Solving MDPs using Two-timescale Simulated Annealing with Multiplicative Weights

Mohammed Shahid Abdulla and Shalabh Bhatnagar
Department of Computer Science and Automation
Indian Institute of Science, Bangalore, INDIA
email: {shahid,shalabh}@csa.iisc.ernet.in

***Abstract*— We develop extensions of the Simulated Annealing with Multiplicative Weights (SAMW) algorithm that proposed a method of solution of Finite-Horizon Markov Decision Processes (FH-MDPs). The extensions developed are in three directions: *a)* Use of the dynamic programming principle in the policy update step of SAMW *b)* A two-timescale actor-critic algorithm that uses simulated transitions alone, and *c)* Extending the algorithm to the infinite-horizon discounted-reward scenario. In particular, *a)* reduces the storage required from exponential to linear in the number of actions per stage-state pair. On the faster timescale, a 'critic' recursion performs policy evaluation while on the slower timescale an 'actor' recursion performs policy improvement using SAMW. We give a proof outlining convergence w.p. 1 and show experimental results on two settings: semiconductor fabrication and flow control in communication networks.**

Keywords

Markov decision processes, reinforcement learning, two timescale stochastic approximation, Simulated Annealing with Multiplicative Weights.

I. Introduction

Markov decision processes (MDPs) are a general framework for solving stochastic control problems. Value iteration and policy iteration are two of the classical approaches for solving the Bellman equation for optimality. Whereas value iteration proceeds by recursively iterating over value function estimates starting from a given estimate, policy iteration does so by iterating over policies and involves updates in two nested loops. The inner loop estimates the value function for a given policy iterate (policy evaluation phase) while the outer loop updates the policy (policy improvement phase).

Since an MDP with finite state and action spaces will also have a finite number of policies, the method of policy iteration converges to the optimal policy in a finite number of steps. However, in both the above classical approaches, one requires complete knowledge of the system model via transition probabilities. Even if these are available, the 'curse of dimensionality' makes the computational requirements prohibitive since classical approaches do not scale well to state- or action-space size. Research on simulation-based methods that largely go under the rubric of reinforcement learning or neuro-dynamic programming [1] is motivated by these constraints and has gathered momentum recently. The main idea in these schemes is to simulate transitions instead of estimating transition probabilities and, in scenarios where the numbers of states and actions are large, use parametric representations of the cost-to-go function and/or policies.

The proposed algorithms are simulation-based analogs of policy iteration, where one proceeds using two coupled recursions driven by different step-size schedules or timescales. The policy evaluation step (termed the 'critic') is performed on the faster timescale while the policy improvement step (the 'actor') is carried out along the slower one. Thus, both recursions are executed in tandem, one after the other, and the actor (resp. critic) converges to the optimal policy (resp. value function corresponding to the optimal policy). Two-timescale stochastic approximation is further applied in [2], where parameterizations of both value-function and policy are considered.

Another application of the two-timescale method, is the simulation-based policy iteration algorithm of [3] which performs updates in the space of deterministic stationary policies and not randomized stationary policies (RSPs) as do [4] and [2]. In the same vein, [5] considers the finite-horizon scenario and therefore uses Markov randomized policies (MRPs). In the 'actor' recursion of [5], [6] and [3] the policy-gradient is computed using the Simultaneous Perturbation Stochastic Approximation (SPSA). Note that unlike classical policy iteration the optimal policy in all the above methods is reached only asymptotically.

The faster timescale recursion in [3] is similar to the corresponding recursion in [4] except that, in [3], an additional averaging over L epochs (for some fixed $L > 1$) is proposed for enhanced convergence behaviour. This averaging takes the form of simulating L transitions out of each state in every step of the critic recursion. We employ a variation of this technique in the proposed SAMW-F and SAMW-D: we simulate L transitions out of a given state x for each action in the feasible set. The tag F in SAMW-F indicates finite horizon, whilst D in SAMW-D stands for infinite horizon discounted reward. SAMW-F performs policy iteration over the space of MRPs (as does the algorithm of [5]), while SAMW-D iterates over the space of RSPs (as do the algorithms in [4], [6]). The key difference is that the actor recursions of SAMW-F and SAMW-D perform policy improvement using the SAMW technique.

A. Comparison with SAMW

The proposed algorithm for the finite-horizon case is known as SAMW-F and the decision-horizon has duration T. At each policy-improvement iteration $i \in \mathcal{Z}_+$, SAMW-F takes a step towards the optimal decision rule $\phi_l^*(x)$ for a given stage-state pair (l, x) by comparing the $Q-$values of each action in the feasible set $U_{l,x}$. Note our somewhat unconventional usage of the term $Q-$value: the $Q-$value of action $a \in U_{l,x}$ is the sum

of the single-stage reward obtained from applying a in (l,x) and the cost-to-go from the subsequent stage $l+1$ under the policy iterate ϕ^i. Assuming that all states in S are feasible in all stages l, the proposed SAMW-F views the entire T-stage policy as a $T\times|S|-$dimensional vector, and updates all components of this vector at each i. We show that as i goes to infinity, the SAMW-F iterates ϕ^i converge to the optimal T-stage finite horizon policy. On the slower timescale, we update the policy ϕ^i using the SAMW technique - the update step involving estimates of $Q-$value computed by the faster timescale recursion. With analogous modifications, the SAMW-D for infinite-horizon discounted-reward also possesses the properties described above.

We outline some key features of the proposed SAMW-F vis-a-vis SAMW:

- Both are policy-iteration algorithms where only asymptotic efficiency is assured. This is in contrast to the policy-iteration behaviour for MDPs with finite state space and finite action sets.
- SAMW uses *meta-policy* iterates ϕ^i i.e. it operates on the space of probability vectors over the entire deterministic policy space Π. In contrast, SAMW-F restricts to (l,x) pairs: ϕ^i has components $\phi^i_l(x)$, $\forall x \in S_l$ and $0 \leq l \leq T-1$ over the (far smaller) feasible action set $U_{l,x}$, where S_l represents the feasible states in stage l.
- SAMW (resp. SAMW-F) has the attraction that policy iterates ϕ^i (resp. $\phi^i_l(x)$) do not exit the probability simplex at any iteration i. In other algorithms that update MRPs (e.g. [4], [7], [5], [6]), the policy iterates $\phi^{i+1}_l(x)$ exit the probability simplex when first computed and need to be projected therein.
- Proposed SAMW-F has two storage requirements: the MRP ϕ^i, and a look-up table of $Q-$values $Q^i_l(x,a)$ thus resulting in a $2\sum_{l=1}^{T-1}\sum_{x=1}^{|S_l|}|U_{l,x}|-$sized storage. SAMW stores the probability vector ϕ over Π, a much larger set (note that $|\Pi| = \times_{l=0}^{T-1}\times_{x=1}^{|S_l|}|U_{l,x}|$).
- In its current form, SAMW cannot be extended to infinite-horizon problems since repeated simulation of infinite trajectories is not possible. SAMW-D overcomes this handicap, by using the Dynamic Programming principle.

When using MRPs or RSPs, a large proportion of processor-time is devoted to projecting $\phi^i_l(x)$ iterates into the probability simplex. In [5] and [6], the computational effort for projection is far in excess of that needed for one policy update and consumes nearly $\frac{2}{3}-$rds of total computation time. The SPSA policy update step in these references only requires two additions, a multiplication and (an easy) division. In contrast, the projection algorithm represents an optimization problem with an iterative algorithm that takes $|\phi^i_l(x)|$ number of steps to solve (cf. [8, §5.5.2]). There are work-arounds available, e.g. Algorithms 3 and 6 of [4] assume that RSPs are of a Gibbsian architecture where the parameters belong to compact intervals, thus eliminating the projection onto a simplex. However, sub-optimality of the resulting policy is an automatic consequence of such a restriction.

To describe the outline of the article: Section II describes the framework of a finite-horizon MDP, provides a brief background of the SAMW algorithm, and identifies the conditions that the stepsizes should satisfy. Section III proposes SAMW-F while Section IV extends it to the infinite-horizon case. In section V, we describe experiments in two frameworks: one of semiconductor fabrication and the other of flow control in communication networks. We dwell on some future directions in section VI.

II. Framework and Description of SAMW

Consider an MDP $\{X_l, l=0,1,...,T\}$ with decision horizon $T<\infty$. Suppose $\{Z_l, l=0,1,...,T-1\}$ be the associated control valued process. Decisions are made at instants $l=0,1,...,T-1$, and the process terminates at instant T. Let state space at epoch l be S_l and let the control space at epoch l be Π_l, $l=0,1,...,T-1$. Note that S_T is the set of terminating states of this process. Let $p_l(x,a,y)$, $x\in S_l$, $a\in\Pi_{l,x}$, $y\in S_{l+1}$, $l=0,1,...,T-1$ denote the transition probabilities associated with this MDP. An admissible deterministic policy π is a set of T functions $\pi=\{\pi_0,\pi_1,...,\pi_{T-1}\}$ with $\pi_l: S_l \longmapsto \Pi_l$ such that $\pi_l(x)\in U_{l,x}$, $\forall x\in S_l$, $l=0,1,...,T-1$. In stage l with the system in state x, a controller under policy π applies action $\pi_l(x)$. Let $K_l(x,a)$ denote the $l-$th stage reward when state is $x\in S_l$, the action chosen is $a\in\Pi_{l,x}$ and the subsequent state is $y\in S_{l+1}$, respectively, for $l=0,1,...,T-1$. Note that rewards of the form $K_l(x,a,y)$ can also be admitted, we have chosen $K_l(x,a)$ for economy of notation. Also, let $K_T(y)$ denote the terminal cost at instant T when the terminating state is $y\in S_T$. The aim here is to find an admissible policy $\pi=\{\pi_0,\pi_1,...,\pi_{T-1}\}$ that maximizes for all $x\in S_0$ the total-reward: $V_{0,x}(\pi)=$

$$E\left\{K_T(X_T)+\sum_{l=0}^{T-1}K_l(X_l,\pi_l(X_l))|X_0=x\right\}, \quad (1)$$

the expectation above being over the joint distribution of $X_1,X_2,...,X_T$.

The Dynamic Programming algorithm for this problem is well-known (see [9]). The problem is broken down into T coupled maximization problems with each of these sub-problems, in turn, defined over the corresponding feasible set $U_{l,x}$. Here,'coupled' means that the $l-$th problem depends on the solution of the $(l+1)-$th problem, for $0\leq l\leq T-1$. In this paper we are interested in scenarios where $p_l(x,a,y)$ are not known, however, where transitions can be simulated. Thus, given state of the system at stage l is x and action a is picked (possibly randomly), we assume that the next state y can be obtained through simulation.

A. Algorithm SAMW

A description of the SAMW algorithm of [10] follows. SAMW operates on meta-policy iterates ϕ^i, as we explain below. Define $U_l = \times_{x=1}^{|S_l|}U_{l,x}$ and further, $\Pi=\times_{l=0}^{T-1}U_l$. A deterministic policy (DP) π is a $\sum_{l=0}^{T-1}|S_l|-$size vector s.t. $\pi\in\Pi$ and the meta-policy iterate ϕ^i in SAMW is a probability vector on the $\times_{l=0}^{T-1}\times_{x=1}^{|S_l|}|U_{l,x}|$ number of DPs. We use the same notation for a Markov randomized policy (RP) iterate ϕ^i having $|U_{l,x}|-$sized components $\phi^i_l(x)$ which is a probability vector over $U_{l,x}$. This policy iterate ϕ^i is such that $\phi^i_l(x)=(\phi^i_l(x,a),\forall a\in U_{l,x})^T$, $\phi^i_l=(\phi^i_l(x),\forall x\in S_l)^T$ and $\phi^i=(\phi^i_l, 0\leq l\leq T-1)^T$. Note that $|\phi^i|=\sum_{l=0}^{T-1}\sum_{x=1}^{|S_l|}|U_{l,x}|$ has a smaller number of elements and coupled with a look-up table of similar size, implies lesser storage.

For each $\sigma\in\Pi$, implementation of SAMW involves generating a trajectory $\{X_l\}_{l=0}^T$ with X_0 chosen according to an a-priori distribution δ and actions $\sigma_l(X_l)$

applied at each stage l. This trajectory produces the $i-$th total-reward estimate for policy σ, the random variable $V_i^\sigma(x_i)$ with $X_0 = x_i$ as the start state. We produce this estimate for each $\sigma \in \Pi$ and update the policy ϕ^i as per the following recursion:

$$\phi^{i+1}(\pi) \quad = \quad \frac{\phi^i(\pi)\beta_i^{V_i^\pi(x_i)}}{\sum_{\sigma\in\Pi}\phi^i(\sigma)\beta_i^{V_i^\sigma(x_i)}}\forall\pi\in\Pi.$$

We note the analogy with how a *deterministic* iterate $\phi^{i+1}(\pi)$ would be produced, the above operator being a soft-maximization.

The key SAMW condition is on β_i, where $\beta_i \downarrow 1$ and $i\cdot\frac{\beta_i-1}{\beta_i} \uparrow \infty$. The $\{\beta_i\}$ used in Section V is $\beta_i = 1 + i^{-0.6}$. In every update i, we need to simulate a trajectory $\forall\sigma\in\Pi$, which may be a simulation burden on the algorithm. In [10], an $\epsilon-$cut SAMW is suggested where trajectories at iteration i are simulated only for such $\sigma\in\Pi$ whose $\phi^i(\sigma) > \epsilon$, otherwise $V_{i-1}^\sigma(x_{i-1})$ are reused. However, one would have to store such σ, upto a maximum of $|\Pi|$.

B. Two-timescale stepsizes

In the proposed SAMW-F and SAMW-D, we require β_i to satisfy additional conditions over [10], and introduce the stepsize α_i. Call $\bar{\beta}_i = \ln\beta_i$, and assume that these satisfy

$$\beta_i \downarrow 1 \quad , \quad i\cdot\frac{\beta_i-1}{\beta_i}\uparrow\infty,$$

$$\sum_i \bar{\beta}_i = \sum_i \alpha_i = \infty, \ \sum_i \bar{\beta}_i^2, \sum_i \alpha_i^2 < \infty,$$

and

$$\bar{\beta}_i = o(\alpha_i).$$

Thus $\{\alpha_i\}$ goes to zero faster than does $\{\bar{\beta}_i\}$, which corresponds to the slower timescale. Likewise, $\{\alpha_i\}$ is the faster timescale. Informally, a recursion using α_i as the stepsize views a recursion guided by $\bar{\beta}_i$ as quasi-static whilst this latter recursion views the former as equilibrated.

III. Proposed Algorithm (SAMW-F)

Terms S_l and U_l denote the state and control spaces, respectively. Further, $U_{l,x}$ denotes the set of feasible actions in state $x\in S_l$ and may vary based on stage: however $|U_{l,x}| < \infty$. As in SAMW, for theoretical reasons we will need the following assurance on per-stage rewards.

Assumption 1: The cost functions $K_l(x,a), x\in S_l$, $a\in U_{l,x}$, are bounded for all $l = 0,1,...,T-1$ s.t. $K_l(x,a) < \frac{1}{T}$. Similarly, $K_T(x) < \frac{1}{T}$, $\forall x\in S_T$.
Note that this assumption is non-restrictive as it is a scaling of the one-step rewards.

Algorithm SAMW-F

- *Step 0 (Initialize): Fix* $\phi_l^1(x,a) = \frac{1}{|U_{l,x}|}$, $\forall x\in S_l$, $\forall a\in U_{l,x}$, $0\le l\le T-1$ *as the initial MRP iterate. Fix integers* L *and (large)* M *arbitrarily. Choose step-sizes* β_i *and* α_i *as in Section II-B. Set critic iterates* $Q_l^k(x,a) = 0$, $0\le l\le T$, $\forall x\in S_l$, $a\in U_{l,x}$ *and* $k = 0,1,...,L-1$ *as initial estimates of* $Q-$*values. Set actor index* $i := 1$.
- *Step 1 (Critic): for* $m = 0,1,...,L-1$, $l = T-1, T-2,...,0$, *and* $x\in S_l$, *do*
  - *for each* $a\in U_{l,x}$, *do*
    - *Simulate next state* $\eta \equiv \eta_l^{iL+m}(x,a)$ *according to distribution* $p_l(x,a,\cdot)$.
    - *Simulate action* $\xi \equiv \xi_l^{iL+m}(x,a)$ *for* η *according to* $\phi_{l+1}^i(\eta)$.
    - $Q_l^{iL+m+1}(x,a) := (1-\alpha_i)Q_l^{iL+m}(x,a)$
    $$+\alpha_i(K_l(x,a) + Q_{l+1}^{iL+m}(\eta,\xi)). \quad (2)$$
- *Step 2 (Actor):* $\forall x\in S_l$ *and* $0\le l\le T-1$*:*
  - $Z_l^i(x) := \sum_{a\in U_{l,x}}\phi_l^i(x,a)\beta_i^{Q_l^{(i+1)L}(x,a)}$
  -
  $$\phi_l^{i+1}(x,a) \quad := \quad \frac{\phi_l^i(x,a)\beta_i^{Q_l^{(i+1)L}(x,a)}}{Z_l^i(x)} \quad (3)$$
  - *Set* $i := i+1$. *If* $i = M$, *go to Step 3; else go to Step 1.*
- *Step 3 (termination): Terminate algorithm and output* ϕ^M *as the final policy.*

Note that two RVs are simulated in Step 1, the first where specific actions $a\in U_{l,x}$ are used to produce the next state $\eta_l^{iL+m}(x,a)$ and the second where action in this state is chosen as per iterate ϕ^i. The termination of Step 3 can also occur after a convergence test - a method used in the experimental results of Section V. Further, efficient performance can be obtained by scaling iterates $Q_l^{(i+1)L}(x,a)$, $\forall a\in U_{l,x}$ in (3) to spread over the interval $[0,1]$.

A. Outline of Convergence Analysis

We now proceed with a brief convergence analysis of the algorithm. To observe the 'actor-critic' interaction between (2)-(3), (3) can be written as an additive recursion:

$$\bar{\phi}_l^{i+1}(x,a) \quad = \quad \bar{\phi}_l^i(x,a) + \bar{\beta}_i(Q_l^{(i+1)L}(x,a) - \frac{\bar{Z}_l^i(x)}{\bar{\beta}_i}),$$

where $\bar{\phi}$, $\bar{\beta}_i$ and $\bar{Z}$ are logarithms of ϕ, β_i, and Z,respectively. Due to $\bar{\beta}_i = o(\alpha_i)$ (cf. Section II-B) and the boundedness of $\frac{Z_l^i(x)}{\beta_i}$, the above recursion is

$$\bar{\phi}_l^{i+1}(x,a) \quad = \quad \bar{\phi}_l^i(x,a) + \alpha_i O(1),$$

implying that it is 'quasi-static' on the scale α_i. Note that, with the present information, the above additive recursion may or may not converge. Even if $\phi^i\to\phi^*$, all components except $a\equiv\xi_{l,x}^*$ (the optimal action in (l,x)) are such that $\bar{\phi}_{l,x}^i(a)\to-\infty$. Yet this is not of consequence, since all we require is that stepsize $\bar{\beta}_i$ be $o(\alpha_i)$. These arguments are detailed in [5] and [3]. Thus, as $i\to\infty$, recursion (3) is supplied with increasingly precise estimates $Q_l^{(i+1)L}(x,a)$ of the $Q-$value $\bar{Q}$ of triplet (l,x,a) under policy ϕ^i

$$\bar{Q}_l(x,a,\phi^i) \quad \triangleq \quad K_l(x,a) + \sum_{y,a} p_l(x,a,y)\cdot$$
$$\phi_{l+1}^i(y,a)\bar{Q}_{l+1}(y,a,\phi^i), \quad (4)$$

with y and a belonging to sets S_{l+1} and $U_{l+1,y}$, respectively.

The results that follow mirror those in [10], with modifications to accommodate use of the dynamic programming principle. Lemma 2 below is a further modification, in that it extends the Cauchy-like property of the KL-entropy $D(\phi_l^i(x),\phi_l^{i+k}(x))\to 0$ to show convergence to a policy element $\bar{\phi}_l(x)$. Take $\phi_l^*(x)$ as the $(l,x)-$th component of the optimal policy ϕ^* with

value 1 on optimal action $\xi_l^*(x)$ and 0 on all other actions. Abusing somewhat the notation $Q_l^{iL+m}(x,a)$ in the algorithm above, consider the random variable $Q_l^j(x,\xi_l^*(x))$ that is an estimate of the reward accrued when $\xi_l^*(x)$ is taken in stage l and the policy ϕ^j followed thence until T. Similarly, let the random variable $V_l(x,\phi^j)$ denote estimates of the $l-$th stage reward using policy ϕ^j. Define $\bar{V}_l(x,\phi^i)$ corresponding to how $\bar{Q}_l(x,a,\phi^i)$ is defined as in (4).

Lemma 1: Keep $\beta \in (1,\infty)$ fixed in SAMW-F. Then, for $0 \leq l \leq T-1$, and $x \in S_l$ with $\phi_l^1(x,a) = \frac{1}{|U_{l,x}|}$, $\forall a \in U_{l,x}$ the sequence of distributions ϕ^i generated by SAMW-D satisfies:

$$\sum_{j=1}^{i} Q_l^j(x,\xi_l^*(x)) \quad \leq \quad \frac{\beta-1}{\ln\beta}\sum_{j=1}^{i} V_l(x,\phi^j) + \frac{\ln|U_{l,x}|}{\ln\beta}$$

Proof: Follows from the proof of Lemma 1 in [10] with minor changes in notation. □

Lemma 2: For β_i that satisfies conditions in Section II-B, the sequence $\phi_l^i(x)$ generated by SAMW-F converge to some $\bar{\phi}_l(x)$

Proof: To the proof of Lemma 2 in [10], we add the following analysis (from Theorem 1 of the same reference). For a given $\epsilon > 0$, $D(\phi^{i+k},\phi^i) \leq \epsilon$ for all $i > T_\epsilon$. Due to finite T, S_l and $U_{l,x}$, $\exists$ a $\tilde{T}_\epsilon$ where $\phi_l^{i+k}(x,a) \leq \phi_l^i(x,a)e^\epsilon \ \forall l,x,a$. Thus, noting that $\phi_l^i(x,a) < 1$, we have $\|\phi_l^{i+k}(x) - \phi_l^i(x)\|_\infty \leq e^\epsilon - 1$, giving us the proof. Alternative norms could be used, in particular, any Euclidean norm on $\mathcal{R}^{|U_{l,x}|}$ suffices. □

We note that a reviewer of this article hinted at an easier proof of the above result via the Csizar inequality of Information Theory.

Theorem 1: For a β_i that satisfies conditions in Section II-B, the sequence of distributions $\phi_l^i(x)$ generated by SAMW-F satisfy:

$$\frac{\beta_i-1}{\ln\beta_i}\frac{1}{i}\sum_{j=1}^{i} V_l(x,\phi^j) + \frac{\ln|U_{l,x}|}{i\ln\beta_i} \quad \to \quad \bar{V}_l(x,\phi^*).$$

Proof: We use a backward induction argument. Consider $l = T-1$, from Lemma 1 we have that $\frac{1}{i}\sum_{j=1}^{i} Q_{T-1}^j(x,\xi_{T-1}^*(x)) \leq$

$$\frac{\beta_i-1}{\ln\beta_i}\frac{1}{i}\sum_{j=1}^{i} V_{T-1}(x,\phi^j) + \frac{\ln|U_{T-1,x}|}{i\ln\beta_i}$$

Using the Law of Large Numbers on the LHS and $\phi^j \to \bar{\phi}$ in RHS (cf. Lemma 2), we have $\bar{Q}_{T-1}(x,\xi_{T-1}^*(x),\bar{\phi}) \leq \bar{V}_{T-1}(x,\bar{\phi})$. An inequality contradicts our assumption of $\phi_{T-1}^*(x)$ being optimal and hence $\bar{\phi}_{T-1}(x) = \phi_{T-1}^*(x)$. We now show for $l = T-2$: using the same argument we have $\bar{Q}_{T-2}(x,\xi_{T-2}^*(x),\bar{\phi}) \leq \bar{V}_{T-2}(x,\bar{\phi})$. Knowing that the $(T-1)-$th component is ϕ_{T-1}^*, we have $\bar{\phi}_{T-2}(x) = \phi_{T-2}^*(x)$. Similarly, we show for $l = T-3,...,0$. □

IV. ALGORITHM SAMW-D

The aim is to find a randomized stationary policy (RSP) $\phi \equiv (\phi_x, \forall x \in S)^T$ that maximizes the infinite-horizon discounted-reward:

$$V(x,\phi) \quad = \quad E\left\{\sum_{l=0}^{\infty}\alpha^l K(X_l,\xi(X_l))|X_0 = x\right\},$$

where discount factor $\alpha \in (0,1)$ and $\xi(x)$ is a U_x-valued random variable generated according to the law $\phi(x)$.

The algorithm SAMW-F extends with only minor changes in notation, primarily removing dependence on the stage marker l. Note also the introduction of discount factor α in (5) below.

Algorithm SAMW-D

- *Step 0 (Initialize): Fix* $\phi^1(x,a) = \frac{1}{|\Pi_x|}$, $\forall x \in S$, $\forall a \in U_x$ *as the initial RSP iterate. Set 'critic' iterates* $Q^k(x,a) = 0$, $\forall x \in S$, $\forall a \in U_x$ *and* $0 \leq k \leq L-1$ *as initial estimates of* $Q-$*values. Set* $i := 1$.
- *Step 1 (Critic): for* $m = 0,1,...,L-1$, *and* $x \in S$, *do*
  - *for each* $a \in U_x$, *do*
    - *Simulate next state* $\eta \equiv \eta^{iL+m}(x,a)$ *according to distribution* $p(x,a,\cdot)$.
    - *Simulate action* $\xi \equiv \xi^{iL+m}(x,a)$ *for* η *according to* $\phi^i(\eta)$.
    - $Q^{iL+m+1}(x,a) := (1-\alpha_i)Q^{iL+m}(x,a) + \alpha_i(K(x,a) + \alpha Q^{iL+m}(\eta,\xi))$. (5)
- *Step 2 (Actor): For each* $x \in S$
  - $Z^i(x) := \sum_{a\in U_x}\phi^i(x,a)\beta_i^{Q^{(i+1)L}(x,a)}$
  - $$\phi^{i+1}(x,a) \quad := \quad \frac{\phi^i(x,a)\beta_i^{Q^{(i+1)L}(x,a)}}{Z^i(x)}$$
  - *Set* $i := i+1$. *If* $i = M$, *go to Step 3; else go to Step 1.*
- *Step 3 (termination): Terminate algorithm and output* ϕ^M *as the final policy.*

All the previous lemmas hold in this case too. However, Theorem 1 uses backward induction and therefore we need a separate proof of convergence:

Theorem 2: For a β_i that satisfies conditions in Section II-B, the sequence $\phi^i(x)$ generated by SAMW-F satisfy:

$$\frac{\beta_i-1}{\ln\beta_i}\frac{1}{i}\sum_{j=1}^{i} V(x,\phi^j) + \frac{\ln|U_x|}{i\ln\beta_i} \quad \to \quad \bar{V}(x,\phi^*).$$

Proof: Analogous to Lemma 1, $\frac{1}{i}\sum_{j=1}^{i} Q^j(x,\xi^*(x))$

$$\leq \quad \frac{\beta_i-1}{\ln\beta_i}\frac{1}{i}\sum_{j=1}^{i} V(x,\phi^j) + \frac{\ln|U_x|}{i\ln\beta_i}.$$

From Lemma 2 we have that $\phi^j \to \bar{\phi}$, and $\bar{Q}(x,\xi^*(x)) \leq \bar{V}(x,\bar{\phi})$, $\forall x \in S$. Notice that, w.p. 1,

$$\bar{Q}(x,\xi^*(x),\bar{\phi}) \quad = \quad K(x,\xi^*(x)) + \alpha\sum_{y\in S} p(x,\xi^*(x),y)\bar{V}(y,\bar{\phi}).$$

Substituting $\bar{Q}(y,\xi^*(y),\bar{\phi})$ for $\bar{V}(y,\bar{\phi})$, (i.e. we take optimal action $\xi^*(y)$ in every state y seen after x), we have $\bar{Q}(x,\xi^*(x),\bar{\phi}) \geq K(x,\xi^*(x)) + \alpha\sum_{y\in S} p(x,\xi^*(x),y)\bar{Q}(y,\xi^*(y),\bar{\phi})$. Repeated substitution yields $\bar{Q}(x,\xi^*(x),\bar{\phi}) \geq \bar{V}(x,\phi^*)$ where inequality cannot hold and equality holds if $\bar{\phi} = \phi^*$. □

V. Simulation Results

A. Capacity Switching in Semiconductor Fabs

We first briefly describe the model of a semiconductor fabrication unit, considered in [11]. The factory process $\{X_l|X_0 = i\}, 1 \leq l \leq T$, $i \in S_0$ is such that each X_l is a vector of capacities at time epochs $l \in \{1, 2, ..., T\}$, the components $X_l^{(A,i),w}$ representing the number of type w machines allocated to performing operation i on product A. The duration of the planning horizon is T, casting this problem as a finite horizon Markov decision process. A type w machine indicates a machine capable of performing all operations which are letters in the 'word' w. Note that the product A requires the operation i for completion and that the word w contains the letter i, among others. The word w of a machine can also contain the action 0 - indicating idling. The control $\pi_l(X_l)$ taken at stages $0 \leq l \leq T-1$ would be to switch $u_l^{(A,i),w,(B,j)}$ machines of type w from performing operation i on product A to performing operation j on product B.

As in inventory control models, the randomness in the system is modeled by the demand D_r^A for product A at stages $0 \leq l \leq T-1$. The per-step transition cost consists of costs for excess inventory (K_A^e, for every unit of product A in inventory), backlog (K_A^b), cost of operation (K_w^o, one-stage operating cost for a machine of type w) and the cost of switching capacity from one type of operation to another (K_w^s, the cost of switching a machine of type w from one type of production to another). In all these costs, further complications can be admitted, e.g., the cost K_w^s could be indexed with the source product-operation pair (A, i) and the destination pair (B, j).

We adopt a finite horizon of $T = 5$ and consider a fab producing two products (A and B), both requiring two operations, 'litho' and 'etch' (call these L and E). We have a fab with 2 litho and 2 etch machines that can each perform the corresponding operation on either A or B. We denote the operations $\{A, L\}$ (i.e., 'litho on A') as 1, $\{A, E\}$ as 2, $\{B, L\}$ as 3 and $\{B, E\}$ as 4, implying that the word w of a litho machine is 013 and that of an etch machine is 024. The product-operation pairs that admit non-zero capacities are therefore: $(A, 1)$, $(A, 2)$, $(B, 3)$, and $(B, 4)$.

The fab's throughput $P_l = (P_l^A, P_l^B)$ is constrained by the following relation: $P_l^A = min(X_l^{(A,1),013}, X_l^{(A,2),024})$ and $P_l^B = 0.5 \cdot min(X_l^{(B,3),013}, X_l^{(B,4),024})$, $0 \leq l \leq T-1$ indicating the slower production of B. We assume that no machine is idling and therefore the capacity allocated to product B, given capacity allocated to A, is simply the remainder from 2 for both 'litho' and 'etch'. Therefore, these are $X_l^{(B,3),013} = 2 - X_l^{(A,1),013}$, $X_l^{(B,4),024} = 2 - X_l^{(B,4),024}$. We also constrain the inventories: $I_l^A \in \{-1, 0, 1\}$, $I_l^B \in \{-0.5, 0, 0.5\}$, $0 \leq l \leq T-1$. The state is thus completely described by the quadruplet: $X_l = (X_l^{(A,1),013}, X_l^{(A,2),024}, I_l^A, I_l^B), 0 \leq l \leq T$. One can choose a lexicographic order among the possible starting states X_0, which are $3^4 = 81$ in number.

Capacity switching at stage $0 \leq l \leq T-1$ is specified by the policy component $\pi_{l,.}$. Thus $\pi_l(X_l)$ is a vector such that $\pi_l(X_l) = (\pi_l(X_l, 1), \pi_l(X_l, 2))$, making $X_{l+1}^{(A,1),013} = X_l^{(A,1),013} + \pi_l(X_l, 1)$ (similarly for $X_{l+1}^{(A,2),024}$). Note that controls $\pi_l(X_l)$ belong to the feasible action set U_{l,X_l}, and that in our setting $0 \leq |U_{l,X_l}| \leq 9$, the state 'group' $(1, 1, x_A, y_B)$, $\forall x_A \in \{-1, 0, 1\}$, $\forall y_B \in \{-0.5, 0, 0.5\}$ having the maximum of 9 actions. We take the various one-step costs to be $K_A^e = 2$, $K_B^e = 1$, $K_A^b = 4$, $K_B^e = 2$, $K_{013}^o = 1.5$, $K_{024}^o = 1$, $K_{013}^s = 2$ and $K_{024}^s = 2$. The noise (demand) scheme is such that: $D_l^A = 1$ w.p. 0.4, $D_l^A = 2$ otherwise, and $D_l^B = 0.5$ w.p. 0.7, $D_l^B = 1$ otherwise, for all $0 \leq l \leq T-1$. We assume that the control $\pi_l(X_l)$ is applied to the state X_l at the beginning of the epoch l whereas the demand D_l is made at the end of it, i.e., $X_{l+1}^{(A,1),013} = X_l^{(A,1),013} + \pi_l(X_r, 1)$, $X_{l+1}^{(A,2),024} = X_l^{(A,2),024} + \pi_l(X_l, 2)$, $I_{l+1}^A = max(min(I_l^A + P_{l+1}^A - D_l^A, 1), -1)$, $I_{l+1}^A = max(min(I_l^B + P_{l+1}^B - D_l^B, 0.5), -0.5))$. We also take the terminal cost $K_T(X_T)$ to be 0. Thus, the per-stage cost $\bar{K}_l(X_l, \pi_l(X_l), X_{l+1}) = 2K_{013}^o + 2K_{024}^o$

$$
\begin{aligned}
&+ \pi_l(X_l, 1) \cdot K_{013}^s + \pi_l(X_l, 2) \cdot K_{024}^s \\
&+ max(I_{l+1}^A, 0) \cdot K_A^e + max(I_{l+1}^B, 0) \cdot K_B^e \\
&+ max(-I_{l+1}^B, 0) \cdot K_A^b + max(-I_{l+1}^B, 0) \cdot K_B^b
\end{aligned}
$$

implies $K_l(X_l, \pi_l(X_l)) = \bar{K}_l^{-1}(X_l, \pi_l(X_l), X_{l+1})$. A check indicates this reward satisfies Assumption 1.

Both algorithms were terminated at the iteration i when $\text{err}_i \leq 0.01$. This criterion is described as:

$$
\text{err}_i = \max_{x \in S, k \in \{1,2,\ldots,50\}} \|\pi^i(x) - \pi^{i-k}(x)\|_2,
$$

where $\pi^i(x) = (\pi_l^i(x, a), 0 \leq l \leq T-1, a \in U_{l,x})^T$, and $\|\cdot\|_2$ is the Euclidean norm in the appropriate dimension $\sum_{l=0}^{T-1} |U_{l,x}|$.

The total rewards obtained using the proposed SAMW-F and RPAFA-2 (a two-simulation variant of the algorithm in [5]) is plotted in Figure 1. The states are sorted in decreasing order of the exact reward (computed using DP). The reward computed using the converged policy of RPAFA-2 is reliably close - albeit lower than DP. The advantage lies in the faster convergence time of SAMW-F, computing performances being outlined in Table I. Note the proportionally less time that SAMW-F takes, a consequence of the absence of probability vector projection. In the experiments we chose the averaging parameter L as 100 for the RPAFA-2 algorithm. To approximately adjust for the number of table look-ups, we chose $L = 20$ for SAMW, i.e. 20 transitions for each $a \in U_{l,x}$ are simulated from the pair (l, x) (the adjustment is approximate since $|U_{l,x}|$ varies). We used the step-sizes $\alpha_i = \frac{1}{i^{0.55}}$ and $\beta_i = 1 + \frac{1}{i^{0.60}}$ in SAMW-F.

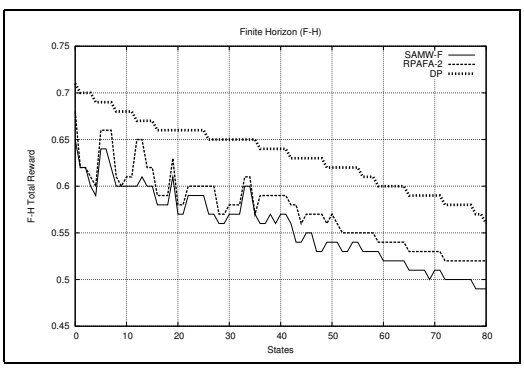

Fig. 1: Comparing Finite Horizon Rewards

	Iter. i	Time in sec.	err_i	Max Error
SAMW-F	100	5	0.01	0.10
RPAFA-2	900	85	0.01	0.09

TABLE I: Comparison of SAMW-F with RPAFA-2

	Iter. i	Time in sec.	err_i
SAMW-D	100	2	0.01
RPAFA-2	220	19	0.01

TABLE II: Comparison of SAMW-D and RPAFA-2

B. Flow control in communication networks

For the proposed SAMW-D algorithm, we consider a continuous time queuing model of flow control. The numerical setting here is somewhat similar to that in [3]. Assume that a single bottleneck node has a finite buffer of size B. Packets are fed into the node by both an uncontrolled Poisson arrival stream with rate $\lambda_u = 0.2$, and a controlled Poisson process with rate $\lambda_c(t)$ at instant $t > 0$. Service times at the node are i.i.d., exponentially distributed with rate 2.0. We assume that the queue length process $\{X_t, t > 0\}$ at the node is continually observed every $\tilde{T}$ instants, for some $\tilde{T} > 0$. Suppose X_l denotes the queue length observed at instant $l\tilde{T}$, $l \geq 0$. This information is fed back to the controlled source which then starts sending packets at $\lambda_c(X_l)$ in the interval $[l\tilde{T}, (l+1)\tilde{T})$, assuming there are no feedback delays. We use $B = 50$, and designate the 'target state' $\hat{T} = 25$ representing the middle of the buffer. The goal is to maximize throughput and minimize the queue length and, therefore, the delay in a queueing system.

The one-step reward under a given randomized stationary policy π is computed as $K(X_l, \xi_l, X_{l+1}) = \frac{1-\alpha}{\max(|X_{l+1}-\hat{T}|,1)}$ where ξ_l is a random variable with law $\phi(X_l)$ and $0 < \alpha < 1$ is the discount factor. Such a cost function penalizes states away from the target state $\hat{T}$. For the current setting where $|U_x| < \infty$, we discretize the interval $[0.05, 4.5]$ so as to obtain five equally spaced actions in each state. For purposes of comparison, we also implemented the Dynamic Programming algorithm. However, one would then require the transition probability matrix $P_{\tilde{T}}$ in order to compute which, we use the approximation method suggested in [12, §6.8]. The calculation is complex: take q (e.g. $q = 5$ here) controls per state, then q number of $P_{\tilde{T}}$ matrices of size $B \times B$ each are required. Also note that the amount of computation required for $P_{\tilde{T}}$ also depends upon the convergence criteria specified for the method in [12, §6.8]. Besides, such probabilities can only be computed for systems whose dynamics are well known.

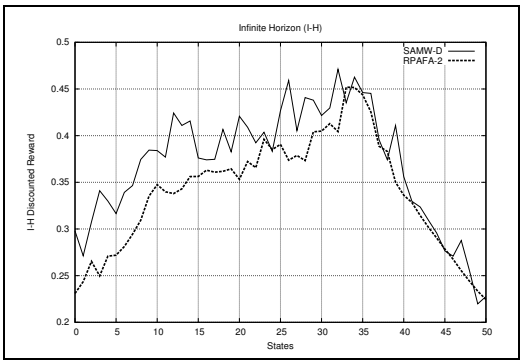

Fig. 2: Comparing Infinite-Horizon Rewards

The comparable algorithm RPAFA-2 (a variant of the RPAFA-2 in [6]) is terminated at iteration i where $\text{err}_i \leq 0.01$, criteria err_i being: $\text{err}_i = \max_{x \in S, k \in \{1,2,\ldots,50\}} \|\phi^i(x,a) - \phi^{i-k}(x,a)\|_2$, where $\phi^i(x) = (\phi^i(x,a), a \in U_x)^T$. Again, $\|\cdot\|_2$ being the Euclidean norm in $\mathcal{R}^q$. Shown in Figure 2 is the discounted reward for each state in a system operating under the optimal policy computed using SAMW-D and RPAFA-2, respectively. The plots shown in Figure 2 are obtained using stochastic approximation from 1×10^4 transitions from each state $x \in S$ to compute the value functions. Tables II shows performance comparisons of SAMW-D with RPAFA-2. The proposed SAMW-D converges much faster, despite $L = 20$ in order to adjust for number of table look-ups.

VI. Future Directions

It appears possible to eliminate the large table required to store the Q-values. This can be done by having only estimates of $\tilde{V}(x, \phi^i)$ stored in the look-up table and estimating $Q-$values by simulation.

Acknowledgments

This work was supported in part by Grant no. SR/S3/EE/43/2002-SERC-Engg from the Department of Science and Technology, Government of India.

References

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

[2] V. Konda and J. Tsitsiklis, "Actor–Critic Algorithms," *SIAM Journal on Control and Optimization*, vol. 42, no. 4, pp. 1143–1166, 2003.

[3] 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.

[4] V. Konda and V. Borkar, "Actor–Critic Type Learning Algorithms for Markov Decision Processes," *SIAM Journal on Control and Optimization*, vol. 38, no. 1, pp. 94–123, 1999.

[5] S. Bhatnagar and M. S. Abdulla, "An Actor-Critic Algorithm for Finite Horizon Markov Decision Processes," in *Proceedings of the 45th IEEE-CDC, Dec 11-13 2006, San Diego, CA, USA*, 2006.

[6] M. S. Abdulla and S. Bhatnagar, "Reinforcement learning based algorithms for average cost markov decision processes," *Accepted for publication in Discrete Event Dynamical Systems*, 2006.

[7] S. Bhatnagar and M. Abdulla, "An Actor-Critic Algorithm for Finite Horizon Markov Decision Processes," *Submitted*, 2006.

[8] V. Raju Ch, *Learning Dynamic Prices In Electronic Markets*. PhD Thesis: Department of CSA, IISc, Bangalore, 2004.

[9] D. Bertsekas, *Dynamic Programming and Optimal Control, Volume I*. Belmont, MA: Athena Scientific, 1995.

[10] H. S. Chang, M. C. Fu, and S. I. Marcus, "An Asymptotically Efficient Algorithm for Finite Horizon Stochastic Dynamic Programming Problems," in *Proceedings of the 42nd IEEE-CDC, Dec 2003, Maui, HI, USA*, 2003.

[11] J. Panigrahi and S. Bhatnagar, "Hierarchical Decision Making in Semiconductor Fabs using Multi-time Scale Markov Decision Processes," in *Proceedings of IEEE Conference on Decision and Control, Paradise Island, Nassau, Bahamas*, 2004.

[12] S. M. Ross, *Introduction to Probability Models, 7/e*. San Diego, CA: Academic Press, 2000.