Implementation of Gradient Estimation to a Constrained Markov Decision Problem

Vikram Krishnamurthy,
Department of Electrical and Computer Engineering, University of British Columbia, Vancouver, V6T1Z4, CANADA
and Department of Electrical and Electronic Engineering, University of Melbourne, Victoria 3010, AUSTRALIA
vikramk@ece.ubc.ca

Katherine Martin and Felisa Vázquez Abad*, member GERAD
Département d’informatique et recherche opérationnelle, Université de Montréal, Québec H3C 3J7, CANADA
{martink,vazquez}@iro.umontreal.ca

Abstract— Consider the problem of a constrained Markov Decision Process (MDP). Under a parameterization of the control strategies, the problem can be transformed into a non-linear optimization problem with non-linear constraints. Both the cost and the constraints are stationary averages. We assume that the transition probabilities of the underlying Markov chain are unknown: only the values of the control variables are known, as well as the instantaneous values of the cost and the constraints, so no analytical expression for the stationary averages is available. To find the solution to the optimization problem, a stochastic version of a primal/dual method with an augmented Lagrangian is used. The updating scheme uses a “measure valued” estimator of the gradients that can be interpreted in terms of a finite horizon version of the Perturbation Analysis (PA) method known as the “perturbation realization factors”. Most finite horizon derivative estimators are consistent as the sample size grows, so it is common to assume that large enough samples can be observed so as to make the bias negligible. This paper deals with the actual implementations of the gradient estimators in finite horizon with small sample sizes, so that the iterates of the stochastic approximation can be performed very often, as would be required for on-line learning. We identify the asymptotic bias of the stochastic approximation for the constrained optimization method, and by so doing we propose several means to correct it. As is very common with these problems, the bias correction introduces a conflict between precision and speed: the smaller the bias, the slower the reaction time.

In the sequel, we present the theoretical basis for the study of bias and learning rate. Our experimental results indicate that smoothing at a faster time scale may not be necessary at all, only at a slower time scale. We include results where the algorithms have to track changes in the environment.

I. INTRODUCTION

Consider the Markov Decision Process (MDP) \{X_n, u_n\}. The transition probabilities are:

\[ P[X_{n+1} = j | X_n = i, u_n = a] = A_{ij}(a), \quad a \in \mathcal{A}(i), \]

where \(i, j\) are states belong to a finite state space \(S\) and \(u_n\) is an action taken from a possible set \(\mathcal{A}(i)\). The control problem is to find the optimal policy \(\mathbf{u}^*:\)

\[ J_{x_0}(\mathbf{u}^*) = \inf_{\mathbf{u} \in \mathcal{U}} J_{x_0}(\mathbf{u}) \quad \forall x_0 \in S, \]

where \(\mathcal{U}\) is the set of admissible policies. When action \(a\) is taken at state \(i\), an instantaneous cost of \(b_0(i, a)\) is accumulated. For any admissible policy \(\mathbf{u} \in \mathcal{U}\), the average cost is defined as:

\[ J_{x_0}(\mathbf{u}) = \lim_{N \to \infty} \mathbb{E} \left[ \frac{1}{N} \sum_{n=1}^{N} b_0(X_n, u_n) \mid X_0 = x_0 \right]. \]

(1)

If the MDP is a unichain, then the limit in (1) exists for any \(x_0 \in S\) and \(\mathbf{u} \in \mathcal{U}\), and the limit is independent of \(x_0\). The constrained MDP problem imposes, in addition, that the conditions:

\[ \lim_{N \to \infty} \mathbb{E} \left[ \frac{1}{N} \sum_{n=1}^{N} b(X_n, u_n) \right] \leq 0, \quad l = 1, \ldots, L \]

be satisfied. It is well known [4], [5] that the optimal control strategy for the constrained optimization problem has the form of a randomized policy:

\[ \mathbb{P}[u_n = a | X_n = i] = \theta_{ia}, \quad a \in \mathcal{A}(i), \quad i \in S \]

where \(\sum_{a \in \mathcal{A}(i)} \theta_{ia} = 1\) for every state \(i \in S\).

The above MDP uses randomized decisions: given that the state is \(X_n = i\), any decision \(a \in \mathcal{A}(i)\) is possible with some positive probability \(\theta_{ia}\). Instead of using the action probabilities \(\theta\) as control variables, in [5] a particular parameterization for the action probabilities was suggested, which has several numerical advantages. Most notably, this parameterization does not require further constraints on the control variable (such as all probabilities adding up to one). In addition, it can achieve much faster convergence to the optimal values, especially when the optimal control probabilities for some states are degenerate, a very common situation. Let \(\mathcal{A}(i) = \{0, \ldots, d(i)\}\). Fix \(i \in S\) and to each \(\theta_{ia}, a \in \mathcal{A}(i)\) associate the value \(\lambda_{ia} = \sqrt{\theta_{ia}}\). Then we have \(\sum_{a \in \mathcal{A}(i)} \lambda_{ia}^2 = 1\) and \(\lambda_{ia}\) can be interpreted as the coordinates of a vector that lies on the surface of the unit sphere in \(\mathbb{R}^{d(i)+1}\). Let \(\alpha_{ip}, p \in \mathcal{P}(i) = \{1, \ldots, d(i)\}\) be the angles of the spherical coordinates, which will be the control
variables. Then a simple change of variables yields:

\[
\theta_{i,a} = \begin{cases} 
\cos^2 \alpha_{i1} & \text{if } a = 0 \\
\cos^2(\alpha_{i,a+1}) \prod_{k=1}^{i-1} \sin^2(\alpha_{ik}) & \text{if } 1 \leq a < d(i) \\
\sin^2(\alpha_{i,a}) \prod_{k=1}^{i-1} \sin^2(\alpha_{ik}) & \text{if } a = d(i)
\end{cases}
\]

Under this parametrization, the control problem can be re-stated as the following non-linear optimization problem:

\[
\max_{\alpha \in \Phi} B_0(\alpha) \\
\text{subject to: } B_l(\alpha) \leq 0, \quad l = 1, \ldots, L,
\]

where \(B_l(\alpha) = \mathbb{E}_\theta[b_l(X_{n_a}, u_a)], \quad l = 0, \ldots, L,\) and \(\mathbb{E}_\theta\) denotes the expectation w.r.t. the stationary measure of the process when the control variable \(\alpha\) is used to determine the randomized policy \(\theta\). Notice that, for each \(i \in S\), the control variable is not physically constrained: \(\alpha_i \in \mathbb{R}^{d(i)}\).

Amongst the possible numerical methods to solve such problems, we will consider here the primal dual method. This method requires the Lagrangian to be locally convex at the optimum. We can "convexify" the Lagrangian by adding a penalty term. Define the augmented Lagrangian:

\[
\mathcal{L}_\rho(\alpha, \lambda) = B_0(\alpha) + \sum_{l=1}^{L} \lambda_l B_l(\alpha) + \frac{\rho}{2} \sum_{l=1}^{L} (B_l(\alpha))^2.
\]

For a sufficiently large \(\rho > 0\), the last term in \(\mathcal{L}_\rho\) assures that the augmented Lagrangian is locally convex. Furthermore, under some regularity conditions (see \cite{1, 5}), the solution of the unconstrained augmented Lagrangian is the optimal solution of the original constrained problem. After the introduction of slack variables to transform the constraints to equality constraints, we obtain the primal dual algorithm for inequality constraints (see \cite{1}, page 429):

\[
\alpha'(n+1) = \alpha'(n) + \epsilon \left( \nabla_{\alpha} B_0(\alpha') - \nabla_{\alpha} B(\alpha')(\lambda'(n) + \rho B(\alpha')) \right) \tag{2}
\]

\[
\lambda'(n+1) = \max \{0, \lambda'(n) + \epsilon B(\alpha')\} \tag{3}
\]

where \(\epsilon > 0\) denotes the step size. Because our goal is to adapt this procedure on-line, a decreasing step size would prevent the algorithm to track optimality under changes in the underlying dynamics and/or costs. Therefore, a constant step size is chosen to endow the algorithm with learning capabilities: as the process is observed, the controller measures the functions \((B_l, \nabla B_l; l = 0, \ldots, L)\) and updates the control variable to achieve optimality.

The problem that we are interested in is the case where the control has to be learned from the observations of the basic "target" \(\{X_n\}\) without knowledge of the transition probabilities \(A_{i,j}(\alpha)\). That is, we assume that the states visited are observable, and by controlling \(\alpha\) we can set the action probabilities \(\theta(\alpha)\). What the controller does not know is exactly how the actions will affect the behavior of the target states (and also of the costs and constraint functions).

Without \(\{A_{i,j}(\alpha)\}\) it is impossible to compute analytically the stationary averages \(B_l(\alpha)\) and their gradients, which are necessary for the implementation of the primal/dual algorithm. Estimating the stationary averages \(B_l(\alpha), l = 0, \ldots, L\) can be done using sample averages, which are consistent estimators. To estimate the gradients, we use the measured valued derivative (MVD) estimator, which does not require explicit knowledge of the transition probabilities and it is consistent (see \cite{5}). The stochastic version of the primal dual above is:

\[
\alpha'(n+1) = \alpha'(n) + \epsilon \left( \nabla_{\alpha} B_0(\alpha) - \sum_{l=1}^{L} \lambda_l(\nabla B_l(\alpha)) \right) \tag{4}
\]

\[
\lambda'(n+1) = \max \{0, \lambda'(n) + \epsilon B(\alpha)\} \tag{5}
\]

Stochastic approximation algorithms of this form have been extensively studied recently (see \cite{3}). For this model, if the stationary averages of the right-hand-side of the finite difference equations (4) and (5), are the same as the deterministic drifts (2) and (3), then the stochastic version recovers the same asymptotic behavior as the deterministic primal dual algorithm, as \(\epsilon \to 0\). Replacing the unknown functions \(B_l(\alpha(n)), \nabla B_l(\alpha(n)), l = 0, \ldots, L\) by their finite horizon estimators introduces a bias in the asymptotic behavior of the stochastic approximation. In this paper we present the theoretical basis for the study of both bias and rate of convergence, as well as experimental results that indicate that smoothing at a faster time scale may not be necessary at all, only at a larger time scale. We include results where the algorithms have to track changes in the environment.

II. GRADIENT ESTIMATION

A. MVD ESTIMATOR

We now summarize the MVD estimator that was derived in \cite{5}, where it was called the "frozen phantom" estimator. For a MDP \(\{(X_{n_a}, u_a)\}\) under randomized decisions \(\{\theta_{i,a}\}\), define the \(k\)-th "phantom" decision \(\tilde{u}_k\) as a random variable, conditioned on the observation of the state \((X_k = i, u_k)\), satisfying:

\[
P(\tilde{u}_k = a) = \frac{\theta_{i,a}}{\prod_{p=1}^{n} \sin^2(\alpha_{ip})} \mathbb{1}_{\{a > u_k\}}, \quad \text{for } u_k < d(i) - 1
\]

\[
\tilde{u}_k = \begin{cases} 
d(i) & u_k = d(i) - 1 \\
(d(i) - 1) & u_k = d(i)
\end{cases}
\]

and define the hitting time of the \(k\)-th phantom process by:

\[
\nu(k) = \min \{n > 0 : X_{k+n} = X_k, u_{k+n} = \tilde{u}_k\}
\]

Let \(B(\alpha) = \mathbb{E}_\theta[b(X_n, u_n)]\) and suppose that the gradient \(\nabla B(\alpha)\) is to be estimated using the observations over an
interval be of the form \( I_n = \{ nN + 1, \ldots, (n + 1)N \} \). Let \( \hat{\Gamma}_n(B) \) be an estimator of \( B(\alpha) \) using observations of the MDP in \( I_n \). The frozen phantom estimators for the components \((i, p)\) of the gradient are:

\[
\frac{\partial B}{\partial \alpha_{i,p}}(n) = \frac{K_{ip}(\alpha)}{N} \sum_{k = nN+1}^{(n+1)N} \delta_{ip}(k) \left( b(i, u_k) - b(i, \tilde{u}_k) + \sum_{j = (k+1)N}^{(n+1)N} b(X_j, u_j) - \nu(k)\hat{\Gamma}_n(B)D_n(k) \right) + \sum_{k \in \mathcal{L}(nN)} \delta_{ip}(k) \left( \sum_{j = nN+1}^{(n+1)N} b(X_j, u_j) - \nu(k)\hat{\Gamma}_n(B)D_n(k) \right)
\]

(6)

where

\[
K_{ip}(\alpha) = -2 \tan(\alpha_{ip}), \quad p < d(i)
\]

\[
K_{ip}(\alpha) = -2 \sin(\alpha_{ip}) \cos(\alpha_{ip}), \quad p = d(i)
\]

\[
\delta_{ip}(k) = \left\{ \begin{array}{ll} 1_{\{X_k = i, u_k = p-1\}}, & p < d(i) \\ 1_{\{X_k = i, u_k = p\}} - 1_{\{X_k = i, u_k = p-1\}}, & p = d(i) \end{array} \right.
\]

\[
D_n(k) = 1_{\{k + \nu(k) \geq I_n\}} \quad (\text{phantom } k \text{ dies in } I_n)
\]

\[
\mathcal{L}(j) = \{ k \leq j; \nu(k) + k > j \}
\]

The interpretation of (6) is as follows. At each step \( k \), the state \( (X_k = i, u_k = a) \) is observed and a new phantom system (labelled by \( k \)) is started, generating the phantom decision \( \tilde{u}_k \) as described above. The \( k \)-th phantom system dies at the hitting time \( n = k + \nu(k) \), otherwise it is contained in the set of “living” phantoms \( \mathcal{L}(n) \). In the program, this corresponds to a list. This phantom will be used to estimate the partial derivative of all the functions (cost and constraints) with respect to \( \alpha_{i,a+1} \) if \( a < d(i) - 1 \). If \( a = d(i) - 1 \) or \( a = d(i) \), the corresponding phantom system contributes with opposite signs to the estimation of the gradient w.r.t. \( \alpha_{i,d(i)} \). The first term inside brackets in (6) contains the initial contribution of a phantom system. Afterwards, while a phantom system \( k \) is alive, it contributes to (6) the term \( b(X_k, u_k) \) at each step, and when it dies \( (D_n(k) = 1) \) it contributes the term \( \nu(k)\hat{\Gamma}_n(B) \) (if death occurs within the interval \( I_n \)). The above equation takes only observations of the trajectory within the current interval, thus a final term appears considering the contributions of all the living phantoms at the start of the interval, because it is possible that phantom systems may survive several estimation intervals. The proof of the following results is detailed in [5].

**Lemma 1:** Let \( \hat{\Gamma}_n(B) \) in (6) be the local sample average:

\[
\hat{\Gamma}_n(B) = \hat{\Gamma}_n = \frac{1}{N} \sum_{k \in I_n} b(X_k, u_k).
\]

If \( \sup_\alpha E_\alpha (|\nu(k)|^2) < \infty \), then for any \( n \in \mathbb{N} \)

\[
\lim_{N \to \infty} \hat{\nabla}B(n) = \nabla B(\alpha), \quad a.s.
\]

**Lemma 2:** If \( \sup_\alpha E_\alpha (|\nu(k)|^2) < \infty \) and if \( \hat{\Gamma}_n(B) \) is replaced by the actual average \( B(\alpha) \) in (6) then the stationary average of the estimator is unbiased: \( E_\alpha [\hat{\nabla}B(n)] = \nabla B(\alpha) \).

**B. Efficiency Improvement**

We were to code (6) exactly, we would need to define a list (a random sized vector) with information about the ages of all the living phantoms, updating it at every step. Upon death of a phantom system \( k \), the value \( \nu(k) \) would have to be recorded and at the end the estimation interval \( I_n \), the current estimate \( \hat{\Gamma}_n(B) \) would then have to be multiplied by the sum of all those values \( \nu(k) \) corresponding to phantom systems that died within \( I_n \). In order to create a more efficient code, we decided not to follow (6), as we now explain.

![Fig. 1. Estimation code](image)

Our code corresponds to the scheme illustrated in Figure 1. Suppose that a phantom \( k \) starts at period \( I_j \) and lives for \( M \) periods. In this case, we re-write \( \nu(k) = \nu_j(k) + \nu_{j+1}(k) + \cdots + \nu_{j+M}(k) \), representing the time that the system in question is alive during each period. Clearly \( \nu_j(k) = N, j < m < M \). At the end of an estimation interval, if a phantom is still alive, we add the contribution using the current estimator of \( B(\alpha) \), instead of waiting for the hitting epoch to occur. That is, instead of using the term \( \nu(k)\hat{\Gamma}_j+M(B) \), our program takes the partial contributions as the estimation intervals are completed:

\[
\sum_{m=0}^{M} \nu_{j+m}(k) \hat{\Gamma}_j+m.
\]

In the code, we update the list \( \mathcal{L} \), removing the phantoms as they die, and simply add one unit of “life” for each living phantom to an aggregate counter at each time step. At the end of the estimation interval, this aggregate number is multiplied by \( \hat{\Gamma}_n(B) \) and it is re-set to zero. We do not keep any more variables in the program, which not only saves memory requirements but also time, and it makes the algorithm more efficient.

Under this modification, the results of Lemma 1 and Lemma 2 are obviously still valid, because \( \hat{\Gamma}_n(B) \) would either be exactly equal, or would converge a.s. to \( B(\alpha) \). In either case, the product \( \hat{\Gamma}_n(B)\nu_n(k) \) converges in expectation to \( B(\alpha) E_\alpha [\nu_n(k)] \).
C. Bias Reduction

If $N$ is sufficiently large, then the local sample average estimator \( \hat{B}_n \) has a small variance (of order $1/N$) and using this for \( \hat{\Gamma}_n(B) \) in (6) may result in a negligible bias, by means of Lemma 1. However, if frequent updates are desirable for the algorithm, one may want to favor small batch sizes, in which case the bias may be reduced by using a series of alternative solutions, based on the result of Lemma 2 and aimed at reducing $\text{Cov}_n(\nu(k), \hat{\Gamma}_n)$:

- Shift the estimator, setting \( \hat{\Gamma}_n(B) = \hat{B}_{n-s} \) as the local sample average obtained $s$ batches before. Because of the Markovian structure of the process, the correlation will decrease as $s$ increases.
- Use the total running average:
  \[
  \hat{\Gamma}_n(B) = \hat{B}_n = \frac{1}{nN} \sum_{k=1}^{nN} b(X_k, u_k).
  \]
- Use a window average of the form ($s > N$):
  \[
  \hat{\Gamma}_n = \frac{1}{s} \sum_{m=n-s}^{n} \hat{B}_m.
  \]
- Use the exponentially discounted average \( \hat{\Gamma}_n(B) = \hat{B}_n^{\delta} \), defined by:
  \[
  \hat{B}_{n+1}^{\delta} = \delta \hat{B}_{n+1} + (1 - \delta) \hat{B}_n^{\delta}, \quad \delta \in (0, 1)
  \]
  where \( \hat{B}_n \) is the local sample average.

Rather than studying the bias in the gradient estimation itself, we study the effect of this (unknown) bias on the behavior of the stochastic approximation procedure in the following section.

III. Behavior of the Self-Optimized MDP

Let \( h_l(\alpha) \) denote the stationary average of the gradient estimators using (6) over an estimation interval:

\[
  h_l(\alpha) = \mathbb{E}_\alpha[\nabla \hat{B}_l(n)], \quad l = 0, \ldots, L
\]

Consider now the MDP under on-line control. The estimation of the various costs, constraints and gradients is performed over observation intervals $I_n$ of size $N \geq 1$, in which $\alpha$ is kept constant. At the end of the interval, $\alpha$ is updated following (4) and (5) using the results of the estimation interval. When $\alpha$ is updated, a new observation interval starts, subject to the new $\alpha$. Define the piece-wise constant interpolated continuous-time process:

\[
  \alpha^*(t) = \alpha^*(n) \quad t \in [n\epsilon, (n+1)\epsilon)
\]

\[
  \lambda^*(t) = \lambda^*(n) \quad t \in [n\epsilon, (n+1)\epsilon),
\]

where $t \in \mathbb{R}^+$ denotes the continuous time. The following result can be shown using the standard methodology for weak convergence (see [3], chapter 8).

**Proposition 1:** As $\epsilon \to 0$, the interpolated process converges in distribution to the solution of the ODE:

\[
  \frac{d\alpha(t)}{dt} = -\left( h_0(\alpha(t)) + \sum_{l=1}^{L} (\lambda_l + \rho B_l(\alpha(t))) h_l(\alpha(t)) + \kappa(\alpha(t)) \right)
\]

\[
  \frac{d\lambda_l(t)}{dt} = \begin{cases} B_l(\alpha(t)) & \text{if } \lambda_l(t) \geq 0 \\ 0 & \text{if } \lambda_l(t) = 0 \text{ and } \frac{d\lambda_l(t)}{dt} < 0 \end{cases}
\]

$l = 1, \ldots, L$, where the added drift is defined as:

\[
  \kappa(\alpha) = \rho \lim_{n \to \infty} \sum_{l=1}^{L} \text{Cov}_n[\hat{B}_l(n), \nabla \hat{B}_l(n)].
\]

When $h_l(\alpha) = \nabla B_l(\alpha), l = 0, \ldots, L$ and $\kappa(\alpha) = 0$, the ODE above corresponds to the asymptotic behavior of the deterministic primal dual algorithm (2)–(3) as $\epsilon \to 0$, whose stable (or limit) points correspond to local minima $\alpha^*$ and corresponding Lagrange multipliers $\lambda^*$, for sufficiently large $\rho$ (see [5] for the details).

Our simulation experiments were all performed for a system with two states $S = \{0, 1\}$ and three actions, $A = \mathcal{A}(i), i \in S$. The transition matrices are:

\[
  A(0) = \begin{pmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{pmatrix}, \quad A(1) = \begin{pmatrix} 0.3 & 0.7 \\ 0.6 & 0.4 \end{pmatrix}
\]

\[
  A(2) = \begin{pmatrix} 0.5 & 0.5 \\ 0.1 & 0.9 \end{pmatrix}.
\]

The cost and constraint matrices $b_l(i, u)$ are given by:

\[
  b_0(i, u) = \begin{pmatrix} 50 & 200 & 10 \\ 3 & 500 & 0 \end{pmatrix}, \quad b_l(i, u) = \begin{pmatrix} 20 & 100 & -8 \\ -3 & 4 & -10 \end{pmatrix}
\]

The optimal cost of the experiments without tracking was $B_0(\alpha^*) = 238.8$. For the tracking experiment the constraint matrix changes suddenly in the middle of the simulation.

A. Batch Size

From the consistency of the estimation in Lemma 1, the behavior of the stochastic approximation should be close to that of the deterministic scheme (2)–(3) when $N$ is large. Our first tests use the local sample averages $\hat{\Gamma}_n(B) = \hat{B}_n$ in (6) as well as in (4)–(5) for the estimation of the constraint: $\hat{B}(n) = \hat{B}_n$. Figure 2 shows the cost $B_0(\alpha^*(n)), n = 1, 2, \ldots, 2000$ of the sample trajectories of the algorithm (4)–(5) using estimation intervals of sizes $N = 10, 100$ and compares their behavior with that of (2)–(3) (labelled "pd_ODE"). As expected, the influence of the added drift $\kappa(\alpha(t))$ is negligible when $N$ is large, because the local sample averages of the constraints are very accurate, $\hat{B}^*(n) \approx B(\alpha(n))$. We performed numerous simulations with different parameters, and all showed the same behavior: small values of $N$ are not asymptotically optimal, and the bias may be very noticeable. For on-line control, it is undesirable to choose a large value of $N$ which may deteriorate the actual cost of the system under control. While this is not usually a problem in the context of simulation optimization, it may hinder the learning
capabilities of an on-line scheme that should be able to react quickly. The scale shown in the graph is iteration number and not actual time as per the MDP dynamics. Because $N$ observations must be made before updating, the plots for the stochastic approximation algorithms are $N$ times slower than the reaction time of their deterministic counterpart. Notice that as $N$ becomes smaller, the “right” direction of the cost function is quickly picked up by the algorithm.

B. Exponential Smoothing

In the experiments that follow, we used a batch size $N = 10$. The methods of smoothing discussed in Section 2 are all valid and may give asymptotically negligible bias (if the parameters are chosen correctly) even when $N = 1$. In this work we focus on the exponential average as a means to counter balance the bias. It follows from Proposition 1 that the two main sources of bias are due to inaccurate estimation of $B(\alpha)$ in (6) through $\hat{\Gamma}_n(B)$ as well as inaccurate estimation of $B(\alpha)$ in (4)–(5), which entails a nonzero value of $\kappa(\cdot)$. We study here the behavior of the algorithm using exponential averaging to counteract the sources of bias. Let $\bar{B}(n) = \bar{B}_n^\delta$ be chosen as an exponential average in (4)–(5). The updating equations can now be re-written by calling $\mu_l^\delta(n) = B_l(n)$, thus:

$$\alpha^\delta(n + 1) = \alpha^\delta(n) + \epsilon \left( \nabla B_0(n) - \sum_{l=1}^{L} (\lambda_l^\delta(n) + \rho \mu_l^\delta(n)) \nabla B_l(n) \right)$$  \hspace{1cm} (7)

$$\lambda_l^\delta(n + 1) = \max \left[ 0, \lambda^\delta(n) + \epsilon \mu_l^\delta(n) \right]$$  \hspace{1cm} (8)

$$\mu_l^\delta(n + 1) = \mu_l^\delta(n) + \delta (\bar{B}_l - \mu_l^\delta(n))$$  \hspace{1cm} (9)

$l = 1, \ldots, L$. Using a two time scale stochastic approximation argument it can be shown that if $\epsilon/\delta \to 0$, e.g., if $\delta = \sqrt{\epsilon}$, and $\epsilon \to 0$, then the asymptotic limit points are unbiased. In practical implementation, for non zero $\delta$, the estimates are biased.

Figure 3 shows the trajectories of the cost function $B_0[\alpha^\delta(n)]$ against iteration number, with estimation intervals of size $N = 10$. The extreme cases are denoted “pd_ODE”, labelling the trajectory of the deterministic algorithm (2)–(3), and “pd_1.0”, which uses the local sample averages $\delta = 1.0$ (no smoothing at all). The plot labelled “pd_0.001BG” corresponds to the algorithm where $\hat{\Gamma}_n(B)$ in (6) uses the exponential smoothing with $\delta = 0.001$ and (7)–(9) is used for the updates with $\delta = \sqrt{\epsilon} = 0.001$. The plot labelled “pd_0.001B” takes the local sample average for the estimates in $\hat{\Gamma}_n(B)$ of the gradient in (6). We verified in numerous experiments that the effect of the bias in $\hat{\Gamma}_n(B)$ for the behavior of the overall learning algorithm is negligible compared to the bias caused by the term $\kappa(\cdot)$. In other words, the self-optimized MDP is more sensitive to the lack of accuracy in the estimation of the Lagrange multiplier through the estimation of the constraint values $B_l(\cdot)$, than to the bias of the finite horizon MVD estimator. The better the estimation of $B_l$, the better $\text{Cov}_n[B_l(\alpha), \nabla B_l(\alpha)] \approx 0$.

Naturally, the asymptotic behavior under exponential smoothing corresponds to a deterministic update algorithm of the form (2), but with the terms $(\lambda^\delta(n) + \rho B_l[\alpha^\delta(n)])$ replaced by $(\lambda^\delta(n) + \rho B_l[\alpha^\delta(n)])$. Instead of (3), use $\lambda_l^\delta(n + 1) = \max \left[ 0, \lambda_l^\delta(n) + \epsilon \mu_l^\delta(n) \right]$, and add the update:

$$\mu_l^\delta(n + 1) = \mu_l^\delta(n) + \delta (\bar{B}_l - \mu_l^\delta(n))$$  \hspace{1cm} (9)

These modifications introduce a memory from the past values and therefore there is a slow down of the transient behavior. This is apparent in Figure 3, and all our results showed that the reactive capacity increases with decreasing values of $\delta$, but the limit points have increasing errors from the optimal value.

C. Tracking

The last experiment shows the behavior of the self-optimized MDP algorithm under exponential smoothing in the slower time scale only; $\hat{\Gamma}_n(B)$ is the local sample average, with $N = 10$. At some point in the middle of
the experiment we drastically changed the values of the constraint. The new optimal value must now be tracked by the learning algorithm. Figure 4 shows the results with varying values of $\delta$. As is very common with these problems, the bias correction introduces a conflict between precision and speed: the smaller the exponential discount factor, the smaller the asymptotic bias, but it entails a slower reactive time. Under no smoothing, the system responds very fast but it gives rise to a bad solution due to a large value of $\kappa(\alpha)$.

![Figure 4. Primal Dual algorithm with tracking](image)

IV. CONCLUDING REMARKS AND FUTURE RESEARCH

In this paper we examined an important practical question when dealing with stochastic approximation algorithms for on-line learning, where it is undesirable to use decreasing learning rates and increasing estimation intervals. While theoretical analysis predicts good asymptotic behavior, the implementation of finite horizon estimates for small sample sizes often introduces a source of bias that can be removed at the expense of slowing down the adaptation capabilities. A practical conclusion of this work is that the predicted bias of the finite horizon perturbation analysis method seems to be negligible (even for small sample sizes $N = 5$ and $N = 1$) which is rather surprising. This fact was verified as a result of numerous experiments, where the behavior of the algorithm using the true theoretical value of $B[\alpha^* (n)]$ for the estimation of $\Gamma_n(B)$ in (6), was virtually identical to the behavior using instead the sample average $\Bar{B}_n$ (even for small sample sizes). This result implies that smoothing at the faster time scale of event clocks may be unnecessary. The bias seems to come mostly from the estimation required for the Lagrange multiplier updates. Amongst the various mechanisms for bias reduction, we studied here the exponential smoothing in the updates in (9), which is very popular in the literature.

As shown in Figure 4, however, the tracking behavior may be inferior to the much simpler solution of using large values of $N$. Indeed when the smoothing is good enough (small $\delta$) to recover the correct asymptotic behavior, it incorporates too much memory in the system and the reaction time may be too slow. We are currently exploring the possibility of dynamically adjusting various parameters, such as $N$ and $\delta$, and compare it with the less desirable alternative of a sliding window, which needs to keep track of many more arrays in the computer code, but may yield a better compromise between speed and precision.

V. ACKNOWLEDGEMENTS

This research was supported by the Australian Research Council and research grants from NSERC, Canada and FCAR, Quebec. The research of the second and third authors was partially completed while visiting the ARC Special Research Centre for Ultra-Broadband Communication Networks, University of Melbourne, Australia.

VI. REFERENCES