Proceedings of the
44th IEEE Conference on Decision and Control, and
the European Control Conference 2005
Seville, Spain, December 12-15, 2005

A Two-Timescale Simulation-Based Gradient Algorithm for Weighted Cost Markov Decision Processes

Ying He, Michael C. Fu, and Steven I. Marcus

Abstract— **We develop a novel two-timescale simulation-based gradient algorithm for weighted cost Markov Decision Process (MDP) problems, illustrate the effectiveness of this algorithm by carrying out numerical experiments on a parking example, and compare the algorithm with two other algorithms in the literature.**

I. Introduction

In this paper, we develop a two-timescale simulation-based gradient algorithm for infinite horizon weighted cost Markov Decision Process (MDP) problems with weighted expected total cost criterion, where the expected total cost is defined over a known probability distribution on the initial state. This problem, a generalization of the stochastic short problem, is first discussed by Marbach [1]. In our setting, in order to deal with "curse of dimensionality" and "curse of modeling" difficulties associated with MDPs for large and complicated systems, we assume that a policy can be parameterized, in such way the weighted cost problem can be transformed to a problem of finding the optimal parameters minimizing only the weighted expected cost, and thus can be solved by simulation-based optimization algorithms such as gradient-based stochastic approximation.

Our two-timescale simulation-based gradient algorithm is developed by combining two simulation-based gradient algorithms: one that updates parameters at regenerative points (as in average cost MDPs) and the other that updates parameters at each time step [1]. These algorithms require the state be fully observable and the transition probabilities known explicitly (as opposed to having only sample paths available). In the regenerative-update algorithm, the length of each regenerative cycle is unknown, which can lead to infrequent update of parameters. On the other hand, if parameters are updated at each time epoch, parameters may change too frequently, which may not be allowed in a real system. The proposed two-timescale algorithm, by updating parameters at a sequence of specially designed time epochs, can avoid the situations of infrequent updates with unknown regenerative points and too much disturbance to system due to too frequent updates.

Several authors have studied two (or multiple)-timescale simulation-based algorithms. In [2], Borkar claimed that one important instance of two-timescale is the infinitesimal perturbation analysis based stochastic approximation, which requires averaging over regeneration periods. Bhatnagar et al. [3] [4] [5] [6] [7] proposed several Kiefer-Wolfowitz-type two-timescale stochastic approximation algorithms for average cost problems. For Markov Decision Processes, Konda et al. [8] [9] and Bhatnagar et al. [10] also proposed actor-critic algorithms and cast them as two-timescale algorithms for convergence proofs. Since our setting requires the state fully observable and transition probabilities known explicitly, the applicability of our two timescale algorithm is more restrictive than most of these other two timescale algorithms.

The effectiveness of the two-timescale simulation-based gradient algorithm for weighted cost MDP problems is demonstrated by numerical experiments on a parking case study.

This work was supported in part by the National Science Foundation under Grant DMI-0323220 and in part by the Air Force Office of Scientific Research under Grant FA95500410210.

Dr. Y. He is with Lister Hill National Center for Biomedical Communications, National Library of Medicine, Bethesda, MD 20894, USA. yihe@mail.nih.gov

Dr. S. I. Marcus and Dr. M. C. Fu are with the faculty in the Institute for Systems Research, University of Maryland, College Park, MD 20742, USA. marcus@isr.umd.edu and mfu@isr.umd.edu

II. Markov Decision Processes

A Markov Decision Process is a framework containing states, actions, costs, probabilities, and the decision horizon for the problem of optimizing a stochastic discrete-time dynamic system. The dynamic system equation is $x_{t+1} = f_t(x_t, u_t, w_t)$, $t = 0, 1, \ldots, T-1$, where t indexes a time epoch; x_t is the state of the system; u_t is the action to be chosen at time t; w_t is a random disturbance which is characterized by a conditional probability distribution $P(\cdot \mid x_t, u_t)$; and T is the decision horizon. We denote the set of possible system states by S and the set of allowable actions in state $i \in S$ by $U(i)$. We assume S, $U(i)$, and $P(\cdot \mid x_t, u_t)$ do not vary with t. We further assume that the sets S and $U(i)$ are finite sets, where S consists of n states denoted by $0, 1, \ldots, n-1$.

If, at some time t, the system is in state $x_t = i$ and action $u_t = u$ is applied, we incur a stage cost $g(x_t, u_t) = g(i, u)$, assumed bounded, and the system moves to state $x_{t+1} = j$ with probability $p_{ij}(u) = P(x_{t+1} = j \mid x_t = i, u_t = u)$, which may be given a priori or calculated from the system equation and the known probability distribution of the random disturbance.

A. Stochastic Shortest Path Problem

In the stochastic shortest path problem, it is assumed that there is a special cost-free termination state n in the system and the system remains there at no further cost once it reaches that state. The objective is to minimize over all policies $\pi = \{\mu_0, \mu_1, \ldots\}$ with $\mu_t : S \to U, \mu_t(i) \in U(i)$ for i and t, the total expected cost, $J_\pi(i) = E\left\{\sum_{t=0}^{T-1} g(x_t, \mu_t(x_t)) \mid x_0 = i\right\}$, where $T = \min\{t > 0 \mid x_t = i^*\}$. A stationary policy is an admissible policy of the form $\pi = \{\mu, \mu, \ldots\}$; we denote it by μ_∞.

B. Weighted Cost Problems

The objective is to minimize the mean cost over initial states, knowing that the initial state x_0 is equal to a specific state $i \in S$ with probability ξ_i. If we denote the cost to go from an initial state x_0 with policy π as $J_\pi(x_0)$, the mean cost to go over initial states with policy π is $E[J_\pi(x_0)] = \sum_{i \in S} \xi_i J_\pi(i)$. In the sense that ξ_i acts like a kind of weight, we call the mean cost to go over initial states a *weighted cost to go* and denote it by

$$\chi_\pi = \sum_{i \in S} \xi_i J_\pi(i), \tag{1}$$

where $\xi_i \geq 0$, $\sum_{i \in S} \xi_i = 1$. The corresponding problem of minimizing χ_π is called *the weighted cost problem*.

0-7803-9568-9/05/$20.00 ©2005 IEEE

Here, we assume that the weighted cost problem has an infinite horizon. We also assume that there exists a termination cost-free state i^* in the system and the system remains there at no further cost once it reaches that state, as in the stochastic shortest path problem. The objective is to minimize (1) over all policies $\pi = \{\mu_0, \mu_1, \ldots\}$, where $J_\pi(i) = E\left\{\sum_{t=0}^{T-1} g(x_t, \mu_t(x_t)) \mid x_0 = i\right\}$, $T = \min\{t > 0 \mid x_t = i^*\}$.

If π is a stationary policy with the form $\pi = \{\mu, \mu, \ldots\}$, the corresponding weighted cost and expected total cost from state i are denoted as χ_μ and $J_\mu(i)$, respectively.

If a randomized decision rule μ is considered, which specifies a probability distribution $q_{\mu(i)}(u)$ on the set of actions, the stage cost and transition probabilities become $g(i, \mu(i)) = \sum_{u \in U(i)} g(i,u) q_{\mu(i)}(u)$, and $p_{ij}(\mu(i)) = \sum_{u \in U(i)} p_{ij}(u) q_{\mu(i)}(u)$, respectively.

In order to solve this problem, we need to derive its optimality equations. Note that the policy minimizing $J_\mu(i)$ in the corresponding stochastic shortest path problem also minimizes χ_μ in the weighted cost problem, because $\chi^* = \sum \xi_i J^*(i) \leq \sum \xi_i J_\pi(i) = \chi_\pi$, as $J^*(i) \leq J_\pi(i)$. Hence, policy iteration, value iteration, and all simulation-based dynamic programming algorithms for the stochastic shortest path problem can be used for the weighted cost problem. If the control space is very large, the search space would accordingly be large, which requires very long computation time. One way to deal with this difficulty is to parameterize the policy.

If we parameterize the policy by a parameter vector $\theta \in R^p$, the parameterized stationary policy is $\pi(\theta) = \{\mu, \mu, \ldots\}$, with $\mu : S \times R^p \to U$, and the corresponding decision rule is $\mu(i, \theta)$, with its dependence on θ assumed known.

Under a randomized policy $\mu(i, \theta)$ characterized by $q_{\mu(i,\theta)}(u)$, the transition probabilities are denoted by $p_{ij}(\theta)$, the one-stage cost at state i is denoted by $g_i(\theta)$ and

$$\begin{aligned} p_{ij}(\theta) &= \textstyle\sum_{u \in U(i)} p_{ij}(u) q_{\mu(i,\theta)}(u), \\ g_i(\theta) &= \textstyle\sum_{u \in U(i)} g(i,u) q_{\mu(i,\theta)}(u). \end{aligned} \tag{2}$$

Hence, under policy $\mu(i, \theta)$, the Markov Decision Process we defined earlier degenerates to a Markov process defined by transition probabilities $p_{ij}(\theta)$ and one-stage costs $g_i(\theta)$, which we call a Markov Cost Process (MCP) depending on θ. The termination state is i^*, with $p_{i^*i^*}(\theta) = 1$ and $g_{i^*}(\theta) = 0$.

Furthermore, the original problem of finding an optimal policy for an MDP becomes a problem of finding the optimal θ minimizing $\chi(\theta)$, where

$$\chi(\theta) = \sum_{i \in S} \xi_i J_i(\theta),$$

$$J_i(\theta) := J_{\pi(\theta)}(i) = E\left\{\sum_{t=0}^{T-1} g_{i_t}(\theta) \mid i_0 = i\right\}, \tag{3}$$

and $T = \min\{t > 0 \mid i_t = i^*\}$.

In order for gradient estimation techniques to be applicable, the following assumptions on $p_{ij}(\theta)$ and $g_i(\theta)$ are necessary.

Assumption 1 (MCP Parameterization): For all $i, j \in S$, the transition probability $p_{ij}(\theta)$, and the stage cost $g_i(\theta)$ are bounded, twice differentiable, and have bounded first and second derivatives. Furthermore, for all states $i, j \in S$, we have $\nabla p_{ij}(\theta) = p_{ij}(\theta) f_{ij}(\theta)$, where the function $f_{ij}(\theta)$ is bounded and differentiable, with bounded first derivative.

Let $P(\theta)$ be the transition matrix with entries $p_{ij}(\theta)$, let $\mathcal{P} = \{P(\theta) \mid \theta \in \mathcal{R}^K\}$ be the set of all possible transition matrices, and let $\overline{\mathcal{P}}$ be its closure. It can be proved that every element $P \in \overline{\mathcal{P}}$ is a stochastic matrix (see Lemma 1 in [1]).

Assumption 2 (MCP Termination): There exists a state $i^* \in S$, such that, for every parameter vector $\theta \in \mathcal{R}^K$, we have $g_{i^*}(\theta) = 0$ and $p_{i^*i^*}(\theta) = 1$, and for every state $i \in S$ and every transition matrix $P(\cdot) \in \overline{\mathcal{P}}$, we have $p_{ii^*}^N > 0$, where N is the number of states in the state space S.
Note that $p_{ii^*}^N > 0$ only means that state i^* is reachable from every other state.

III. Two-Timescale Simulation-Based Gradient Algorithm

In this section, we develop a two-timescale simulation-based gradient algorithm for weighted cost problems using gradient estimation techniques.

We begin with defining a new Markov cost process with transition probabilities and one-stage costs as in [1]:

$$\begin{aligned} p_{\xi,ij}(\theta) &= \begin{cases} p_{ij}(\theta) & \text{if } i \neq i^*; \\ \xi_j & \text{if } i = i^*, \end{cases} \\ g_{\xi,i}(\theta) &= g_i(\theta). \end{aligned} \tag{4}$$

Note that the new Markov cost process is a renewal process and $\chi(\theta)$ is equal to the expected cumulative cost over a regenerative cycle. By using renewal theory, the following expressions for $\chi(\theta)$ and its gradient can be obtained [1]:

$$\chi(\theta) = E_{\xi,\theta}[T] \sum_{i \in S} \pi_{\xi,i}(\theta) g_{\xi,i}(\theta), \tag{5}$$

$$\nabla\chi(\theta) = E_{\xi,\theta}[T] \sum_{i \in S} (\nabla\pi_{\xi,i}(\theta) g_{\xi,i}(\theta) + \pi_{\xi,i}(\theta) \nabla g_{\xi,i}(\theta)), \tag{6}$$

where $E_{\xi,\theta}[T]$ is the mean recurrence time and $\pi_{\xi,i}(\theta)$ is the steady state probability distribution of being in state $i \in S$. However, for a problem with a large state space, it is generally infeasible to compute this gradient exactly, since this requires to compute, for every $i \in S$, the steady state probability $\pi_{\xi,i}(\theta)$ and its gradient $\nabla\pi_{\xi,i}(\theta)$. One method to deal with this difficulty is to develop a simulation-based estimator of $\nabla\chi(\theta)$ by techniques such as perturbation analysis.

Before we present a particular gradient estimator for $\nabla\chi(\theta)$, let's discuss why we consider two-timescale simulation-based algorithms and how we develop such algorithms.

Let $(i_1, i_2, \ldots)$ be a sample path of the renewal process, t_m be the mth visit to the recurrent state i^*, and $i_{t_m}, i_{t_m+1}, \ldots, i_{t_{m+1}-1}$ be the mth regenerative cycle. Suppose $\hat{F}(\theta)$ is an unbiased estimator of $\nabla\chi(\theta)$, a simulation-based gradient algorithm based on this estimate is:

$$\theta_{m+1} = \theta_m + \alpha_m \hat{F}_m(\theta_m), \tag{7}$$

where $\hat{F}(\theta_m)$ is a sample estimate of $\nabla\chi(\theta_m)$ and the step sizes α_m are deterministic, nonnegative, and satisfying $\sum_{m=1}^{\infty} \alpha_m = \infty$ and $\sum_{m=1}^{\infty} \alpha_m^2 < \infty$. This algorithm, which we call the regenerative-update simulation-based algorithm, updates at visits to the regenerative state i^*.

And if $\hat{F}_m(\theta_m)$ can be reformulated as:

$$\hat{F}_m(\theta) = \sum_{k=t_m}^{t_{m+1}-1} \hat{R}_k(\theta),$$

another simulation-based gradient algorithm is given by:

$$\theta_{k+1} = \theta_k + \alpha_k \hat{R}_k(\theta_k), \tag{8}$$

where the step sizes α_k satisfy the same conditions as before. We call such an algorithm the every-update simulation-based gradient algorithm.

Note that the length of each regenerative cycle is unknown, and maybe very long, which can lead to infrequent updates of θ for the regenerative-update simulation-based algorithm. On the other hand, if θ is updated at each time epoch, as in the every-update simulation-based gradient algorithm, θ may change too frequently which may not be allowed in a real system. Hence, we propose a general two-timescale simulation-based gradient algorithm as follows:

$$\theta_{l+1} = \theta_l + \sum_{k=n_l}^{n_{l+1}-1} \alpha_k \hat{R}_k(\theta_l), \tag{9}$$

where

$$n_{l+1} = \min\{j > n_l | \sum_{k=n_l}^{j-1} \alpha_k \geq \beta_l\}, \tag{10}$$

and we assume the following on the two-timescale step sizes α_k and β_l:

Assumption 3: The step sizes α_k and β_k are deterministic, nonnegative and satisfy $\sum_{k=1}^{\infty} \alpha_k = \infty$, $\sum_{k=1}^{\infty} \alpha_k^2 < \infty$, $\sum_{k=1}^{\infty} \beta_k = \infty$, $\sum_{k=1}^{\infty} \beta_k^2 < \infty$, $\alpha_k/\alpha_{k+1} \to 1, \beta_k/\beta_{k+1} \to 1, \alpha_k = o(\beta_k)$.

Next, we discuss a particular gradient estimator for $\chi(\theta)$.

A. A Modified Gradient Estimator and its Decomposition

In this part, we review Marbach's results [1] on the gradient of $\chi(\theta)$ and the gradient estimator, and present a modified gradient estimator and a decomposition of this modified gradient estimator.

By using perturbation analysis, Marbach [1] obtained the following expressions for $\nabla\chi(\theta)$, the gradient of $\chi(\theta)$:

$$\nabla\chi(\theta) = E_{\xi,\theta}[T] \sum_{i\in S} \pi_{\xi,i}(\theta)(\nabla g_i(\theta) + \sum_{j\in S} \nabla p_{ij}(\theta) J_j(\theta)), \tag{11}$$

based on (6). Note that $\nabla\chi(\theta)$ in Eq. (11) can be rewritten as [1]:

$$\begin{aligned}\nabla\chi(\theta) = \quad & E_{\xi,\theta}[T] \textstyle\sum_{i\in S} \pi_{\xi,i}(\theta) \cdot \\ & \left(\nabla g_i(\theta) + \textstyle\sum_{j\in S_i(\theta)} p_{ij}(\theta) \left(\frac{\nabla p_{ij}(\theta)}{p_{ij}(\theta)} J_j(\theta)\right)\right),\end{aligned} \tag{12}$$

where $S_i(\theta) = \{j \in S \mid \nabla p_{ij}(\theta) \neq 0\}$. Let t_m be the time epoch that state i^* is visited for the mth time and the sequence $\{i_{t_m}, i_{t_m+1}, \ldots, i_{t_{m+1}-1}\}$ be the mth regenerative cycle.

Based on Eq. (12), Marbach put forward the following estimate for $\nabla\chi(\theta)$:

$$F_m(\theta) = \sum_{n=t_m}^{t_{m+1}-1} \left(\tilde{J}_{i_n}(\theta) \frac{\nabla p_{i_{n-1} i_n}(\theta)}{p_{i_{n-1} i_n}(\theta)} + \nabla g_{i_n}(\theta)\right), \tag{13}$$

where

$$\tilde{J}_{i_n}(\theta) = \begin{cases} \sum_{k=n}^{t_{m+1}-1} g_{i_k}(\theta) & \text{if } t_m < n \leq t_{m+1}-1, \\ 0 & \text{if } n = t_m. \end{cases} \tag{14}$$

Alternatively, we propose the following:

$$\bar{F}_m(\theta) = \sum_{n=t_m}^{t_{m+1}-1} \left(\tilde{J}_{i_{n+1}}(\theta) \frac{\nabla p_{i_n i_{n+1}}(\theta)}{p_{i_n i_{n+1}}(\theta)} + \nabla g_{i_n}(\theta)\right), \tag{15}$$

where

$$\tilde{J}_{i_n}(\theta) = \begin{cases} \sum_{k=n}^{t_{m+1}-1} g_{i_k}(\theta), & \text{if } t_m < n \leq t_{m+1}-1, \\ \sum_{k=t_m}^{t_{m+1}-1} g_{i_k}(\theta), & \text{if } n = t_{m+1}. \end{cases} \tag{16}$$

Comparing the two estimators with Eq. (12), we argue that the corrected estimator corresponds better to Eq. (12), since i and j in Eq. (12) are consistent with i_n and i_{n+1} in Eq. (15), respectively, and $\tilde{J}_{i_{t_{m+1}}}(\theta)$ in Eq. (16) matches the definition of $J_{i^*}(\theta)$, which is the cumulative cost over a regenerative cycle. Since the cumulative cost over the next regenerative cycle is not available at t_{m+1}, in our estimator the current regenerative cycle from t_m to $t_{m+1}-1$ is used for $\tilde{J}_{i_{t_{m+1}}}(\theta)$ estimating, instead of the next regenerative cycle from t_{m+1} to $t_{m+2}-1$ suggested by Eq. (12). Using the corrected estimator, we can otherwise follow proof of Proposition 4 in [1] to prove that our estimator (15) is an unbiased estimate of the gradient.

Note that we can decompose (15) as

$$\begin{aligned}
&\bar{F}_m(\theta) = \textstyle\sum_{n=t_m}^{t_{m+1}-1} \left(\tilde{J}_{i_{n+1}}(\theta) \frac{\nabla p_{i_n i_{n+1}}(\theta)}{p_{i_n i_{n+1}}(\theta)} + \nabla g_{i_n}(\theta)\right) \\
&= \textstyle\sum_{n=t_m}^{t_{m+1}-2} \frac{\nabla p_{i_n i_{n+1}}(\theta)}{p_{i_n i_{n+1}}(\theta)} \sum_{k=n+1}^{t_{m+1}-1} g_{i_k}(\theta) \\
&\quad + \frac{\nabla p_{i_{t_{m+1}-1} i_{t_{m+1}}}(\theta)}{p_{i_{t_{m+1}-1} i_{t_{m+1}}}(\theta)} \tilde{J}_{i_{t_{m+1}}}(\theta) + \textstyle\sum_{n=t_m}^{t_{m+1}-1} \nabla g_{i_n}(\theta) \\
&= \textstyle\sum_{h=t_m+1}^{t_{m+1}-1} \frac{\nabla p_{i_{h-1} i_h}(\theta)}{p_{i_{h-1} i_h}(\theta)} \sum_{k=h}^{t_{m+1}-1} g_{i_k}(\theta) \\
&\quad + \frac{\nabla p_{i_{t_{m+1}-1} i_{t_{m+1}}}(\theta)}{p_{i_{t_{m+1}-1} i_{t_{m+1}}}(\theta)} \tilde{J}_{i_{t_{m+1}}}(\theta) + \textstyle\sum_{n=t_m}^{t_{m+1}-1} \nabla g_{i_n}(\theta) \\
&= \textstyle\sum_{k=t_m+1}^{t_{m+1}-1} g_{i_k}(\theta) \sum_{h=t_m+1}^{k} \frac{\nabla p_{i_{h-1} i_h}(\theta)}{p_{i_{h-1} i_h}(\theta)} \\
&\quad + \frac{\nabla p_{i_{t_{m+1}-1} i_{t_{m+1}}}(\theta)}{p_{i_{t_{m+1}-1} i_{t_{m+1}}}(\theta)} \tilde{J}_{i_{t_{m+1}}}(\theta) + \textstyle\sum_{n=t_m}^{t_{m+1}-1} \nabla g_{i_n}(\theta) \\
&= \textstyle\sum_{k=t_m+1}^{t_{m+1}-1} \left(\nabla g_{i_k}(\theta) + g_{i_k}(\theta) \sum_{h=t_m+1}^{k} \frac{\nabla p_{i_{h-1} i_h}(\theta)}{p_{i_{h-1} i_h}(\theta)}\right) \\
&\quad + \frac{\nabla p_{i_{t_{m+1}-1} i_{t_{m+1}}}(\theta)}{p_{i_{t_{m+1}-1} i_{t_{m+1}}}(\theta)} \sum_{h=t_m}^{t_{m+1}-1} g_{i_h}(\theta) \\
&= \textstyle\sum_{k=t_m+1}^{t_{m+1}-1} \Big(\nabla g_{i_k}(\theta) + g_{i_k}(\theta) z_k \\
&\quad + \frac{\nabla p_{i_k i_{k+1}}(\theta)}{p_{i_k i_{k+1}}(\theta)} (L_k + g_{i_k}(\theta)) I_{\{k=t_{m+1}-1\}} \Big)
\end{aligned} \tag{17}$$

where $I_{\{\cdot\}}$ is the indicator function,

$$z_{k+1} = \begin{cases} 0, & \text{if} \quad i_{k+1} = i^*; \\ z_k + \frac{\nabla p_{i_k i_{k+1}}(\theta)}{p_{i_k i_{k+1}}(\theta)} & \text{otherwise}, \end{cases} \tag{18}$$

and

$$L_{k+1} = \begin{cases} 0, & \text{if} \quad i_{k+1} = i^*; \\ L_k + g_{i_k}(\theta), & \text{otherwise}. \end{cases} \tag{19}$$

Note that at $k = t_m$, both $\nabla g_{i_k}(\theta)$ and $g_{i_k}(\theta)$ are zero. Hence, $\hat{R}_k(\theta)$ in Equations (8) (9) can be substituted by

$$\begin{aligned}&R(x_k, \theta) = \\ &\nabla g_{i_k}(\theta) + g_{i_k}(\theta) z_k + \frac{\nabla p_{i_k i_{k+1}}(\theta)}{p_{i_k i_{k+1}}(\theta)} (L_k + g_{i_k}(\theta)) I_{\{i_{k+1}=i^*\}},\end{aligned} \tag{20}$$

where $x_k = (i_k, z_k, L_k)$.

B. Two-Timescale Simulation-Based Gradient Algorithm

Now we present our special two-timescale simulation-based gradient algorithm, which updates at some given time epoch n_l, defined by two-timescale step sizes α_k and β_l. In this algorithm,

the parameter θ_l is updated as follows:

$$\theta_{l+1} = \theta_l + \sum_{k=n_l}^{n_{l+1}-1} \alpha_k \Big(\nabla g_{i_k}(\theta_l) + g_{i_k}(\theta_l) z_k + \frac{\nabla p_{i_k i_{k+1}}(\theta_l)}{p_{i_k i_{k+1}}(\theta_l)} (L_k + g_{i_k}(\theta_l)) I_{\{i_{k+1}=i^*\}} \Big), \tag{21}$$

where n_l satisfies (10), and z_k and L_k are given by (18) and (19), respectively. Note that z_k and L_k are updated at every time k, which is a faster timescale; θ_l is updated at time n_l, which is a slower timescale.

To prove its convergence, we make additional assumptions as in [1].

Assumption 4: The step sizes α_k are non-increasing. Furthermore, there exist a positive integer p and a positive scalar A such that $\sum_{k=n}^{n+t}(\alpha_n - \alpha_k) \leq At^p\alpha_n^2$, for all positive integers n and t.

Assumption 5 (MCP Strong Termination): There exist a state $i^* \in S$ and a positive integer N_0, such that, for every parameter vector $\theta \in \mathcal{R}^K$, we have $g_{i^*}(\theta) = 0$, and $P_{i^*i^*}(\theta) = 1$, and, for every state $i \in S$ and every collection $\{P_1, \ldots, P_{N_0}\}$ of N_0 matrices in the set $\overline{\mathcal{P}}$, we have $Q_{ii^*} > 0$, where the matrix Q is given by $Q = P_1 \cdots P_{N_0}$.

We have the following convergence result for the two-timescale simulation-based gradient algorithm.

Proporsition 1: Let Assumption 1, 3, 4, and 5 hold, and let θ_l be the sequence of parameter vector generated by the two-timescale simulation-based gradient algorithm (21). Then, $\chi(\theta_l)$ converges and

$$\lim_{l\to\infty} \| \nabla\chi(\theta_l) \| = 0$$

with probability 1.

Proof: See [11].

IV. Parking Problem

This case study involves a well-known academic example that has been used in [12] to demonstrate the approximate policy iteration method. Here we adopt this example to illustrate the simulation-based gradient algorithms presented in the last two sections.

A driver is looking for a low-priced parking space on the way to his destination. The parking area contains N spaces. The driver starts at space N and crosses the parking spaces from space s to space $s-1$, $s = N, N-1, \ldots, 1$. The destination is parking space 0. Each parking space is empty with probability p independently of whether other parking spaces are empty or not. The driver can spot whether a parking space is empty only when he reaches it, and then, if it is empty, he makes a decision whether or not to park in that space. If he parks in space s, $s = N, N-1, \ldots, 1$, he incurs a cost $c(s) > 0$. If he reaches the destination without having parked, he must park in the destination's garage, which is expensive, and costs $C > 0$. The objective is to find the optimal parking policy.

We formulate this problem as a weighted cost MDP problem with termination state i^*, which is similar to the stochastic shortest path formulation in [12]. In addition to the termination state, state 0 corresponds to reaching the expensive garage, state (s, F) $(s = 1, \ldots, N)$ corresponds to space s being empty ("free"), and state $(s, \bar{F})$ $(s = 1, \ldots, N)$ corresponds to space s being not empty ("not free"). Two possible actions, u_p and u_n, represent the actions "park" and "do not park", respectively.

An MDP formulation for this weighted cost MDP problem follows.

States: $i \in S = \{0, i^*, (s, F), (s, \bar{F})\}$, for all s.

Actions:

$$u(i) = U(i) = \begin{cases} \{u_n, u_p\}, & i = (s, F), \text{ for all } s. \\ \{u_n\}, & i = (s, \bar{F}), \text{ for all } s. \\ \{u_p\}, & i = 0, \\ \emptyset, & i = i^*. \end{cases}$$

Transition probabilities:

$$\begin{aligned} p_{(s,F)i^*}(u_p) &= 1, \quad \text{for all } s, \\ p_{(1,j)0}(u_n) &= 1, \quad j \in \{F, \bar{F}\}, \\ p_{(s,j)(s-1,F)}(u_n) &= p, \quad s = 2, \ldots, N, \quad j \in \{F, \bar{F}\}, \\ p_{(s,j)(s-1,\bar{F})}(u_n) &= 1-p, \quad s = 2, \ldots, N, \quad j \in \{F, \bar{F}\}, \\ p_{ii^*}(u_p) &= 1, \quad i \in \{0, i^*\}, \quad u \in U(i). \end{aligned}$$

One-stage cost:

$$\begin{aligned} g((s, F), u_p) &= c(s), \quad \text{for all } s, \\ g((s, j), u_n) &= 0, \quad \text{for all } s, \quad j \in \{F, \bar{F}\}, \\ g(0, u_p) &= C, \\ g(i^*, u) &= 0, \quad u = \emptyset. \end{aligned}$$

Objective function:

The objective is to minimize over all policies $\pi = \{\mu_0, \mu_1, \ldots\}$ with $\mu_t : S \to U, \mu_t(i) \in U(i)$ for i and t, the weighted total cost from state i,

$$\chi_\pi = \sum_{i\in S} \xi_i J_\pi(i), \tag{22}$$

where

$$J_\pi(i) = E\left\{ \sum_{t=0}^{T-1} g(x_t, \mu_t(x_t)) \mid x_0 = i \right\}, . \tag{23}$$

An optimal policy of this weighted cost MDP problem has the form:

$$\mu^*(s) = \begin{cases} u_p, & \text{if space } s \text{ is free and } c(s) \leq J^*(s-1), \\ u_n, & \text{otherwise.} \end{cases}$$

Furthermore, if $c(s)$ is monotonically increasing in s, there is an integer s^* such that it is optimal to park at space s if and only if s is free and $s \leq s^*$, since $J^*(s)$ is monotonically nonincreasing in s [12]. Thus, the optimal policy is a threshold policy, i.e., to park at the first available space after a threshold is reached.

Here we use randomized threshold policy where its parameterized version is as follows(see [11] for details):

$$\begin{aligned} q_{\mu((s,F),\theta)}(u_n) &= \frac{1}{1+\exp(\theta - s)} \\ q_{\mu((s,F),\theta)}(u_p) &= 1 - q_{\mu((s,F),\theta)}(u_n), \end{aligned}$$

where θ is the threshold and the problem is converted to finding the optimal θ.

Using Eq. (2), we can also construct the transition probabilities and the corresponding one-stage costs under the randomized policy. We omit the details due to space consideration. With parameterized policy, the objective function turns to be: $\chi(\theta) = \xi_1 J_{(N,F)}(\theta) + \xi_2 J_{(N,\bar{F})}(\theta)$, where $\xi_1 = p$, $\xi_2 = 1-p$, and $J_i(\theta)$ is defined as in Eq. (3).

Now, given a fixed $\bar{\theta}$, the state sequence $i_0(\bar{\theta}), \ldots, i_T(\bar{\theta})$ generated according to the above transition probabilities and one-stage costs is a Markov cost process. For the simulation-based gradient algorithms, we also define a new Markov cost process with new transition probabilities and new one-stage costs as in Eq. (4). The new Markov cost process is a renewal process. So we need only

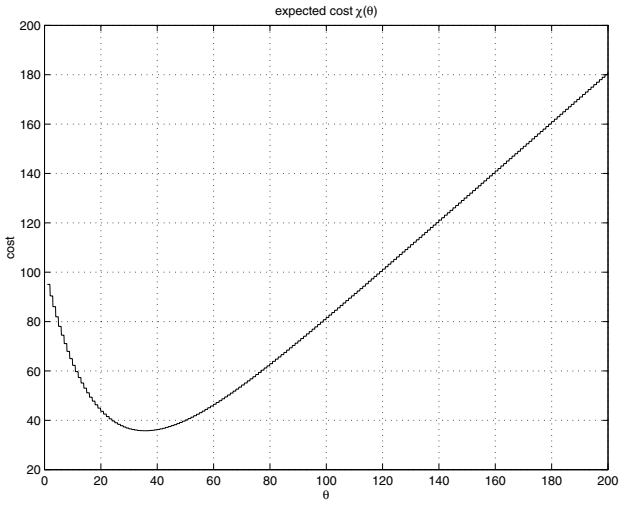

Fig. 1. Expected cost $\chi(\theta)$

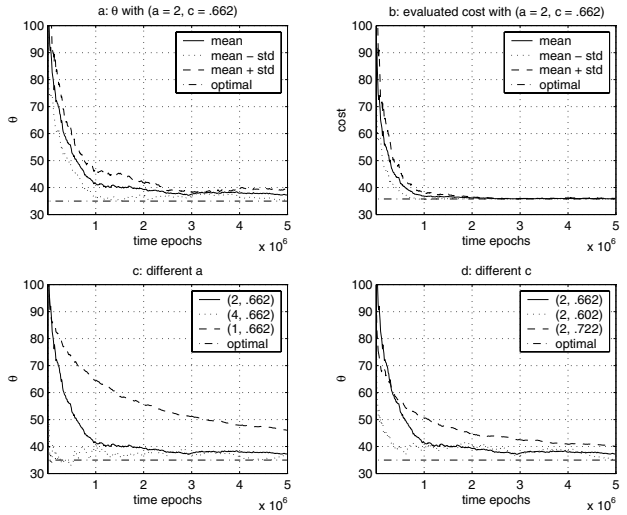

Fig. 2. Simulation-Based Regenerative-Update Gradient Algorithm

one single sample path when we apply simulation-based gradient algorithms, where the initial state is i^*.

V. Numerical Experiments

We now compare the three algorithms on the parking problem: the two-timescale algorithm [given by Eq. (21)], the (corrected) regenerative-update simulation-based gradient algorithm [Eq. (7) with $\bar{F}_m(\theta_m)$ given by Eq. (15) in place of $\hat{F}_m(\theta_m)$], and the (corrected) every-update simulation-based gradient algorithm [Eq. (8) with $R_k(x_k, \theta_k)$ given by (20) in place of $\hat{R}_k(\theta_k)$]. Note that the convergence of the latter two algorithms can be proved following the same arguments as in the proofs of Proposition 5 and Proposition 14 in [1]. In our numerical experiments, we consider the same case as in [12] where p=0.05, $c(s)$=s, C=100, and N=200. The optimal policy for this case is to park in the first available space after the parking space 35 is reached. Fig. 1 shows the expected cost $\chi(\theta)$ as a function of threshold θ. The optimal cost $\chi(\theta^*)$ = 35.7639 when $\theta^* = 35$.

All three algorithms were implemented in C, and experiments were conducted on a Sun Microsystems ULTRA10 running Solaris 2.6 operating system. For all the experiments, we selected initial parameter $\theta_0 = 100$, for which $\chi(\theta_0) = 81.7045$. For each result, we simulated four sample paths using independent seeds and show the mean and standard deviation of θ. When we implemented these three algorithms, we used step sizes $\alpha_j = a/j^c$ (where j is the updates index) and $\beta_l = b/l^d$ (required for the two-timescale algorithm), where a, b, c, and d are tunable positive parameters. We also set $n_0 = 1$ for the two-timescale algorithm.

regenerative-update algorithm: Since we dealt with a renewal process, we only needed to simulation one single sample path starting from state i^*. At every time epoch n, we calculated one-stage cost differential $\nabla g_{i_n}(\theta_m)$, transition probability $p_{i_n i_{n+1}}(\theta_m)$, transition probability differential $\nabla p_{i_n i_{n+1}}(\theta_m)$, and cumulative costs $\tilde{J}_{i_{n+1}}(\theta_m)$, where time epoch n is in the mth regenerative cycle. At the end of the mth regenerative cycle, we computed the gradient estimate using Eq. (15) and updated θ_m.

First we consider the regenerative-update algorithm implementation with fixed a and c. Figures 2(a) and 2(b) show how θ_m and $\hat{\chi}(\theta_m)$ converge to a near-optimal value, with $a = 2.0$ and $c = 0.662$. Note that we plot the progressions as a function of time epochs rather than as a function of iterative (regenerative) updates to facilitate comparison with the other algorithms. At time epoch n, $\theta_{n,1}$ to $\theta_{n,4}$ are four samples, their mean is $\bar{\theta}_n = (\sum_{i=1}^{4} \theta_{n,i})/4$, and their sample standard deviation (std) is calculated by $\sqrt{\sum_{i=1}^{4}(\theta_{n,i} - \bar{\theta}_n)^2/3}$. The trajectories shown in Figure 2(a) (or b) are the mean, the mean plus the standard deviation, the mean minus standard deviation, and the optimal value of θ_n (or $\chi(\theta_n)$). Note that θ converges to a value with near-optimal cost within $1,000,000$ time epochs. After that, θ_n improves more slowly, with some small oscillations. The final value $\theta_{5,000,000}$ = 37.34±1.82 and $\chi(\theta_{5,000,000})$ = 35.89±0.16.

Next, we investigated the sensitivity of this algorithm to a and c. From Fig. 2(c), we can observe that too small a value of a leads to slow convergence, but larger values cause overshoots. In contrast, Fig. 2(d) indicates the effect is opposite for c, i.e., too large a value leads to slow convergence, whereas small values can result in fluctuations. This type of sensitivity is typical of stochastic approximation algorithms.

Note that implementation of the regenerative-update algorithm requires storage of the accumulated costs $\tilde{J}_{i_{n+1}}(\theta)$ for all time epochs in a regenerative cycle before estimating the gradient estimate. If the expected length of a regenerative cycle is large, the storage requirements of this algorithm might make it impractical.

every-update algorithm: Implementation of the every-update algorithm requires calculation at each time epoch k of the one-stage cost $g_{i_k}(\theta_k)$, one-stage cost differential $\nabla g_{i_k}(\theta_k)$, transition probability $p_{i_k i_{k+1}}(\theta_k)$, and transition probability differential $\nabla p_{i_k i_{k+1}}(\theta_k)$, and then updating θ_k, L_k and z_k using (8), (19), and (18). At the end of each regenerative cycle, L_k and z_k must be reset.

With $a = 20.0$ and $c = 0.662$, Fig. 3(a) and 3(b) show how θ_k and $\hat{\chi}(\theta_k)$ converge to the optimal value. Again, θ_k is near-optimal within about $1,000,000$ time epochs, converging more slowly to the optimal after that, with $\theta_{5,000,000}$ = 37.12±0.70 and $\chi(\theta_{5,000,000})$ = 35.82±0.08. Sensitivity analysis of this algorithm w.r.t. a and c, as seen in Fig. 3(c) and 3(d), is basically the same as for the regenerative-update algorithm.

two-timescale algorithm: Implementation of the two-timescale algorithm is similar to the implementation of the every-update

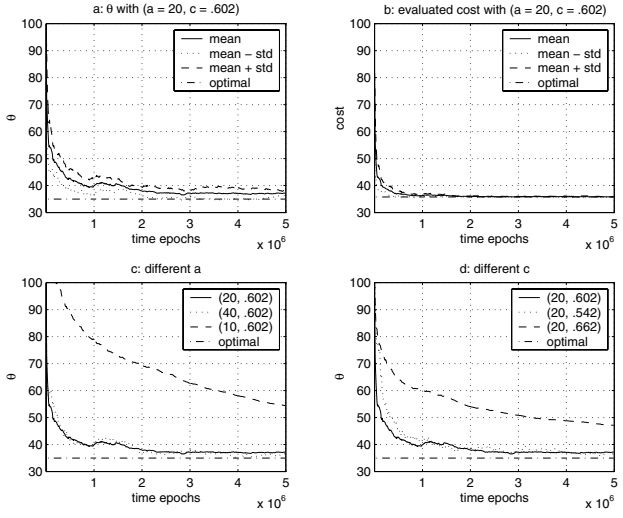

Fig. 3. Simulation-Based Every-Update Gradient Algorithm

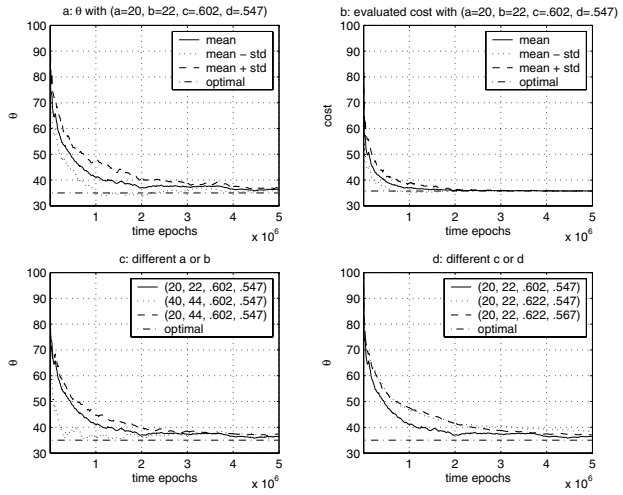

Fig. 4. Simulation-Based Two-Timescale Gradient Algorithm

algorithm, the main difference being that θ is updated at a slower scale, instead of at every time epoch. The number of time epochs before the next update of θ is characterized by n_l (see Eq. (10)) where l is the number of updates so far. At each time epoch k of the θ_l update, the quantities $g_{i_k}(\theta_l)$, $\nabla g_{i_k}(\theta_l)$, $p_{i_k i_{k+1}}(\theta_l)$, $\nabla p_{i_k i_{k+1}}(\theta_l)$ must be calculated, and L_k and z_k updated. At the end of each regenerative cycle, L_k and z_k are reset.

From Fig. 4(a) and 4(b), we can see how θ_l and $\hat{\chi}(\theta_l)$ converge to a near-optimal value, with $a = 20$, $b = 22$, $c = 0.602$, and $d = 0.547$. Again, θ_l proceeds rapidly to near optimality within $1,000,000$ time epochs, with final value $\theta_{5,000,000}$ = 36.37$\pm$0.75 and $\chi(\theta_{5,000,000})$ = 35.77$\pm$0.01.

Next, we investigated the sensitivity of this algorithm to a, b, c and d. We can see the influence of both a and b is similar to that of a in the regenerative-update algorithm from Fig. 4(c), and the influence of both c and d is similar to that of c in the regenerative-update algorithm from 4(d). Note that b and d affect the convergence behavior of the two-timescale algorithm only through n_l.

Comparing these three algorithms, only the regenerative-update algorithm's implementation requires storing interim values. Comparing the every-update algorithm and the two-timescale algorithm, the former requires more computations since it updates θ at each time epoch, but the latter contains more tunable parameters.

VI. DISCUSSION

In this paper, we proposed a new two-timescale simulation-based algorithm for weighted cost problems, and compared it with every-update and regenerative-update algorithms via numerical experiments on a parking example. Implementation of the regenerative-update algorithm requires storing interim values before each regenerative point. The numerical experiments show that the performance of the two-timescale algorithm is close to that of the every-update algorithm. However, in some situations when we need frequent estimation but infrequent control, the two-timescale algorithm is a better choice than the every-update algorithm.

In [13] [14], the authors proposed several simulation-based gradient algorithms for average cost Markov decision processes, and we are interested in formulating their two-timescale versions and compare their performances.

REFERENCES

[1] P. Marbach, "Simulation-based optimization of Markov decision processes," Ph.D. dissertation, Massachusetts Institute of Technology, 1998.

[2] V. S. Borkar, "Stochastic approximation with two time scales," *Systems and Control Letters*, vol. 29, no. 5, pp. 291–294, 1997.

[3] S. Bhatnagar and V. S. Borkar, "A two timescale stochastic approximation scheme for simulation-based parametric optimization," *Probability in the Engineering and Information Sciences*, vol. 12, pp. 519–531, 1998.

[4] ——, "Multiscale stochastic approximation for parametric optimization of hidden Markov models," *Probability in the Engineering and Information Sciences*, vol. 11, pp. 509–522, 1997.

[5] S. Bhatnagar, M. C. Fu, S. I. Marcus, and P. J. Fard, "Optimal structured feedback policies for abr flow control using two timescale SPSA," *IEEE/ACM Transactions on Networking*, vol. 9, no. 4, pp. 479–491, August 2001.

[6] S. Bhatnagar, M. C. Fu, S. I. Marcus, and S. Bhatnagar, "Two timescale algorithms for simulation optimization of hidden Markov models," *IIE Transactions*, vol. 33, no. 3, pp. 245–258, March 2001.

[7] S. Bhatnagar and V. S. Borkar, "Multiscale chaotic SPSA and smoothed functional algorithms for simulation optimization," *Simulation*, vol. 79, no. 10, pp. 568–580, 2004.

[8] V. R. Konda and V. S. Borkar, "Actor-critic-type learning algorithms for Markov decision processes," *SIAM Journal on Control and Optimization*, vol. 38, pp. 94–123, 2000.

[9] V. R. Konda and J. N. Tsitsiklis, "On actor-critic algorithms," *SIAM Journal on Control and Optimization*, vol. 42, pp. 1143–1166, 2004.

[10] S. Bhatnagar and S. Kumar, "A simultaneous perturbation stochastic approximation-based actor-critic algorithm for Markov decision processes," *IEEE Transactions on Automatic Control*, vol. 49, no. 4, pp. 592–598, 2004.

[11] Y. He, "Simulation based algorithms for Markov decision processes," Ph.D. dissertation, University of Maryland at College Park, 2002.

[12] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-Dynamic Programming*. Belmont: Athena Scientific, 1996.

[13] P. Marbach and J. N. Tsitsiklis, "Simulation-based optimization of Markov reward processes," *IEEE Transactions on Automatic Control*, vol. 46, no. 2, pp. 191–209, February 2001.

[14] ——, "Approximate gradient methods in policy-space optimization of Markov reward processes," *Journal of Discrete Event Dynamical Systems*, vol. 13, pp. 111–148, 2003.