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

ASYNCHRONOUS VALUE ITERATION FOR MARKOV DECISION PROCESSES WITH CONTINUOUS STATE SPACES

Xiangyu Yang
Jian-Qiang Hu

School of Management
Fudan University
Shanghai, CHINA

Jiaqiao Hu

Department of Applied Mathematics and Statistics
SUNY at Stony Brook
Stony Brook, NY 11794-3600, USA

Yijie Peng

Guanghua School of Management
Peking University
Beijing, CHINA

ABSTRACT

We propose a simulation-based value iteration algorithm for approximately solving infinite horizon discounted MDPs with continuous state spaces and finite actions. At each time step, the algorithm employs the shrinking ball method to estimate the value function at sampled states and uses historical estimates in an interpolation-based fitting strategy to build an approximator of the optimal value function. Under moderate conditions, we prove that the sequence of approximators generated by the algorithm converges uniformly to the optimal value function with probability one. Simple numerical examples are provided to compare our algorithm with two other existing methods.

1 INTRODUCTION

Markov decision processes (MDPs) (e.g., Bertsekas 2015) are among the most widely used frameworks for studying sequential decision making problems under uncertainty. However, numerically optimizing such models in practice has been recognized as a very challenging task because the optimal solution is characterized by an underlying functional equation called the Bellman equation, solving of which would require explicit knowledge of state transitions and the ability to enumerate all state-action pairs. Consequently, effective MDPs solutions typically rely on the idea of approximately solving the Bellman equation through a fusion of tools from function approximation and computer simulation (Chang et al. 2013; Gosavi 2014). This has given rise to a variety of techniques collectively known as reinforcement learning (RL) (Sutton and Barto 1998; Busoniu et al. 2010) or approximate dynamic programming (ADP) (Bertsekas and Tsitsiklis 1996; Powell 2007). Since many RL algorithms are rooted in classical approaches such as value iteration (VI) and policy iteration (PI), their applications are mostly centered on MDPs with discrete state spaces.

For continuous-state MDPs, a mainstream approach is based on a suitable (often implicit) discretization of the state space, so that the techniques developed for discrete-state problems can be directly carried over by working with a countable number of states. One popular algorithm of this kind is the fitted value iteration (FVI) (e.g., Chandramohan et al. 2010; Gordon 1995; Mbuwir et al. 2017). The algorithm requires a batch of transition samples to be pre-generated (off-line) at a given finite set of states, and uses a supervised learning procedure to iteratively approximate the value function based on the transition samples. Stachurski (2008) provides error bounds on the performance of FVI by using non-expansive function

approximators. Munos and Szepesvári (2008) show that as the numbers of states and transition samples become sufficiently large, FVI can achieve arbitrarily high accuracy with large probabilities. Other useful alternatives for continuous-state MDPs include the fitted Q-iteration (Ernst et al. 2005; Ernst et al. 2005) and fitted policy evaluation (Busoniu et al. 2010). The former is essentially a variant of FVI applied to estimating the Q-function, whereas the latter is used to gauge the performance of a given policy, which can be employed in conjunction with a policy improvement procedure to obtain better policies. Although the general idea of converting a continuous-state problem into a discrete-state one is natural and intuitive, the practical performance of these algorithms could be highly influenced by the choice of the pre-determined states and transitions samples. In addition, their convergence guarantees typically rely on non-expansive approximation of the value-/Q-function (Busoniu et al. 2010; Ernst et al. 2005; Ormoneit and Sen 2002), so care must be taken when selecting approximation techniques to prevent possible divergence.

In this paper, we present a new RL algorithm based on the classical VI for solving continuous-state MDPs. The algorithm complements the aforementioned approaches in the sense that it is fully online and discretization-free, and hence does not resort to the use of pre-selected transition samples. The underlying idea is to combine the principle of VI with those of interpolation-based approximation techniques and the shrinking ball method. Interpolation-based methods have been especially prominent in the simulation optimization (SO) literature (e.g., Barton 2009; Gutmann 2001; Jones et al. 1998; Kleijnen 2009) but less explored in the RL context. In a continuous-state RL setting, such an approximator has the benefit of allowing the value function at an unsampled state to be reliably predicted using estimates at previously sampled states, leading to good control decisions even at states that have not been visited thus far. The shrinking ball method was originally introduced in Baumert and Smith (2002) for estimating the objective function values in continuous SO. We adapt the method to construct value function estimates, so that the variance of the estimate at a visited state can be effectively reduced by averaging estimates at other near-by states sampled along a single trajectory produced by the algorithm.

For a given problem, our algorithm works with a learning policy and proceeds iteratively by generating new transition samples from the policy. This information is used at each step in an asynchronous version of the shrinking ball method to update the value function estimates at all previously encountered states. These point estimates are then fully retained in an interpolation-based fitting strategy to construct an approximator of the value function. We show that a careful coupling of these steps leads to a natural generalization of VI for continuous-state MDPs and that under moderate conditions, the sequence of value function approximators converges to the optimal value function in a uniform manner.

The rest of this paper is organized as follows. We begin by introducing the problem setting in Section 2. A detailed description of the proposed algorithm is then provided in Section 3. In Section 4, we present the main convergence property of our algorithm with a sketch of proof. In Section 5, we provide two simple numerical examples to illustrate and compare the performance of the algorithm with two existing approaches. Finally, we conclude the paper in Section 6.

2 PROBLEM SETTING

We consider an infinite horizon discounted MDPs defined by a tuple (S,A,p,R,β). We focus on the case where the state space S is a compact subset of $\mathbb{R}^d$ and the action space A is a (discrete) finite set. Let $p(u|s,a), s,u \in S, a \in A$ be the Markov transition density, $R(\cdot,\cdot) : S \times A \to \mathbb{R}$ be the reward function, and $\beta \in (0,1)$ be the discount factor. To simplify the exposition, we assume all actions $a \in A$ are admissible for all states $s \in S$ and that the one-stage reward $R(s,a)$ can be explicitly observed at a given state-action pair (s,a). Throughout our paper, we consider the scenario in which the transition density p is unknown but can be simulated.

Let Π denote the set of stationary deterministic Markovian policies. For each $\pi \in \Pi$, $\pi(s)$ denotes the action applied at state s under policy π. For an initial state $s_0 = s$, we define the value function associated with policy π as $V^\pi(s) = \mathsf{E}[\sum_{t=0}^{\infty} \beta^t R(s_t, \pi(s_t))|s_0 = s]$, where s_t is the state of the process at time t. The

objective is to find an optimal policy $\pi^* \in \Pi$ that attains the supremum of V^π, i.e.,

$$V^*(s) = V^{\pi^*}(s) = \sup_{\pi \in \Pi} V^\pi(s), \tag{1}$$

for all initial states $s \in S$. Under mild conditions (see, e.g., Rust 1997), it can be shown that there exists a policy π^* satisfying (1) and that the optimal value function $V^*(s)$ is uniquely determined by the solution to the following Bellman's equation:

$$\begin{aligned} V^*(s) &= \max_a \left[R(s,a) + \beta \int V^*(u) p(u|s,a) du \right] \\ &= \max_a \left[R(s,a) + \beta \mathsf{E}\left[V^*(s') \right] \right], \end{aligned} \tag{2}$$

where $s' \sim p(\cdot|s,a)$ is a generic random variable representing the new state observed after taking action a at state s.

3 ALGORITHM DESCRIPTION AND CONVERGENCE ANALYSIS

In this section, we provide a description of the proposed algorithm and present the main convergence result. We begin by introducing the following notation. Let $\{r_t > 0\}_{t=0}^{\infty}$ be a sequence of positive nonincreasing real numbers and $B(s,r)$ be an open ball centered at s with radius $r > 0$. For two states s_l and s_t sampled at iterations l and t ($l < t$), let $I_t(s_l) = I\{s_t \in B(s_l, r_t)\}$ be an indicator function indicating whether the state s_t is contained in the ball $B(s_l, r_t)$.

3.1 Algorithm Description

A main difficulty involved in solving a continuous-state MDP is that the state space S cannot be enumerated so that classical methods such as VI are no longer applicable. Therefore, one must instead consider suitable approximations of the value function by working with only a countable number of states. Similar to a typical RL method, our algorithm can be viewed as a simulation-based approach for approximately solving the Bellman equation (2). The basic idea is to work with a learning policy (i.e., an action selection distribution) and then iteratively construct a sequence of value-function approximators based on simulation samples generated from the policy in an online manner. In our algorithm, the function approximation is carried out using an interpolation-based fitting strategy. Such a strategy has been widely employed in simulation optimization to approximate the response surface of an unknown function (e.g., Barton 2009; Fan and Hu 2018; Gutmann 2001; Jones et al. 1998; Kleijnen 2009). Compared to other popular approaches such as linear combinations of basis functions and neural networks, an interpolation-based approximation technique does not require prior knowledge about the problem at hand and often results in smooth interpolators that allow easy quantification of their approximation errors.

Suppose at time t, a function approximator V_t of the value function V^* has been obtained. Then, a value function estimate $\tilde{V}_t(s_t)$ at the state s_t encountered at time t can be formed by replacing V^* in (2) with V_t,

$$\tilde{V}_t(s_t) = \max_a \left[R(s_t,a) + \beta \mathsf{E}\left[V_t(s'_{t+1}) \right] \right], \tag{3}$$

where $s'_{t+1} \sim p(\cdot|s_t,a)$ is the sampled next state. Note that since the approximator V_t has an analytical expression, the expectation involved in (3) can be evaluated very accurately (e.g., via Monte Carlo method by sampling a large number of s'_{t+1} from the transition density) with little computational effort. However, unlike the discrete-state-space setting where the estimate $\tilde{V}_t(s_t)$ can be repeatedly updated whenever s_t is revisited by an (online) algorithm, we cannot expect multiple visits to the same state in a continuous state domain. Existing approaches for addressing this issue are mostly off-line techniques (cf. e.g., Busoniu et al. 2010), where a common procedure is to predetermine a set of states within S and then update $\tilde{V}_t$ (and

V_t) only on the set of selected states. This, in essence, is equivalent to converting a continuous state-space MDP into a discrete one and then applying (approximate) value iteration on the discretized state space.

In our proposed algorithm, we circumvent this problem from a different viewpoint, by employing an online averaging procedure adapted from the shrinking ball method. The idea is not to use $\tilde{V}_t(s_t)$ to directly update the function approximator V_t, but to use it to improve the value function estimates at all other states that are close to s_t. Specifically, let $s_1,\ldots,s_{t-1}$ be the sequence of states generated from a learning policy prior to time t and denote by $\tilde{V}_{t-1}(s_l)$, $l=1,\ldots,t-1$, the value function estimates obtained at these states. Then for each state s_l sampled before time t, if the state s_t is considered to be sufficiently close to s_l, then the current estimate at s_l, $\tilde{V}_{t-1}(s_l)$, will be updated by incorporating $\tilde{V}_t(s_t)$; otherwise, $\tilde{V}_{t-1}(s_l)$ remains unchanged. This results in the following recursion:

$$\tilde{V}_t(s_l) = (1-\alpha_t(s_l)I_t(s_l))\tilde{V}_{t-1}(s_l) + \alpha_t(s_l)I_t(s_l)\tilde{V}_t(s_t), \tag{4}$$

where recall that $I_t(s_l) = I\{s_t \in B(s_l, r_t)\}$, and $\alpha_t(s_l) \in (0,1)$ is the learning rate used at time t, which may depend on the state s_l. The intuition is that while the state s_l cannot be revisited by the algorithm, its value function estimate can still be continuously improved by using estimates at other states that lie in the vicinity of s_l (assuming the smoothness of the value function). Note that in a discrete-space setting, if the shrinking ball radius r_t is sufficiently small, then each ball $B(s_l, r_t)$ will only contain s_l itself. Thus, the update (4) will only be carried out when the same state s_l is revisited by the algorithm, in which case (4) reduces to the usual stochastic averaging procedure. Equations (3) and (4), when coupled with an interpolation-based function approximator, lead to our proposed algorithm called adaptive asynchronous value iteration (AAVI), whose basic steps are presented in Algorithm 1.

Algorithm 1 Adaptive Asynchronous Value Iteration for Continuous-state MDPs

Input: A learning policy $\pi = \{\pi_t,\ t=0,1,\ldots\}$; an initial state s_0; learning rates $\alpha_t(s) \in (0,1)\ \forall s \in S$ and $\forall t$; shrinking ball radiuses $\{r_t\}$;

Initialize $V_0(s) = 0\ \forall s \in S$, $\Lambda_0 = \emptyset$ and the iteration counter $t = 0$;
for all $t = 0,1,\ldots$ **do**
  Set $\Lambda_{t+1} = \Lambda_t \cup \{s_t\}$;
  Obtain the point estimate of the value function at state s_t

$$\tilde{V}_t(s_t) = \max_a \left[R(s_t,a) + \beta \mathsf{E}\left[V_t(s'_{t+1})\right]\right],$$

  where $s'_{t+1} \sim p(s'|s_t,a)$;
  for all state $s_l \in \Lambda_t$ **do**

$$\tilde{V}_t(s_l) = (1-\alpha_t(s_l)I_t(s_l))\tilde{V}_{t-1}(s_l) + \alpha_t(s_l)I_t(s_l)\tilde{V}_t(s_t);$$

  end for
  Construct V_{t+1} by interpolating the data $\{(s, \tilde{V}_t(s)) : s \in \Lambda_t\}$;
  Apply π, obtain action $a_t = \pi_t(s_t)$, and simulate a next state $s_{t+1} \sim p(s|s_t,a_t)$.
  set $t = t+1$;
  if A stopping criterion is satisfied **then**
    terminate
  end if
end for
return $\{V_t\}$;

Note that the algorithm requires the function approximator to be updated at each iteration. Thus, on problems that require many iterations to arrive at a good solution, the cost associated for data fitting may

become prohibitive as the number of states in the interpolation increases. One simple empirical solution to alleviate this issue is to update the approximator every a few iterations (as opposed to every iteration). In addition, one may consider selecting a subset of the sampled states that tend to evenly fill the state space, and then constructing the interpolator only based on observations collected at the subset of states. In certain curve fitting techniques such as kriging (e.g., Kleijnen 2009), the uncertainty in the prediction can often be quantified. It is also possible to exploit such information in ways similar to those discussed in e.g., Sacks et al. (1989), Wang and Hu (2018), and adaptively determine whether a sampled state should be included in the construction of the function approximator. We leave this as a task for future research.

3.2 Convergence Result

For a given state s_l, let $N_t(s_l) = \Sigma_{j=l}^{t} I_j(s_l)$, signifying the number of times of the shrinking neighborhoods $B(s_l, r_j)$ $j = l, \ldots, t$ of s_l have been visited by the algorithm up to time t. We assume that the learning rate at s_l is a function of $N_t(s_l)$, i.e., $\alpha_t(s_l) = f(N_t(s_l))$ for some function f. For two sequences $\{a_t\}$ and $\{b_t\}$, we say $a_t = \Omega(b_t)$ if $\exists\ c, K > 0$ such that $a_t \geq c b_t$ for all $t \geq K$.

Our main result shows the uniform convergence of the sequence of function approximators $\{V_t\}$ to the optimal value function with probability one. This is established based on the following assumptions:

Assumption 1 $R_{\max} := \sup_{s,a} |R(s,a)| < \infty$ and there exists a constant $K_R < \infty$ such that for all $s, s' \in S$ and $a \in A$,

$$\left|R(s,a) - R(s',a)\right| \leq K_R d(s,s'),$$

where $d(s,s')$ is the Euclidean distance between states s and s'.

Assumption 2 The transition density $p(s'|s,a)$ is continuous and $p(s'|s,a) > 0$ for all $s, s' \in S$, $a \in A$. In addition, for all $s, s', s'' \in S$ and $a \in A$, there exists a function $K_p(s)$ satisfying $\int_S K_p(s) ds < \infty$ such that

$$\left|p(s''|s,a) - p(s''|s',a)\right| \leq K_p(s'') d(s,s').$$

Assumption 3 The interpolators V_t's are Lipschitz continuous with their Lipschitz constants uniformly bounded by $L < \infty$ w.p.1.

Assumption 4 The function f satisfies $f(i) \in (0,1)$ for all i and

$$\sum_{i=1}^{\infty} f(i) = \infty, \quad \sum_{i=1}^{\infty} f^2(i) < \infty.$$

Assumption 5 The shrinking ball radius r_t is nonincreasing in t and satisfies $r_t = \Omega(t^{-\gamma})$ for some constant $\gamma \in (0, \frac{1}{2d})$.

We remark that the first two assumptions ensure that the optimal value function is sufficiently smooth. This in turn justifies the use of the online averaging procedure (4). The Lipschitz continuity condition in Assumption 3 is weaker than the typical non-expansiveness requirement on the function approximator used in the existing literature (cf. e.g., Busoniu et al. 2010). Assumptions 4 and 5 are conditions on the algorithm input parameters. The main convergence theorem is stated below.

Theorem 1 Under Assumptions 1−5, we have

$$\lim_{t \to \infty} \sup_{s \in S} |V_t(s) - V^*(s)| = 0 \quad \text{w.p.1.}$$

Proof. Due to space constraints, we only provide a sketch of the proof. The detailed proof is highly similar to that of Hu et al. (2020). The main idea is to write (4) in the form of a generalized stochastic approximation recursion involving a bias term (due to the use of the shrinking ball strategy) and an error term caused by replacing the optimal value function V^* by V_t in (2). Consequently, the convergence of the point estimators can be studied using existing tools in stochastic approximation theory. Next, using the

interpolating structure of the approximator and the continuity of the value function, it can be shown that the values of V_t will eventually coincide with those of V^* at the interpolation states. Finally, the uniform convergence of V_t follows from a similar argument used in Tsitsiklis (1994). □

4 NUMERICAL EXAMPLES

In this section, we illustrate the proposed AAVI algorithm on a continuous-state queueing control problem in both one and two dimensions, and compare its performance with two other existing methods.

4.1 One-Dimensional Queueing Control Example

We consider an admission control problem in a single server queueing system. Denote w_t as the workload at time t, which is the "real" load on the processor. Without loss of generality, we assume that the processor works at a unit rate, i.e., if no further work arrives after t, then the server would become idle at $t + w_t$. Suppose that customers (or jobs) arrive at the system with constant (unit) interarrival times. Every time a customer arrives, the decision-maker faces two options: accept or reject the customer. If a customer is accepted, then the system receives a fixed reward $p > 0$; otherwise, no reward is received. In addition, a holding cost is incurred at a constant rate c_w per unit of workload per unit time. Specifically, at the beginning of a stage, if there is a workload $w \geq 0$ on the server, then the cost incurred during the stage is given by

$$c_w r(w) = c_w \int_0^1 (w-t)^+ dt = c_w \int_0^{1\wedge w} (w-t) dt = -\frac{c_w}{2}(w-t)^2 \Big|_{t=0}^{1\wedge w},$$

where we set $c_w = 1$ in our numerical experiments.

The problem can be modeled as an infinite-horizon discounted MDP with a continuous state space and a discrete action space. The state is the amount of workload w in the system, the action set contains two actions, "accept" and "reject", and the transition dynamics from times t to $t+1$ ($t = 0, 1, 2, \ldots$) are given by

$$w_{t+1} = \begin{cases} (w_t - 1)^+ + s, & a_t = \text{"accept"}; \\ (w_t - 1)^+, & a_t = \text{"reject"}, \end{cases}$$

where a_t is the action used at time t and s is the service time of the accepted customer. The goal is to maximize the total discounted return, where costs are viewed as negative rewards. It can be easily seen that the Bellman equation is given below:

$$V^*(w) = \max\left\{p - r(w) + \beta \mathsf{E} V^*((w-1)^+ + s), -r(w) + \beta V^*((w-1)^+)\right\}.$$

In our numerical experiment, we take $p = 2$, $\beta = 0.9$, and s is uniformly distributed between 0 and 3. Further, to ensure the set of states is compact, we set an upper bound on the amount of workload to 11 in our experiment. We assume that the service times of arriving customers are generated beforehand. If the required service time of an arrival plus the current workload exceeds the prescribed upper bound, then that arrival will be turned away. Thus, from the state transition equation, it is easy to observe that the state space is given by $[0, 10]$.

For comparison purposes, we apply three different approaches to approximate the optimal solution to the above queueing control problem: discretization method (DM), fitted value iteration (FVI), and the proposed AAVI algorithm. For DM, we first discretize the state space $[0, 10]$ into evenly-spaced points by using a mesh size 0.05 and approximate the service time using a discrete uniform random variable on $\{0.05k : k = 0, \ldots, 60\}$. This converts the original problem into a discrete-state MDP, whose transition probabilities can be computed exactly from the state transition equation. We then apply the standard VI algorithm on the discretized problem. FVI is implemented by pre-generating 5000 states uniformly

distributed over state space $[0,10]$. Then at each iteration, a polynomial regression model is iteratively used to fit the point estimates collected at these pre-generated states. We set the total number of algorithm iterations to 200. The parameters for AAVI are set as follows: learning rate $\alpha_t(s_l) = N_t(s_l)^{-0.9}$, shrinking ball radius $r_t = t^{-0.2}$, and the interpolation scheme is the ordinary kriging interpolation with an exponential semivariogram (implemented by using the PyKrige package https://pypi.org/project/PyKrige/). AAVI is run for two scenarios according to different behavior policies at each time step. One is an ε-greedy policy (AAVI$^{(1)}$) that chooses at each step an action uniformly with probability ε and chooses the greedy action (i.e., the action that maximizes (3)) with probability $1-\varepsilon$. Here, we set $\varepsilon = 0.1$. The other one is a nonuniform random policy (AAVI$^{(2)}$) that accepts the customer with a 2/3 probability and reject the customer with a 1/3 probability. In both cases, the initial state is taken to be $w_0 = 2$ and we stop the algorithm after 5000 transition steps. Note that in AAVI (and also in FVI), the expectation at step 4 of the algorithm is estimated by using a sample average of V_t values based on 1000 independent draws from the transition density.

Figure 1 shows the value function approximations obtained by the three comparison algorithms upon termination. In Table 1, we also reported the value function estimates at 5 selected states $w \in \{1,3,5,7,9\}$, where the results for FVI and AAVI are averaged over 30 independent replication runs (with the corresponding standard errors indicated in parentheses).

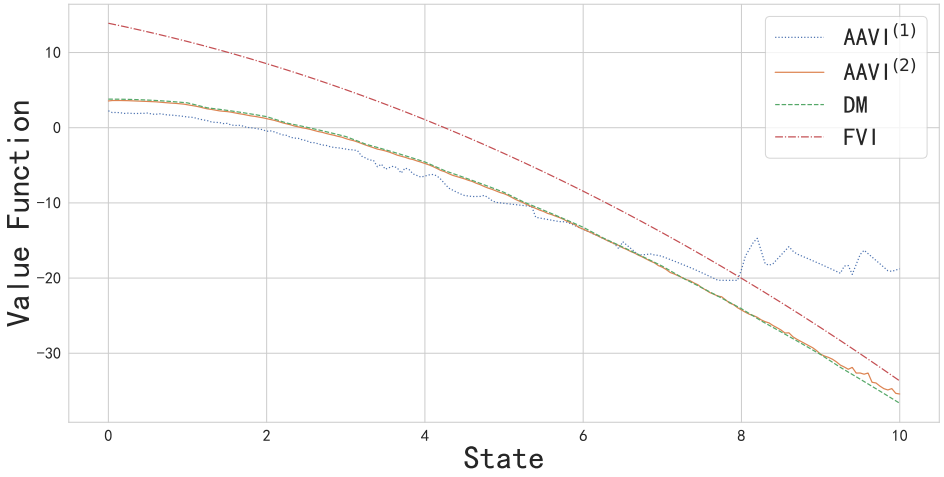

Figure 1: Value function approximators obtained by comparison algorithms

Note that since a fine discretization is used, the approximator returned by DM is very close to the true optimal value function, and hence can be used as a benchmark to gauge the performance of AAVI and FVI. From both Figure 1 and Table 1, we see that our proposed algorithm shows better overall performance than FVI. However, when compared with AAVI$^{(2)}$, the approximator generated by AAVI$^{(1)}$ shows significant performance degradation when the states are large. This is because the initial state is set small in our experiments, so that the points generated from the ε-greedy policy tend to be clustered around regions where the state bears small values. As a result, there are an insufficient number of points generated in the region where the state value is large, rendering the value function estimate less reliable over that area. In contrast, since AAVI$^{(2)}$ makes the sampled points more evenly distributed over the entire state space, it shows a much improved performance over AAVI$^{(1)}$. This indicates that the performance of AAVI could be influenced by the choice of the learning policy. In particular, if one is interested in approximating the entire value function, then the policy should be sufficiently exploratory to allow a relatively unbiased access to all parts of the state space. The typical behavior of AAVI$^{(2)}$ is also shown in Figure 2, which

plots the current value function estimates at the 5 selected states as a function of the number of algorithm iterations (time periods simulated). The figure clearly indicates the fast convergence of these estimates to their corresponding optimal values.

Table 1: Value function estimates obtained by comparison algorithms

	w =1.0	w =3.0	w =5.0	w =7.0	w =9.0
DM	3.30	-1.17	-8.60	-18.42	-30.17
FVI	11.40(0.10)	4.99(0.08)	-3.47(0.06)	-14.01(0.05)	-26.61(0.04)
AAVI$^{(1)}$	2.02(0.12)	-2.25(0.12)	-8.76(0.20)	-11.15(0.39)	-12.75(0.59)
AAVI$^{(2)}$	2.78(0.13)	-1.70(0.13)	-9.14(0.13)	-18.86(0.12)	-30.08(0.13)

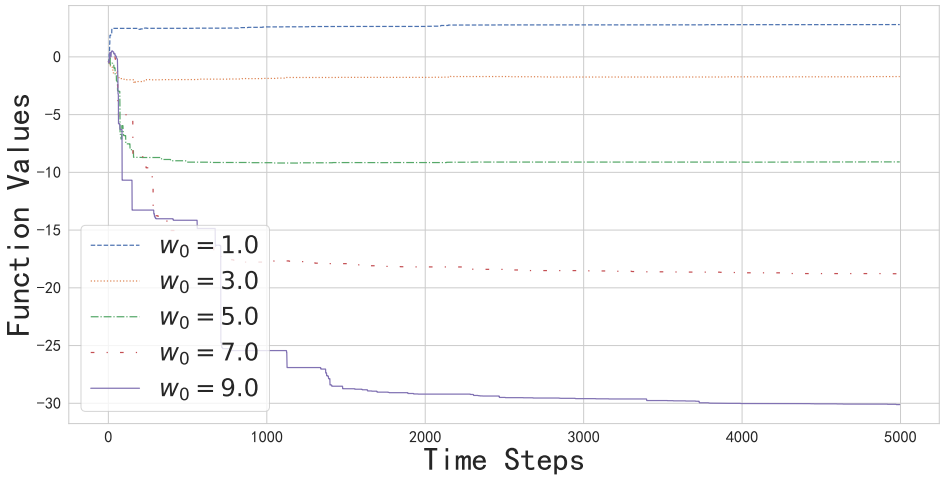

Figure 2: Convergence behavior of AAVI$^{(2)}$

4.2 Two-Dimensional Queueing Control Example

To illustrate the performance of the algorithm in higher dimensions, we consider a simple two-dimensional extension of the previous example with the underlying system consisting of two single-server queues in parallel. When a customer arrives at the system, the decision-maker can apply one of the three actions: accept entry to queue 1, accept entry to queue 2, or reject entry to either queue. The state variable is now denoted by $s := (w_1, w_2)$, which indicates the workload at queue 1 and queue 2, respectively. Thus, by using the same transition function and cost structure as described in the one-dimensional case, the Bellman equation can be formulated as follows:

$$\begin{aligned} V^*(w_1,w_2) \quad = \quad & \max\{p_1 - r(w_1) - r(w_2) + \beta \mathsf{E} V^*\left((w_1-1)^+ + s, (w_2-1)^+\right), \\ & p_2 - r(w_2) - r(w_1) + \beta \mathsf{E} V^*\left((w_1-1)^+, (w_2-1)^+ + s\right), \\ & -r(w_1) - r(w_2) + \beta V^*\left((w_1-1)^+, (w_2-1)^+\right)\}, \end{aligned}$$

where p_i $(i = 1,2)$ is the reward earned by admitting an arrival into queue i and s is the service time of the accepted arrival. In the numerical experiments, we take $p_1 = p_2 = 2$, $\beta = 0.9$, and s is uniformly distributed between 0 and 4. The upper bound on the amount of workload allowed at each queue is set to 5.

We implemented DM, FVI, and AAVI on this two-dimensional problem. The parameter setting for AAVI is basically the same as in the one-dimensional case except that the shrinking ball radius is taken to

Table 2: Value function estimates obtained by comparison algorithms

	s =(0.5,2.5)	s =(1.5,2.0)	s =(2.5,1.0)	s =(3.5,0.5)
DM	-3.13	-3.06	-3.50	-5.32
FVI	1.34(0.06)	0.76(0.03)	0.74(0.04)	-0.42(0.08)
AAVI	-3.29(0.08)	-3.16(0.08)	-3.66(0.08)	-5.16(0.47)

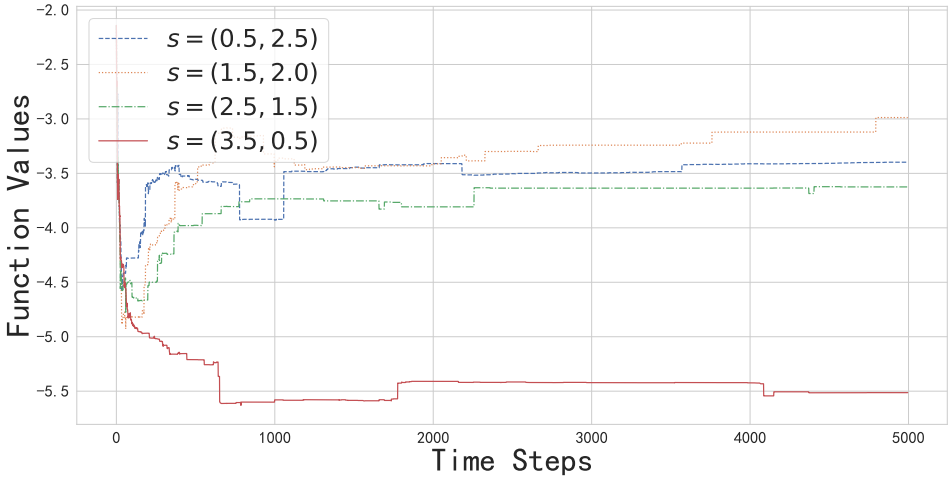

Figure 3: Convergence behavior of AAVI at different states

be $r_t = C/\ln(1+t)$, where the constant C is chosen to ensure that the shrinking ball radius is approximately 10% of the size of the state space when the algorithm terminates. In addition, a "warm-up" phase is used to speed up the algorithm performance. In particular, instead of setting $V_0 = 0$ as in Step 1 of the algorithm, similar to FVI, we have used the initial value function estimates at a set of pre-sampled states to fit the initial V_0. This step aims at providing the function approximator with the initial information necessary to reasonably predict the general trend of the value function, which is often helpful in improving the algorithm's transient behavior.

Table 2 shows the value function estimates (at selected states) obtained by AAVI using an ε-greedy policy after 5000 iterations (its convergence behavior is shown in Figure 3). Also included are the performances of DM and FVI, both are implemented using the same procedures described in Section 4.1. The results are similar to those in the one-dimensional case, with AAVI providing superior performance over FVI. We remark that although we have used DM as a benchmark algorithm, its performance relies on explicit knowledge of transition probabilities and cost functions, which prevents its application in a simulation-based environment. FVI is model-free but often requires transition samples to be generated off-line (e.g., from a simulation model) and then stored in advance. In contrast, AAVI is a fully online technique that allows value function estimates at visited states to be adaptively updated based on a single sample trajectory produced from a learning policy.

5 CONCLUSIONS

In this paper, we have proposed a novel algorithm called AAVI for solving continuous-state discrete-action MDPs in a simulation-based setting. The algorithm can be viewed as an extension of the approximate VI algorithm to the continuous-state domain. The key idea is to incorporate the shrinking ball idea from continuous SO into a RL context so that the value function estimates at a visited state can be effectively

improved by using estimates at other states collected along a single trajectory produced from a learning policy. The algorithm is fully adaptive, discretization-free, and provably convergent to the optimal value function. Preliminary simulation results and comparison studies indicate promise of the approach.

ACKNOWLEDGMENTS

This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grants 71901003 and 71720107003.

REFERENCES

Barton, R. R. 2009. "Simulation Optimization Using Metamodels". In *Proceedings of the 2009 Winter Simulation Conference*, edited by M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin, and R. G. Ingalls, 230–238. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Baumert, S., and R. L. Smith. 2002. "Pure Random Search for Noisy Objective Functions". Technical Report, Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor.

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

Bertsekas, D. P. 2015. *Dynamic Programming and Optimal Control,*. 4th ed, Volume II. Belmont, MA: Athena Scientific.

Busoniu, L., R. Babuska, B. De Schutter, and D. Ernst. 2010. *Reinforcement Learning and Dynamic Programming using Function Approximators*, Volume 39. Boca Raton: CRC Press, Inc.

Chandramohan, S., M. Geist, and O. Pietquin. 2010. "Optimizing Spoken Dialogue Management with Fitted Value Iteration". In *Eleventh Annual Conference of the International Speech Communication Association*. Aug 28^{th}-31^{st}, Florence, Italy.

Chang, H., J. Hu, M. Fu, and S. Marcus. 2013. *Simulation-Based Algorithms for Markov Decision Processes*. 2nd ed. NY: Springer.

Ernst, D., P. Geurts, and L. Wehenkel. 2005. "Tree-Based Batch Mode Reinforcement Learning". *Journal of Machine Learning Research* 6:503–556.

Ernst, D., M. Glavic, P. Geurts, and L. Wehenkel. 2005. "Approximate Value Iteration in the Reinforcement Learning Context. Application to Electrical Power System Control.". *International Journal of Emerging Electric Power Systems* 3(1):1066.

Fan, Q., and J. Hu. 2018. "Surrogate-Based Promising Area Search for Lipschitz Continuous Simulation Optimization". *INFORMS Journal on Computing* 30:677–693.

Gordon, G. J. 1995. "Stable Function Approximation in Dynamic Programming". In *Machine Learning Proceedings 1995*, 261–268. Amsterdam: Elsevier.

Gosavi, A. 2014. *Simulation-Based Optimization:Parametric Optimization Techniques and Reinforcement Learning*. second ed. New York, NY: Springer.

Gutmann, H.-M. 2001. "A Radial Basis Function Method for Global Optimization". *Journal of Global Optimization* 19:201–227.

Hu, J., X. Yang, J. Hu, and Y. Peng. 2020. "Adaptive Q-learning for Markov Decision Processes with Continuous State Spaces". *working paper*.

Jones, D., M. Schonlau, and W. Welch. 1998. "Efficient Global Optimization of Expensive Black-box Functions". *Journal of Global Optimization* 13:455–492.

Kleijnen, J. P. C. 2009. "Kriging Metamodeling in Simulation: A Review". *European Journal of Operational Research* 192(3):707–716.

Mbuwir, B. V., F. Ruelens, F. Spiessens, and G. Deconinck. 2017. "Battery Energy Management in a Microgrid Using Batch Reinforcement Learning". *Energies* 10(11):1846.

Munos, R., and C. Szepesvári. 2008. "Finite-Time Bounds for Fitted Value Iteration". *Journal of Machine Learning Research* 9(May):815–857.

Ormoneit, D., and S. Sen. 2002. "Kernel-Based Reinforcement Learning". *Machine Learning* 49(2-3):161–178.

Powell, W. 2007. *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. NJ, USA: Wiley-Interscience.

Rust, J. 1997. "Using Randomization to Break the Curse of Dimensionality". *Econometrica: Journal of the Econometric Society*:487–516.

Sacks, J., W. J. Welch, T. J. Mitchell, and H. P. Wynn. 1989. "Design and Analysis of Computer Experiments". *Statistical Science*:409–423.

Stachurski, J. 2008. "Continuous State Dynamic Programming via Nonexpansive Approximation". *Computational Economics* 31(2):141–160.

Sutton, R., and A. G. Barto. 1998. *Reinforcement Learning*. Cambridge, Massachusetts: The MIT Press.

Tsitsiklis, J. N. 1994. "Asynchronous Stochastic Approximation and Q-Learning". *Machine learning* 16(3):185–202.

Wang, B., and J. Hu. 2018. "Some Monotonicity Results for Stochastic Kriging Metamodels in Sequential Settings". *INFORMS Journal on Computing* 30:278–294.

AUTHOR BIOGRAPHIES

XIANGYU YANG is a PhD candidate in the School of Management at Fudan University. He holds a bachelor's degree in Engineering Management from Shandong University. His research interests include optimization and simulation-based MDPs and financial statistics. His e-mail address is xyyang19@fudan.edu.cn.

JIAQIAO HU is an Associate Professor in the Department of Applied Mathematics and Statistics at the State University of New York, Stony Brook. He received the B.S. degree in automation from Shanghai Jiao Tong University, the M.S. degree in applied mathematics from the University of Maryland, Baltimore County, and the Ph.D. degree in electrical engineering from the University of Maryland, College Park. His research interests include Markov decision processes, applied probability, and simulation optimization. His e-mail address is jqhu@ams.sunysb.edu.

JIAN-QIANG HU is the Hongyi Professor of Management Science in School of Management, Fudan University. He was an Associate Professor with the Department of Mechanical Engineering and the Division of Systems Engineering at Boston University before joining Fudan University. His research interests include discrete-event stochastic systems, simulation, stochastic optimization, with applications towards supply chain management, financial engineering, and healthcare. His e-mail addresses is hujq@fudan.edu.cn.

YIJIE PENG is an Assistant Professor in the Department of Management Science and Information Systems at Peking University, Beijing, China. His research interests include stochastic modeling and analysis, simulation optimization, machine learning, data analytics, and healthcare. He is a member of INFORMS and IEEE, and serves as an Associate Editor of the Asia-Pacific Journal of Operational Research and the Conference Editorial Board of the IEEE Control Systems Society. His e-mail address is pengyijie@pku.edu.cn.