2018 European Control Conference (ECC)
June 12-15, 2018. Limassol, Cyprus

Price Management in Resource Allocation Problem with Approximate Dynamic Programming

Ali Forootani, Massimo Tipaldi, Davide Liuzza, and Luigi Glielmo

Abstract— **The problem of managing the price for resource allocation arises in several applications, such as purchasing plane tickets, reserving a parking slot, booking a hotel room or renting SW/HW resources on a cloud. In this paper, we model a price management resource allocation problem with parallel Birth-Death stochastic Processes (BDPs) to account for the fact that the same resource can be possibly purchased by customers at different prices. In addition, customers can hold the resource at the purchase price to the necessary extent. The maximization of the revenue in both the finite and infinite time horizon cases is addressed in this paper with Stochastic Dynamic Programming (DP) approaches. To overcome the difficulty in solving the corresponding optimization problem due to the state space explosion, Approximate Dynamic Programming (ADP) techniques (in particular, the Least Square Temporal Difference method along with Monte Carlo simulations) are adopted. Furthermore, a MATLAB Toolbox is developed with the aim of solving stochastic DP/ADP problems and supporting probabilistic analysis. Extensive simulations are performed to show the effectiveness of the proposed model and the optimization approach.**

I. INTRODUCTION

Resource allocation and price management are crucial problems for several different companies that need to arrange the production of goods or provide a service. These type of problems arise, among others, in transportation [17], hospital ward optimization [2], business process management [10], and, recently, cloud computing [12]. A possible way to approach resource allocation problems and, related pricing management is to see them as sequential decision making problems, and thus leveraging on the Markov Decision Process (MDP) approaches [1]. This idea has been recently investigated in [17], where the problem is formulated as an MDP with significant model simplification to define the allocation policy.

In [18], the resource allocation problem has been addressed as MDP with service disruptions (e.g., a flight delay) and random resource capacities. In [2], the service rate in the hospital ward management is considered to define a tactical decision tool, able to optimize the matching between resources and demand. In cloud computing, Approximate Dynamic Programming (ADP) has been applied to a state space with actions of accepting/rejecting [13]. In some other works, resource allocation has been applied to business process management where the problem has been modeled as an MDP and heuristic reinforcement learning was applied to solve it [16] [10]. Moreover, in [3], a resource allocation problem has been set to decide how to share resources among different companies facing financial difficulties.

Conversely to these papers, we consider here the case where a certain resource can be sold at different prices at different times. This kind of scenario is common in the everyday life when customers purchase airplane or train tickets, theater tickets or reserve a room in a hotel. In this regard, we model the resource allocation problem as a set of parallel Birth-Death Processes (BDPs) [7], each of them corresponding to a specific price. The overall management strategy will assign a price policy at each time instant, implicitly exploiting how much (and how long) different customers are willing to pay for the resource. Contrary to the previous works, our approach can cope with any kind of price level (not only accepting/rejecting) and any possible arbitrary time interval the customer may hold the resource (e.g., in a hotel room booking, the customers are assigned to rooms with different day prices and different time periods). The problem will be formally addressed with a DP framework, thus recasting the pricing problem as an optimization problem for the considered parallel BDPs model.

Since DP suffers from the *curse of dimensionality* [5], ADP methods, such as Least Square Temporal Difference (LSTD) [6] and state aggregation methods [21], have been taken into account. In particular, the former takes the advantage of Monte Carlo simulations and linear features-based approximation to address the curse of dimensionality and to perform high-dimensional linear algebra calculations by using low-dimensional matrix-vector operations [6]. Based on DP and LSTD methods, we have developed an ad-hoc MATLAB based toolbox that performs the state space construction for the problem in hand, the related probability analysis, the exact DP, the Monte Carlo simulation, and the ADP.

The rest of the paper is organized as follows. Background on BDPs, DP, and ADP is given in Section II. Section III is dedicated to model the price management problem for resource allocation based on parallel BDPs. The problem definition as DP and ADP optimization is given in Section IV. Simulation results are discussed in Section V and finally, conclusion is drawn in Section VI.

II. BACKGROUND

A BDP is a continuous-time Markov chain with an ordered set of states, where the state transitions among two consecutive states are of only two types: *birth*, with a transition to the successive state, and *death*, with a transition to the

The authors are with the Department of Engineering, University of Sannio, Benevento, Italy. {aforootani, mtipaldi, davide.liuzza, glielmo}@unisannio.it

preceding state. So, calling $y(t)$ the state at current time t, with $y(t) \in Y = \{\upsilon^0, \upsilon^1, \dots\}$, when a birth occurs, the process goes from state υ^n to υ^{n+1}, while, when a death occurs, the process goes from state υ^n to state υ^{n-1} [7]. This kind of process can be represented as a Poisson process.

Poisson processes can be approximated as Bernoulli processes by choosing an appropriate sampling time [4].

The Bernoulli approximation allows us to model the process as a discrete time Markov chain. This will be useful later for applying DP algorithms. Specifically, calling $x(k)$ the state of the sampled-time BD Bernoulli process at time k, with $x \in X = \{\xi^0, \xi^1, \dots, \xi^{\chi-1}\}$, χ being its cardinality, the BD process can be written with the following Markov chain:

$$P\big[x(k+1) = \xi^0\big] = (1-\lambda(k))P\big[x(k) = \xi^0\big] + \\ + \mu(k)P\big[x(k) = \xi^1\big], \quad (1a)$$

$$P\big[x(k+1) = \xi^n\big] = (1-\lambda(k)-\mu(k))P\big[x(k) = \xi^n\big] \\ + \mu(k)P\big[x(k) = \xi^{n+1}\big] \\ + \lambda(k)P\big[x(k) = \xi^{n-1}\big], \quad (1b)$$

$$P\big[x(k+1) = \xi^{\chi-1}\big] = \lambda(k)P\big[x(k) = \xi^{\chi-2}\big] \\ + (1-\mu(k))P\big[x(k) = \xi^{\chi-1}\big], \quad (1c)$$

with $n = 1, 2, \dots, \chi - 2$.

An MDP [5] is defined by a tuple $\mathscr{C} = \langle X, U, \mathscr{T}, R \rangle$ where,

- X is a finite set of states,
- U is a finite set of actions,
- $\mathscr{T} \subseteq X \times U \times X$ is the transition probability relation from state $x(k) \in X$ to state $x(k+1) \in X$ after that action $u(k) \in U$ at time k is taken,
- $R : X \times U \to [0, +\infty)$ is the reward function, obtained considering action u at state x. In our case, as detailed later, the reward will only depend on the state, and so we will write $R(x)$.

Dynamic programming (DP) is a general technique for solving stochastic sequential decision problems modeled as MDPs. Denoting with $\pi = \{u(0), \dots, u(T-1)\}$ the *policy* (i.e., the sequence of function u applied on the state space over a horizon T, where for simplicity we removed the dependency to state space), and the cumulative revenue with

$$J_\pi(x(0)) = \\ E\left\{ R(x(T)) + \sum_{k=0}^{T-1} \alpha^k R(x(k+1)|x(k), u(k)) \right\}, \quad (2)$$

where $E\{\cdot\}$ is the expectation operator, $0 < \alpha \leq 1$ discount factor, $J_\pi(x(0))$ expected value function if policy π is applied for the entire horizon if the process started from initial state $x(0)$. With slightly abuse of notation, we simply denote $J(x)$, as the value function for any state $x \in X$. The aim is to find the optimal policy π^* that maximizes the expected total reward defined as

$$J^*(x(0)) = \max_\pi J_\pi(x(0)). \quad (3)$$

It has been proved that DP satisfies monotonicity condition and contraction mapping for discounted problems [5].

It is also well known that DP suffers from the *Curse of Dimensionality* [5] due to the state space explosion as well as action space. For this reason, efforts have been devoted for finding techniques able to solve problem (3) in an approximate way. Approximate Dynamic Programming (ADP) techniques have been derived both for infinite and finite time horizon problems [14].

The key idea to cope with the state space explosion is to substitute the original value function with an approximated representation defined over a restricted set of selected features [8], [19]. Among the available methods, the so called linear architecture consists in expressing an approximation of the value function $J(x)$ at each state, as $\tilde{J}(x,r) = \phi(x)'r$, where $\phi(x) = [\phi_1(x), \phi_2(x), \dots, \phi_q(x)]'$ is a vector of feature functions evaluated over x and $r = [r_1, r_2, \dots, r_q]'$ is a vector of parameters. Note that, in general we have $q \ll \chi$. Different algorithms have been proposed in the literature [6]. Here we describe the one adopted in our paper, namely the Least Square Temporal Difference (LSTD) along with Monte Carlo simulations method to evaluate the current policy and approximate the value function. In the LSTD approach, the vector r is calculated via successive approximations called $\hat{r}$. Such recursive iterations are computed via Monte Carlo simulations, hence it is called $\hat{r}$. The core of such an algorithm is as the following:

while $s \leq M$, $s \in \{1, 2, \dots, M\}$ **do**
- Generate new state $x(s)$ starting from $x(s-1)$ based on the transition probability matrix of one of the actions by Monte Carlo simulation.
- Compute the corresponding $\phi(x(s))$.
- Compute the corresponding value component $R(x(s))$.
- Calculate matrix C_s by $C_s = \frac{1}{s+1}\sum_{l=0}^{s} \phi(x(l)) \Big(\phi(x(l)) - \alpha\phi(x(l+1)) \Big)'$.
- Calculate y_s by $y_s = \frac{1}{s+1}\sum_{l=0}^{s} \phi(x(l))R(x(l))$.
- Having C_s and y_s, we choose $\hat{r}$ by following recursive iterations:
- **if** $s = 1$ **then**
  $\hat{r}_2 = (C_1'\Sigma^{-1}C_1 + \beta I)^{-1}(C_1'\Sigma^{-1}y_1 + \beta\bar{r})$,
  else
  $\hat{r}_{s+1} = (C_s'\Sigma^{-1}C_s + \beta I)^{-1}(C_s'\Sigma^{-1}y_s + \beta\hat{r}_s)$.
  end
- Put $s \leftarrow s+1$

end

where $\bar{r}$ is a heuristic guess based on some intuition about the problem, $\hat{r}_s$ is the updated approximation of vector r after each iteration, Σ usually is selected as a symmetric positive definite matrix, β is a positive scalar, C_s is a positive definite matrix. For more details, we refer the reader to [11], where the *LSTD* method and its convergence proof has been deeply investigated.

III. Price management modeling

In this section, we model the problem of price management for resource allocation via parallel stochastic Markov chains. Specifically, we consider the problem of dynamically pricing N resources when customers ask for allocating them along the time. We suppose that the management of the customer resource allocation/deallocation requests can be modeled via a stochastic Poisson process. At each time instant the decision maker proposes the price of the resource among m possible different choices. Note also that, when the resource is allocated to the customer, the latter may hold it at the purchase price to the necessary duration. Notice that, the price manager will in principle apply, for the same resource, different prices to different customers, since they will ask for the resource at different times and with a different availability of the resource itself. In this regard, each BDP corresponds to a certain price, while the state of each chain represents the current number of customers holding a resource at that price. The controller will dynamically assign the price to a new customer (by deciding at each time which BDP is active for births), while all other chains are active for deaths (since users may release the resource at any time). By introducing the controller, parallel BDPs are so recasted as MDPs.

The modeling, assumptions and notations can be defined as follows:

- We consider m different prices *per unit of time*, namely $c_1, c_2, \ldots, c_m$ (and hence m different BDPs).
- $x_i(k)$ denotes the number of customers with assigned cost c_i at time k, with $i \in \{1, \ldots, m\}$. The overall state of the system at time k will be represented in stack form as $x(k) = (x_1(k), \ldots, x_m(k))'$.
- We denote by $\lambda_i(k)$, $i = 1, \ldots, m$, at time k, the probability a customer requires a resource at cost c_i.
- We denote by $\mu_i(k)$, $i = 1, \ldots, m$, at time k, the probability a customer releases the resource she previously purchased at cost c_i. For our model, we will consider the case of constant $\lambda_i(k)$ and $\mu_i(k)$, and so, we will simply write λ_i and μ_i, omitting the explicit dependence on time.
- The parallel BDPs are not independent since they share the common constraint $\sum_{i=1}^{m} x_i \leq N$, where N is the total number of resources.

The objective of the price manager is to decide the appropriate price at each time slot k such that the maximum revenue is collected at the (possibly unbouded) end of the time horizon T.

The MDP *regarding each price* c_i is defined by a tuple $\mathscr{C}_i = \langle X_i, U_i, \mathscr{T}_i, R_i \rangle$ where,

- X_i is the state space, $X_i = \{\xi_i^0, \xi_i^1, \cdots, \xi_i^N\}$. The state variable at time k is denoted with $x_i(k)$, where $x_i(k) \in X_i$. In simpler words, when $x(k) = \xi_i^{h_i}$, this means that h_i resources are allocated at current time k at price c_i. Notice that the number of resources is supposed finite and equivalent to N.
- U_i is a finite set of actions (also called inputs), defined as $U_i = \{c_i, \nu\}$, where c_i represents "allocation" with price c_i and ν denotes the action of "no allocation". With a slight abuse of notation, We denote with $U_i(x_i)$ the set of input admissible at state x_i. Hence, for each state $\xi_i^{h_i}$, with $h_i = 0, 1, \ldots, N$, we have

$$\begin{cases} U_i(\xi_i^{h_i}) = \{c_i, \nu\}, & \text{if} \quad h_i < N, \\ U_i(\xi_i^N) = \{\nu\}. \end{cases} \tag{4}$$

  The input variable at time k is denoted as $u_i(k) \in U_i(x_i(k))$.
- $\mathscr{T}_i \subseteq X_i \times U_i \times X_i$ is the transition probability relation with elements

$$p_{\xi_i^v \xi_i^w}(u_i(k)) := P\big(x_i(k+1) = \xi_i^w | x_i(k) = \xi_i^v, u_i(k)\big). \tag{5}$$

- $R : X_i \to [0, +\infty)$ is the reward function. In our case

$$R(x_i) = c_i x_i, \tag{6}$$

where, we write the expression $c_i x_i$ to mean $c_i h_i$, with $x_i = \xi_i^{h_i}$. Similarly, in what follows we will write x_i to mean the index h_i of the state $\xi_i^{h_i}$ assumed by x_i.

Note that, for the problem considered, the reward depends only on the state, since the information about the number of customers and the applied price is completely as known by only knowing the state at the current time. The MDP $\mathscr{C}_i$ is shown in Fig. 1. Such figure depicts the transition probabilities among the various states for control value, at time instant k. With the decision $u_i = c_i$, the resulting Markov chain $\mathscr{C}_i$ allows a birth transition from the current state with probability λ_i (representing the probability that a customer requires the resource at price c_i), and a death transition with probability μ_i (representing the probability a customer releases a resource previously purchased at price c_i) and a transition with probability $1 - \lambda_i - \mu_i$ representing the fact that no customer releases or asks for a resource. On the other hand, when the decision $u_i = \nu$ is taken, no customer can purchase the resource, and only a death transition and a self-transition from the current state are allowed.

The "composition" of m different $\mathscr{C}_i$ corresponding to each price and a common constraint representing the fact that the number of resources is equal to N, gives the overall system $\mathscr{C}$.

The MDP of the overall system is defined by a tuple $\mathscr{C} = \langle X, U, \mathscr{T}, R \rangle$ where,

- X is the entire state space, $X = \big\{x = (x_1, \cdots, x_m)' \in \bigotimes_{i=1}^{m} X_i : ||x||_1 \leq N\big\}$, where, given $x = \xi^h = (\xi_1^{h_1}, \xi_2^{h_2}, \cdots, \xi_m^{h_m})$, we denote with $\|\cdot\|_1$ the expression $||x||_1 = \sum_{i=1}^{m} h_i$. In addition, $x(k)$ stands for the state at the time slot k.
- $U = \{c_1, \ldots, c_m, \nu\}$ is the overall input set. With a slight abuse of notation, we denote with $U(x)$ the set of inputs admissible at state x. We have

$$\begin{cases} U(\xi^h) = \{c_1, \ldots, c_m, \nu\}, & \text{if} \quad ||\xi^h||_1 < N, \\ U(\xi^h) = \{\nu\}, & \text{if} \quad ||\xi^h||_1 = N. \end{cases} \tag{7}$$

  We denote with $u(k)$ the input at time k.

- $\mathscr{T} \subseteq X \times U \times X$ is the transition probability relation, whose elements are

$$p_{\xi^v \xi^w}(u(k)) := P\big[x(k+1) = \xi^w | x(k) = \xi^v, u(k)\big], \quad (8)$$

such that

$$p_{\xi^v \xi^w}(c_i) = p_{\xi_i^{v_i} \xi_i^{w_i}}(c_i) \cdot \prod_{\substack{j=1 \\ j \neq i}}^{m} p_{\xi_j^{v_j} \xi_j^{w_j}}(\nu), \quad \forall\ i = 1, \cdots, m \quad (9)$$

$$p_{\xi^v \xi^w}(\nu) = \prod_{j=1}^{m} p_{\xi_j^{v_j} \xi_j^{w_j}}(\nu). \quad (10)$$

- $R : X \to [0, +\infty)$ is the reward function, defined as this case

$$R(x) = \sum_{i=1}^{m} c_i x_i. \quad (11)$$

Note that, as for the case of the single $\mathscr{C}_i$, the overall reward depends only on the state. In order to highlight the

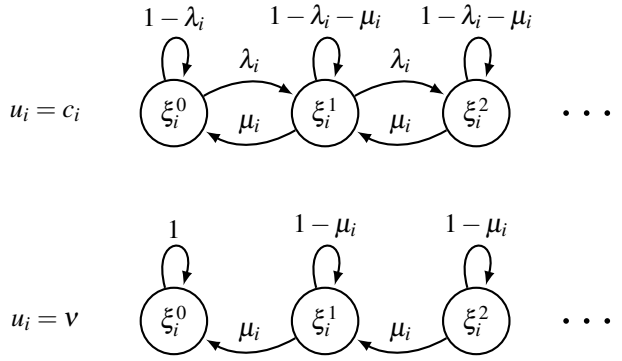

Fig. 1. MDP $\mathscr{C}_i$. Transitions allowed with the input $u_i = c_i$ are depicted in the top. Transitions allowed with the input $u_i = \nu$ are depicted in the bottom.

complexity of the problem, it can be shown that for each state of the entire process there are $3 \times 2^{m-1}$ transitions.

IV. PROBLEM FORMULATION AS DYNAMIC PROGRAMMING

The modeling conducted in the previous section allows us to solve the problem with DP and ADP tools both for the finite and infinite time horizon cases. In what follows, we briefly describe these two cases. Specifically, we aim at finding the optimal control that maximizes the total revenue over the time horizon T. More formally, we aim at solving the following optimization problem

$$J^*(x(0)) \triangleq \max_{\pi}\ E\left[\sum_{k=0}^{T}\sum_{i=1}^{m} c_i x_i(k)\right].$$

The sequence $\pi^* = \{u^*(0), u^*(1), \cdots, u^*(T-1)\}$ maximizing the above objective function will be the optimal policy. By exploiting Bellman's principle of optimality [5], the optimal revenue function at time k can be written as:

$$J_k(x(k)) = \max_{u(k) \in \{c_1, c_2, \ldots, c_m, \nu\}} E\left[\sum_{i=1}^{m} c_i x_i(k) + J_{k+1}(x(k+1))\right]. \quad (12)$$

A. Exact Dynamic Programming

By applying Bellman's recursive backward procedure and starting from the terminal condition

$$J_T(x(T)) = \sum_{i=1}^{m} c_i x_i(T),$$

the optimal revenue function can be expressed as:

$$\begin{aligned} J_k(x(k)) &= \max_{u(k) \in \{c_1, c_2, \ldots, c_m, \nu\}} E\left\{\sum_{i=1}^{m} c_i x_i(k) + J_{k+1}(x(k+1))\right\} \\ &= \sum_{i=1}^{m} c_i x_i(k)\ + \max_{u(k) \in \{c_1, c_2, \ldots, c_m, \nu\}} E\ \{J_{k+1}(x(k+1))\} \\ &= \sum_{i=1}^{m} c_i x_i(k) + \max\Big[E\left\{J_{k+1}(x(k+1))\ |u(k) = c_1\right\}; \\ &\qquad \ldots; E\left\{J_k(x(k+1))\ |u(k) = c_m\right\}; \\ &\qquad E\left\{J_k(x(k+1))\ |u(k) = \nu\right\}\Big]. \end{aligned}$$

$J_0(x(0))$, generated at the last step, is equal to the optimal revenue $J^*(x(0))$.

B. Approximate DP for Infinite Horizon

For the infinite horizon case, we assume the following discounted objective function

$$J^*(x(0)) = \max_{\pi} \lim_{T \to \infty} E\left[\sum_{k=0}^{T} \alpha^k \sum_{i=1}^{m} c_i x_i(k)\right]. \quad (13)$$

For the value function, we use an approximation, linear in feature functions $\phi_i(x)$, as described in Section II. We define a total of $1+m$ basis functions depending on the state $x = (x_1, \ldots, x_m)$ as

$$\phi_0(x) = 1, \quad \phi_i(x) = x_i, \quad i = 1, \ldots, m. \quad (14)$$

The related linear approximation is

$$\tilde{J}(x, r) = r_0 + \sum_{i=1}^{m} r_i x_i, \quad (15)$$

where $r' = (r_0, r_1, \ldots, r_m)$ is the parameter vector.

Moreover, once we have calculated the vector r by simulation, it is possible to assign an optimal policy to a given state $x(k)$ by the following formula

$$u^*(x(k)) = \arg \max_{u \in \{c_1, c_2, \ldots, c_m, \nu\}} E\left[\sum_{i=1}^{m} c_i x_i(k) + \alpha \phi(x(k+1))' r\right], \quad (16)$$

where $\phi(x)$ is given in Section II.

V. SIMULATION RESULTS

In this section, some numerical examples are reported in order to show the effectiveness of the proposed method. Both exact DP and ADP techniques have been used with the support of our on-purpose developed MATLAB toolbox. The latter is configurable, meaning that it provides the capability of defining the values of the problem to be solved, e.g. λ_i and μ_i for $i \in \{1, \ldots, m\}$, the number of resources (N) , the prices (m), the time horizon (T). When ADP techniques

are adopted, Monte Carlo based simulations are taking into account the dynamical model of the process to approximate the value function.

A. Exact DP

Table I highlights the state space explosion even with relatively small values of parameters m and N. Indeed, increments in such parameters cause an exponential growth in the size of the state space. Hence, the exact DP becomes impractical also for small problem instances. As a consequence, we have used an exact DP approach only for problems with a limited number of states.

TABLE I
STATE SPACE DIMENSIONS WITH DIFFERENT NUMBER OF RESOURCES AND PRICES

Example of state space explosion		
m	N	Number of States
2	10	66
3	20	1 771
5	20	53 130
6	20	230 230
4	50	316 251
5	50	3 478 761

Example 1: Number of prices $m = 3$, resources $N = 4$, horizon $T = 60$; price $c = [0.9\ \ 1\ \ 1.1]$; $\lambda = [0.6\ \ 0.5\ \ 0.3]$; $\mu = [0.2\ \ 0.2\ \ 0.4]$;

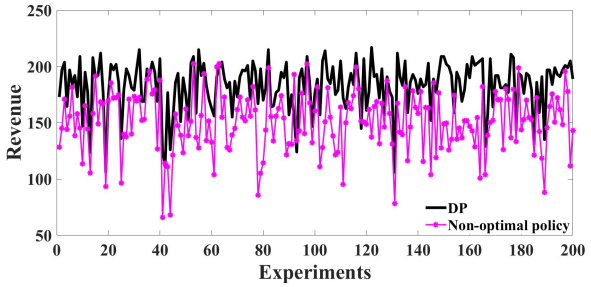

Fig. 2. Comparison of exact DP versus non-optimal policy for Example 3.

In this example, we have considered 200 randomly generated experiments to compare the exact DP with the following non-optimal policy:

- Choose a random price coherently with its death rate, i.e., prices c_1 and c_2 are more likely to be chosen then c_3..

The results of these experiments are shown in Fig. 2. As expected, the mean value of the revenue achieved by DP is greater than the one achieved by the other non-optimal policy. The DP mean revenue is 183, whereas non-optimal policy mean revenue is 150.

B. ADP Monte Carlo simulation based on LSTD

For the ADP-LSTD Monte Carlo based simulations, we consider the following points:

- We select the basis functions for the approximate linear value function as described in Section IV.
- Due to the large number of states and the number of control actions for each of them, it is burdensome to perform the policy improvement step in the policy iteration algorithm. Therefore, given the current state $x(s)$ along the simulated Monte Carlo trajectory, the next state $x(s+1)$ is a function of the applied control action u. Hence, the calculated vector $\hat{r}_s$ is a function of u, as well. In each single iteration s of the LSTD algorithm, for any generic state $x(s)$, we have $u \in \{c_1, c_2, \ldots, c_m, v\}$ possible actions to generate the next new states. Therefore, there will be $m+1$ possible new states. Correspondingly, the calculated vector $\hat{r}_s$, which has been evaluated at each control action u, can assume $m+1$ possible values. We simply denote it with $\hat{r}_s(u)$ to show the dependency of vector $\hat{r}_s$ on the control action u. In this paper, heuristically, we choose the control action u at the iteration s such that 2-norm of vector $\hat{r}_s$ is maximized. It is worth noting that this approach does not contradict the contraction mapping of LSTD-ADP since there is no assumption on the stationary policy to be evaluated.
- We use multiple simulation trajectories, each having the length M as mentioned in Section II with an arbitrarily chosen initial state. Once a trajectory is terminated, the calculated $\hat{r}_s$ is replaced as the initial guess for the next one, see Section II.

Example 2: Number of prices $m = 4$, resources $N = 50$: $\alpha = 0.99$, price $c = [0.9\ \ 1\ \ 1.1\ \ 1.2]$, $\lambda = [0.6\ \ 0.5\ \ 0.3\ \ 0.2]$, $\mu = [0.2\ \ 0.2\ \ 0.4\ \ 0.4]$, $\beta = 0.01$, an arbitrary positive symmetric matrix Σ. In this example, we use 1000 trajectories, 20000 steps in length. Linear basis functions of the form (14) have been used to represent the approximation subspace. The calculated parameter vector is $\hat{r}'_k = [3735.3\ \ 12.52\ \ 2.64\ \ 1.6\ \ 8.36]$. For this example, the exact DP algorithm can not be used due to the curse of dimensionality. Furthermore, we have compared our ADP method with the same non-optimal policy used for the case of finite horizon exact DP.

We have carried out 200 experiments over a time horizon of 100 samples to compare our ADP method with such non-optimal policies. The outcome of the experiments are shown in Fig. 3. Also for this case, it is possible to see that, the mean value of the revenue achieved by the ADP is greater than the one achieved by the other non-optimal policy for all the carried experiments. Several other experiments have been conducted to validate our approach, versus heuristic reasonable non-optimal policies. In all the cases, the considered approach shows higher revenue values. These experiments are not reported here due to space limits.

VI. CONCLUSIONS

A price management strategy for resource allocation problems has been modeled by using parallel Birth-Death processes (BDPs). Specifically, we have considered a problem where a decision maker optimally manages the dynamic

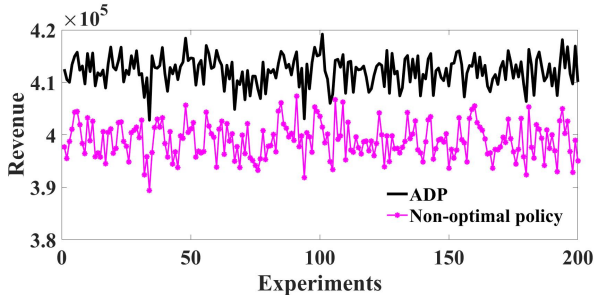

Fig. 3. Comparison of ADP versus non-optimal policy for Example 2.

prices of the resources, with the aim of achieving the maximum total reward.

To do so, we have considered a Stochastic Dynamic Programming in both the cases of finite and infinite time horizon. Our approach allows to consider any arbitrary number of price levels. However, when the number of available prices increases, the state space of our model grows exponentially. In order to cope with this drawback, we have exploited Least Square Temporal Difference along with Monte Carlo simulation as the method for ADP. A MATLAB Toolbox has been developed with the aim of solving DP/ADP problems and supporting probabilistic analysis. Extensive simulation results have showed the effectiveness of the proposed BDPs model, the related set of features chosen and the ADP adopted.

As future work, we plan to analytically investigate the convergence properties of our approach and to test it against currently applied heuristic methods and with other value function approximations.

REFERENCES

[1] O. Alagoz, H. Hsu, A. J. Schaefer, and M. S. Roberts, "Markov decision processes: a tool for sequential decision making under uncertainty," *Medical Decision Making, SAGE Publications*, vol. 30, no.4, pp. 474-483, 2010.

[2] A. R. Andersen, B. F. Nielson , and L. B. Reinhardt, "Optimization of hospital ward resources with patient relocation using Markov chain modeling," *European Journal of Operational Research*, vol. 260, no. 3, pp. 1152-1163, 2017.

[3] U. Ayesta, M. Erausquin, E. Ferreira , and P. Jacko, "Optimal dynamic resource allocation to prevent defaults," *Operations Research Letters*, vol. 44, no.4, pp. 451-456, 2016.

[4] D. P. Bertsekas, and J. N. Tsitsiklis, *Introduction to Probability*. USA: Massachusetts Institute of Technology, 2000.

[5] D. P. Bertsekas, *Dynamic Programming and Optimal Control*. USA: Athena Scientific, Belmont, Massachusetts, 1995.

[6] D. P. Bertsekas, "Approximate policy iteration: A survey and some new methods," *Journal of Control Theory and Applications*, vol. 9, no. 3, pp. 310-335, 2011.

[7] V. Bansaye, and S. Méléard, *Stochastic Models for Structured Populations*. Springer, 2015.

[8] D. P. De Farias, and B. Van Roy, "The linear programming approach to approximate dynamic programming," *Operations Research*, vol. 51, no. 6, pp. 850-865, 2003.

[9] L. Farina, and S. Rinaldi, *Positive Linear Systems: Theory and Applications*. John Wiley & Sons, vol. 50, 2011.

[10] Z. Huang, W. M. Van Der Aalst, Xu. Lu, and H. Duan, "Reinforcement learning based resource allocation in business process management," *Data & Knowledge Engineering*, vol. 70, no. 1, pp. 127-145, 2011.

[11] A. Nedić, D. P. Bertsekas, "Least squares policy evaluation algorithms with linear function approximation," *Discrete Event Dynamic Systems*, vol. 13, no. 1, pp. 79-110, 2003.

[12] C. Nguyen, P. Wang, D. Niyato, Y. Wen,and Z. Han, "Resource management in cloud networking using economic analysis and pricing models: a survey," *IEEE Communications Surveys & Tutorials*, 2017.

[13] A. Pietrabissa, F. D. Priscoli, Al. D. Giorgio, Al. Giuseppi, M. Panfili, and V. Suraci, "An approximate dynamic programming approach to resource management in multi-cloud scenarios," *International Journal of Control*, vol. 90, no. 3, pp. 492-503, 2017.

[14] W. B. Powell, *Approximate Dynamic Programming: Solving the curses of dimensionality*. John Wiley & Sons, vol. 703, 2007.

[15] M. J. Valenti, "Approximate dynamic programming with applications in multi-agent systems," Ph. D. Thesis, Dept. of Electrical Engineering and Computer Science, MIT., 2007.

[16] G. Vilcot, J. Ch. Billaut, "A tabu search and a genetic algorithm for solving a bicriteria general job shop scheduling problem," *European Journal of Operational Research*, vol. 190, no. 2, pp. 398-411, 2008.

[17] Xi. Wang, "Static and dynamic resource allocation models for single-leg transportation markets with service disruptions," *Transportation Research Part E: Logistics and Transportation Review*, vol. 103, pp. 87-108, 2017.

[18] Xi. Wang, "Stochastic resource allocation for containerized cargo transportation networks when capacities are uncertain," *Transportation Research Part E: Logistics and Transportation Review*, vol. 93, pp. 334-357, 2016.

[19] H. Yu, D. P. Bertsekas, "Convergence results for some temporal difference methods based on least squares," *IEEE Trans. Automatic Control*, vol. 54, no. 7 , pp. 1515-1531, 2009.

[20] M. A. Alsheikh, "Markov Decision Processes With Applications in Wireless Sensor Networks: A Survey," *IEEE Communication Surveys & Tutorials*, vol. 17, No. 3, Third Quarter, 2015.

[21] M. Tipaldi, and L. Glielmo, "State Aggregation Approximate Dynamic Programming for Model-based Spacecraft Autonomy," *European Control Conference (ECC)*, 2016.