelligence in Games*. Springer Verlag, 2001.