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 $\psi$ did not improve the results from the RL algorithm, and hence $t$ and $\psi$ were dropped from the state space. Such approximation of state-space is common in the RL community (Sutton and Barto, 1998). The following function from Gosavi (2004a) 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 $i$th class and $\theta$ 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 $\theta$ 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 $\phi$ is often referred to as a feature in the literature (Bertsekas and Tsitsiklis, 1996). 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 $\phi$ 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 $(c, \phi)$.

*EMSR-b.* A widely used heuristic in the airline industry for solving the single-leg version of the problem is called EMSR-b (Talluri and van Ryzin, 2004b). 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 $i$th 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 $i$th 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 $i$th fare class and (ii) the so-called aggregate revenue, $\bar{f}_i$, for the $i$th 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 $i$th fare class and that for all classes with fares exceeding $f_i$. Also, for $i = 1, 2, \ldots, n$,

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

Next, the heuristic solves the following equation, also called Littlewood's equation (Littlewood, 1972): For $i = 1, 2, \ldots, n-1$,

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

where $P_{i+1}$ is the so-called protection level for the $i$th 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, $\bar{Y}_i$, 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 $i$th 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. Section 3.1 discusses the background material focusing on the classical actor–critic. Section 3.2 presents the discounted-reward version on the MDP model. Section 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 (Bertsekas, 2007) 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 $i$
- $\eta$: A tunable contraction factor, set in the range $(0,1)$, close to 1
- $\lambda$: The discount factor in discounted-reward MDPs
- $\alpha$: The learning rate or step-size for the actor
- $\beta$: The learning rate or step-size for the critic
- $\gamma$: 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 $\bar{P}$ be a large positive number, such that $e^{\bar{P}}$ 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

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

    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:

    $$P(i,a) \leftarrow P(i,a) + \alpha\left[r(i,a,j) + \lambda V(j) - V(i)\right]. \quad (3)$$