2

Multi-stage Adaptive Sampling Algorithms

In this chapter, the goal is to accurately and efficiently estimate the optimal value function under the constraint that there is a finite number of simulation replications to be allocated per state in stage i. The straightforward approach to this would be simply to sample each action feasible in a state equally, but this is clearly not an efficient use of computational resources, so the main question to be decided is which action to sample next. The algorithms in this chapter adaptively choose which action to sample as the sampling process proceeds, based on the estimates obtained up to that point, and lead to value function estimators that converge to the true value asymptotically in the number of simulation replications allocated per state. These algorithms are targeted at MDPs with large, possibly *uncountable*, state spaces and relatively smaller *finite* action spaces. The primary setting in this chapter will be *finite-horizon* models, which lead to a recursive structure, but we also comment on how the algorithms can be used for infinite-horizon problems. Numerical experiments are used to illustrate the algorithms.

Once we have an algorithm that estimates the optimal value/policy for finite-horizon problems, we can create a nonstationary randomized policy in an on-line manner in the context of receding-horizon control for solving infinite-horizon problems. This will be discussed in detail in Chapter 5.

Letting $\hat{V}_i^{N_i}(x)$ denote the estimate of the optimal reward-to-go function, $V_i^*(x)$, defined by Equation (1.5) for a given state x and stage i, based on N_i simulations in stage i, the objective is to estimate the optimal value $V^*(x_0)$ for a given starting state x_0, as defined by Equation (1.2). The approach will be to optimize over actions, based on the recursive optimality equations given by (1.8) and (1.17). The former involves an optimization over the action space, so the main objective of the approaches in this chapter is to adaptively determine which action to sample next. Using a random number w, the chosen action will then be used to simulate $f(x, a, w)$ in order to produce a simulated next state from x. This is used to update the estimate of $Q_i^*(x, a)$, which will be called the Q-function estimate and denoted by $\hat{Q}_i^{N_i}(x, a)$, which in

General Adaptive Multi-stage Sampling Framework

Input: stage $i < H$, state $x \in X$, $N_i > 0$, other parameters.
(For $i = H$, $\hat{V}_H^{N_H}(x) = V_H^{N_H}(x) = 0$.)

Initialization: algorithm parameters; total number of simulations set to 0.

Loop until total number of simulations reaches N_i:

- Determine an action $\hat{a}$ to simulate next state via $f(x, \hat{a}, w),\ w \sim U(0, 1)$.
- Update the following:
  number of times action a has been sampled $N_{\hat{a}}^i(x) \leftarrow N_{\hat{a}}^i(x) + 1$,
  Q-function estimate $\hat{Q}_i^{N_i}(x, \hat{a})$ based on $R'(x, \hat{a}, w)$ and $\hat{V}_{i+1}^{N_{i+1}}(f(x, \hat{a}, w))$,
  the current optimal action estimate (for state x in stage i),
  and other algorithm-specific parameters.

Output: $\hat{V}_i^{N_i}(x)$ based on Q-function estimates $\{\hat{Q}_i^{N_i}(x, a)\}$.

Fig. 2.1. Adaptive multi-stage sampling framework

turn determines the estimate $\hat{V}_i^{N_i}(x)$, albeit not necessarily using Equation (1.8) as the estimate for the optimal value function. Figure 2.1 provides a generic algorithm outline for the adaptive multi-stage sampling framework of this chapter.

Specifically, $Q_i^*(x, a)$ is estimated for each action $a \in A(x)$ by a sample mean based on simulated next states and rewards from a fixed state x:

$$\hat{Q}_i^{N_i}(x, a) = \frac{1}{N_a^i(x)} \sum_{j=1}^{N_a^i(x)} \left[R'(x, a, w_j^a) + \gamma \hat{V}_{i+1}^{N_{i+1}}(f(x, a, w_j^a)) \right], \qquad (2.1)$$

where $N_a^i(x)$ is the number of times action a has been sampled from state x in stage i ($\sum_{a \in A(x)} N_a^i(x) = N_i$), and the sequence $\{w_j^a, j = 1, ..., N_a^i(x)\}$ contains the corresponding random numbers used to simulate the next states $f(x, a, w_j^a)$. Note that the number of next-state samples depends on the state x, action a, and stage i.

In the general framework that estimates the Q-function via (2.1), the total number of sampled (next) states is $O(N^H)$ with $N = \max_{i=0,...,H-1} N_i$, which is independent of the state space size. One approach is to select "optimal" values of $N_a^i(x)$ for $i = 0, ..., H-1$, $a \in A(x)$, and $x \in X$, such that the expected error between the values of $\hat{V}_0^{N_0}(x)$ and $V_0^*(x)$ is minimized, but this problem would be difficult to solve. Both algorithms in this chapter construct a sampled tree in a recursive manner to estimate the optimal value at an initial state and incorporate an adaptive sampling mechanism for selecting which action to sample at each branch in the tree. The upper confidence bound (UCB) sampling algorithm chooses the next action based on the exploration-exploitation tradeoff captured by a multi-armed bandit model, whereas in the

pursuit learning automata (PLA) sampling algorithm, the action is sampled from a probability distribution over the action space, where the distribution tries to concentrate mass on ("pursue") the estimate of the optimal action. The analysis of the UCB sampling algorithm is given in terms of the expected bias, whereas for the PLA sampling algorithm we provide a probability bound. Another algorithm that also uses a distribution over the action space but updates the distribution in a different manner using multiple samples, and can handle infinite action spaces, is presented in Section 4.5.

2.1 Upper Confidence Bound Sampling

The UCB sampling algorithm is based on the expected regret analysis for multi-armed bandit problems, in which the sampling is done based on upper confidence bounds generated by simulation-based estimates. The UCB algorithm determines $N_a^i(x)$ for $i = 0, ..., H-1$, $a \in A(x)$, and $x \in X$ such that the expected difference is bounded as a function of $N_a^i(x)$ and N_i, $i = 0, ..., H-1$, and such that the bound (from above and from below) goes to zero as N_i, $i = 0, ..., H-1$, go to infinity. The allocation rule (sampling algorithm) adaptively chooses which action to sample, updating the value of $N_a^i(x)$ as the sampling process proceeds, such that the value function estimator is asymptotically unbiased (i.e., $E[\hat{V}_0^{N_0}(x)] \rightarrow V_0^*(x)$ as $N_i \rightarrow \infty, \forall\ i = 0, ..., H-1$), and an upper bound on the bias converges to zero at rate $O(\sum_i \frac{\ln N_i}{N_i})$, where the logarithmic bound in the numerator is achievable uniformly over time. The running-time complexity of the algorithm is at worst $O\left((|A| \max_{i=0,...,H-1} N_i)^H\right)$, which is independent of the state space size, but depends on the size of the action space, because the algorithm requires that each action be sampled at least once for each sampled state.

2.1.1 Regret Analysis in Multi-armed Bandits

The goal of the multi-armed bandit problem is to play as often as possible the machine that yields the highest (expected) reward. The regret quantifies the exploration/exploitation dilemma in the search for the true "optimal" machine, which is unknown in advance. The goal of the search process is to explore the reward distribution of different machines while also frequently playing the machine that is empirically best thus far. The regret is the expected loss due to not always playing the true optimal machine. For an optimal strategy the regret grows at least logarithmically in the number of machine plays, and the logarithmic regret is also achievable uniformly over time with a simple and efficient sampling algorithm for arbitrary reward distributions with bounded support.

Specifically, an M-armed bandit problem is defined by random variables $\eta_{i,j}$ for $1 \leq i \leq M$ and $j \geq 1$, where successive plays of machine i yield

"rewards" $\eta_{i,1}, \eta_{i,2}, ...$, which are independent and identically distributed according to an unknown but fixed distribution η_i with unknown expectation μ_i, and the goal is to decide the machine i at each play to maximize

$$E\left[\sum_{j=1}^{n} \eta_{i,j}\right].$$

The rewards across machines are also independently generated. Let $T_i(n)$ be the number of times machine i has been played by an algorithm during the first n plays. Define the *expected regret* $\rho(n)$ of an algorithm after n plays by

$$\rho(n) = \mu^* n - \sum_{i=1}^{M} \mu_i E[T_i(n)], \text{ where } \mu^* := \max_i \mu_i.$$

Any algorithm that attempts to minimize this expected regret must play a best machine (one that achieves μ^*) exponentially (asymptotically) more often than the other machines, leading to $\rho(n) = \Theta(\ln n)$. One way to achieve the asymptotic logarithmic regret is to use upper confidence bounds, which capture the tradeoff between exploitation – choosing the machine with the current highest sample mean – and exploration – trying other machines that might have higher actual means. This leads to an easily implementable algorithm in which the machine with the current highest upper confidence bound is chosen.

We incorporate these results into a sampling-based process for finding an optimal action in a state for a single stage of an MDP by appropriately converting the definition of regret into the difference between the true optimal value and the approximate value yielded by the sampling process. We then extend the one-stage sampling process into multiple stages in a recursive manner, leading to a multi-stage (sampling-based) approximation algorithm for solving MDPs.

2.1.2 Algorithm Description

Figure 2.2 presents the upper confidence bound (UCB) adaptive sampling algorithm for estimating $V_0^*(x)$ for a given state x. The inputs to the algorithm are the stage i, a state $x \in X$, and the number of samples $N_i \geq \max_{x \in X} |A(x)|$, and the output is $\hat{V}_i^{N_i}(x)$, the estimate of the optimal reward-to-go value from state x, $V_i^*(x)$, given by (2.5), which is the weighted average of Q-value estimates over the sampled actions. (Alternative optimal value function estimators are presented in Section 2.1.3.) Since the Q-function estimate given by (2.1) requires the optimal value estimate $\hat{V}_{i+1}^{N_{i+1}}(y)$ for the simulated next state $y \in X$ in the next period $i+1$, the algorithm requires recursive calls at (2.2) and (2.4) in the **Initialization** and **Loop** portions of the algorithm, respectively. The initial call to the algorithm is done with $i = 0$, the initial

state x_0, and N_0, and every sampling is done independently of previous samplings. To help understand how the recursive calls are made sequentially, in Figure 2.3, we graphically illustrate the sequence of calls with two actions and $H = 3$ for the **Initialization** portion.

For an intuitive description of the allocation rule, consider first only the one-stage approximation. That is, we assume for now that the $V_1^*(x)$-value for each sampled state $x \in X$ is known. To estimate $V_0^*(x)$, obviously we need to estimate $Q_0^*(x, a^*)$, where $a^* \in \arg\max_{a \in A(x)}(Q_0^*(x, a))$. The search for a^* corresponds to the search for the best machine in the multi-armed bandit problem. We start by sampling a random number $w^a \sim U(0, 1)$ for each possible action once at x, which leads to the next (sampled) state $f(x, a, w^a)$ according to f and reward $R'(x, a, w^a)$. We then iterate as follows (see **Loop** in Figure 2.2). The next action to sample is the one that achieves the maximum among the current estimates of $Q_0^*(x, a)$ plus its current upper confidence bound (cf. (2.3)), where the estimate $\hat{Q}_0^{N_0}(x, a)$ is given by the sample mean of the immediate reward plus V_1^*-values (multiplied by the discount factor) at all of the simulated next states (cf. Equation (2.4)).

Among the N_0 samples for state x, $N_a^0(x)$ denotes the number of samples using action a. If the sampling is done appropriately, we might expect that $N_a^0(x)/N_0$ provides a good estimate of the likelihood that action a is optimal in state x, because in the limit as $N_0 \to \infty$, the sampling scheme should lead to $N_{a^*}^0(x)/N_0 \to 1$ if a^* is the unique optimal action, or if there are multiple optimal actions, say a set A^*, then $\sum_{a \in A^*} N_a^0(x)/N_0 \to 1$, i.e., $\left\{N_a^0(x)/N_0\right\}_{a \in A(x)}$ should converge to a probability distribution concentrated on the set of optimal actions. For this reason, we use a weighted (by $N_a^0(x)/N_0$) sum of the currently estimated value of $Q_0^*(x, a)$ over $A(x)$ to approximate $V_0^*(x)$ (cf. Equation (2.5)). Ensuring that the weighted sum concentrates on a^* as the sampling proceeds will ensure that in the limit the estimate of $V_0^*(x)$ converges to $V_0^*(x)$.

The running-time complexity of the UCB adaptive sampling algorithm is $O((|A|N)^H)$, where $N = \max_i N_i$. To see this, let M_i be the number of recursive calls made to compute $\hat{V}_i^{N_i}$ in the *worst* case. At stage i, the algorithm makes at most $M_i = |A|N_i M_{i+1}$ recursive calls (in **Initialization** and **Loop**), leading to $M_0 = O((|A|N)^H)$. In contrast, backward induction has $O(H|A||X|^2)$ running-time complexity. Therefore, the main benefit of the UCB sampling algorithm is independence from the state space size, but this comes at the expense of exponential (versus linear, for backwards induction) dependence on both the action space and the horizon length.

2.1.3 Alternative Estimators

We present two alternative estimators to the optimal reward-to-go value function estimator given by Equation (2.5) in the UCB sampling algorithm. First, consider the estimator that replaces the weighted sum of the Q-function estimates in Equation (2.5) by the maximum of the estimates, i.e., for $i < H$,

Upper Confidence Bound (UCB) Sampling Algorithm

Input: stage $i < H$, state $x \in X$, $N_i \geq \max_{x \in X} |A(x)|$.
(For $i = H$, $\hat{V}_H^{N_H}(x) = V_H^{N_H}(x) = 0$.)

Initialization: Simulate next state $f(x, \hat{a}, w_1^a)$, $w_1^a \sim U(0,1)$ for each $a \in A(x)$; set $N_a^i(x) = 1 \ \forall a \in A(x)$, $\bar{n} = |A(x)|$, and

$$\hat{Q}_i^{N_i}(x,a) = M_i(x,a) = R'(x,a,w_1^a) + \gamma \hat{V}_{i+1}^{N_{i+1}}(f(x,a,w_1^a)) \ \ \forall a \in A(x), \quad (2.2)$$

where $\{w_j^a\}$ is the random number sequence for action a,
$N_a^i(x)$ is the number of times action a has been sampled thus far,
and $\bar{n}$ is the overall number of samples thus far.

Loop until $\bar{n} = N_i$:

- Generate $w^{\hat{a}}_{N^i_{\hat{a}}(x)+1} \sim U(0,1)$ for current estimate of optimal action a^*:

$$\hat{a} \in \arg\max_{a \in A(x)} \left(\hat{Q}_i^{N_i}(x,a) + R_{\max}(H-i)\sqrt{\frac{2 \ln \bar{n}}{N_a^i(x)}} \right), \quad (2.3)$$

  where

$$\hat{Q}_i^{N_i}(x,a) = \frac{1}{N_a^i(x)} \sum_{j=1}^{N_a^i(x)} \left[R'(x,a,w_j^a) + \gamma \hat{V}_{i+1}^{N_{i+1}}(f(x,a,w_j^a)) \right]. \quad (2.4)$$

- Update Q-function estimate for $a = \hat{a}$ using simulated next state $f(x, \hat{a}, w^{\hat{a}}_{N^i_{\hat{a}}(x)+1})$:

$$\begin{aligned} M_i(x,\hat{a}) &\leftarrow M_i(x,\hat{a}) \\ &\quad + R'(x,\hat{a},w^{\hat{a}}_{N^i_{\hat{a}}(x)+1}) + \hat{V}_{i+1}^{N_{i+1}}(f(x,\hat{a},w^{\hat{a}}_{N^i_{\hat{a}}(x)+1})), \\ N_{\hat{a}}^i(x) &\leftarrow N_{\hat{a}}^i(x) + 1, \\ \hat{Q}_i^{N_i}(x,\hat{a}) &\leftarrow \frac{M_i(x,\hat{a})}{N_{\hat{a}}^i(x)}. \end{aligned}$$

- $\bar{n} \leftarrow \bar{n} + 1$.

Output:

$$\hat{V}_i^{N_i}(x) = \sum_{a \in A(x)} \frac{N_a^i(x)}{N_i} \hat{Q}_i^{N_i}(x,a). \quad (2.5)$$

Fig. 2.2. Upper confidence bound (UCB) sampling algorithm description

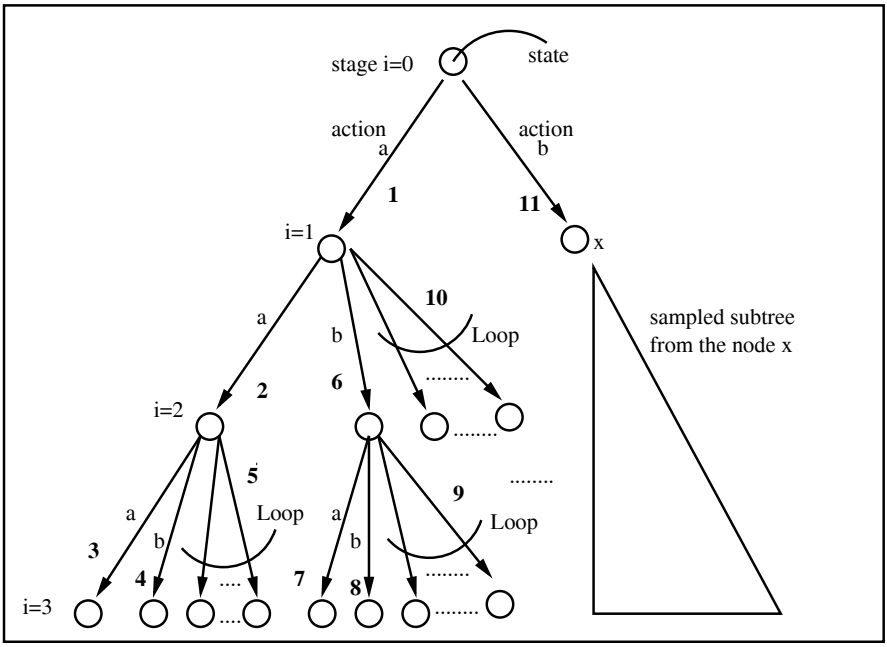

Fig. 2.3. Graphical illustration of a sequence of recursive calls made in **Initialization** of the UCB sampling algorithm, where each circle corresponds to a simulated state, each arrow with associated action signifies a sampling for the action (and a recursive call), and the boldface number near each arrow indicates the sequencing for the recursive calls (for simplicity, an entire **Loop** process is signified by a single number)

$$\hat{V}_i^{N_i}(x) = \max_{a \in A(x)} \hat{Q}_i^{N_i}(x, a). \tag{2.6}$$

For the nonadaptive case, it can be shown that this estimator is also asymptotically unbiased, but with a finite-sample "optimistic" bias in the opposite direction as the original estimator (i.e., upwards for maximization problems and downwards for minimization problems such as the inventory control problem).

Next, consider an estimator that chooses the action that has been sampled the most thus far in order to estimate the value function. It can be easily shown that this estimator is less optimistic than the previous alternative, and so combining it with the original estimator gives the following estimator:

$$\hat{V}_i^{N_i}(x) = \max \left\{ \hat{Q}_i^{N_i}(x, \hat{a}), \sum_{a \in A(x)} \frac{N_a^i(x)}{N_i} \hat{Q}_i^{N_i}(x, a) \right\}, \quad \hat{a} \in \arg\max_a \{N_a^i(x)\}, \tag{2.7}$$

which would again replace Equation (2.5) in the algorithm. Intuitively, the rationale behind combining via the max operator is that the estimator would be choosing the best between two possible estimates of the Q-function.

It is conjectured that all of these alternatives are asymptotically unbiased, with the estimator given by Equation (2.6) having an "optimistic" bias (i.e., high for maximization problems, low for minimization problems). If so, valid, albeit conservative, confidence intervals for the optimal value could also be easily derived by combining the two oppositely biased estimators. Such a result can be established for the nonadaptive versions of these estimators, but proving these results in our setting and characterizing the convergence rate of the estimator given by Equation (2.6) in a similar manner as for the original estimator is considerably more difficult, so we restrict our convergence analysis to the original estimator.

2.1.4 Convergence Analysis

Now we show the convergence properties of the UCB sampling algorithm. In particular, we show that the final estimate of the optimal value function generated by the algorithm is asymptotically unbiased, and the bias can be shown to be bounded by a quantity that converges to zero at rate $O\left(\sum_{i=0}^{H-1} \frac{\ln N_i}{N_i}\right)$.

We start with a convergence result for the one-stage approximation. Consider the following one-stage sampling algorithm (OSA) in Figure 2.4 with a *stochastic value function* U defined over X, where $U(x)$ for $x \in X$ is a *nonnegative random variable* with *unknown* distribution and bounded above for all $x \in X$. As before, every sampling is done independently, and we assume that there is a black box that returns $U(x)$ once x is given to the black box. Fix a state $x \in X$ and index each action in $|A(x)|$ by numbers from 1 to $|A(x)|$. Consider an $|A(x)|$-armed bandit problem where each a is a gambling machine. Successive plays of machine a yield "bandit rewards" that are i.i.d. according to an unknown distribution η_a with unknown expectation

$$Q(x,a) = E[R'(x,a,w) + \gamma E[U(f(x,a,w))]], w \sim U(0,1)$$

and are independent across machines or actions. The term $T_a^x(n)$ signifies the number of times machine a has been played (or random number for action a has been sampled) by OSA during the n plays. Define the *expected regret* $\rho(n)$ of OSA after n plays by

$$\rho(n) = V(x)n - \sum_{a=1}^{|A(x)|} Q(x,a)E[T_a^x(n)],$$

where $V(x) = \max_{a \in A(x)} Q(x,a)$, and let

$$U_{\max} = \max_{x,a} Q(x,a) = \max_x V(x).$$

We now state a key theorem in [4], which will be the basis of our convergence results for the OSA algorithm.

One-stage Sampling Algorithm (OSA)

Input: state $x \in X$ and $n \geq |A(x)|$.

Initialization: Simulate next state $f(x, a, w_1^a), w_1^a \sim U(0,1)$ for each $a \in A(x)$; set $T_a^x(\bar{n}) = 1\ \forall a \in A(x)$, $\bar{n} = |A(x)|$, and

$$\tilde{Q}(x,a) = R'(x,a,w_1^a) + \gamma U(f(x,a,w_1^a)) \quad \forall a \in A(x),$$

where $\{w_j^a\}$ is the random number sequence for action a,
$T_a^x(\bar{n})$ is the number of times action a has been sampled thus far,
and $\bar{n}$ is the overall number of samples thus far.

Loop until $\bar{n} = n$:

- Generate $w_{T_{\hat{a}}^x(\bar{n})+1}^{\hat{a}^*} \sim U(0,1)$ for current estimate of optimal action:

$$\hat{a} \in \arg\max_{a \in A(x)} \left(\tilde{Q}(x,a) + U_{\max} \sqrt{\frac{2 \ln \bar{n}}{T_a^x(\bar{n})}} \right),$$

where

$$\tilde{Q}(x,a) = \frac{1}{T_a^x(\bar{n})} \sum_{j=1}^{T_a^x(\bar{n})} \left[R'(x,a,w_j^a) + \gamma U(f(x,a,w_j^a)) \right]. \tag{2.8}$$

- Update Q-function estimate for $a = \hat{a}$ via (2.8) using simulated next state $f(x, \hat{a}, w_{T_{\hat{a}}^x(\bar{n})+1}^{\hat{a}})$, with $T_{\hat{a}}^x(\bar{n}) \leftarrow T_{\hat{a}}^x(\bar{n}) + 1$.
- $\bar{n} \leftarrow \bar{n} + 1$.

Output:

$$\tilde{V}^n(x) = \sum_{a \in A(x)} \frac{T_a^x(n)}{n} \tilde{Q}(x,a). \tag{2.9}$$

Fig. 2.4. One-stage sampling algorithm (OSA) description

Theorem 2.1. *For any x with $|A(x)| > 1$, if OSA is run on $|A(x)|$-machines having arbitrary bandit reward distributions $\eta_1, ..., \eta_{|A(x)|}$ with finite $U_{\max}$, then*

$$\rho(n) \leq \sum_{a:Q(x,a)<V(x)} \left[\frac{8U_{\max}^2 \ln n}{V(x) - Q(x,a)} + \left(1 + \frac{\pi^2}{3}\right) (V(x) - Q(x,a)) \right],$$

where

$$V(x) = \max_{a \in A(x)} \Big(E[R'(x,a,w) + \gamma E[U(f(x,a,w))]] \Big), w \sim U(0,1),\ \forall\ x \in X,$$

and $Q(x,a)$ is the expected value of bandit rewards with respect to η_a.

Proof. The proof is a slight modification of the proof of Theorem 1 in [4]. For $a \in A(x)$, define $\Delta_a := V(x) - Q(x,a)$ and $\tilde{Q}_m(x,a) = \frac{1}{m}\sum_{j=1}^{m}(R'(x,a,w_j^a) + \gamma U(f(x,a,w_j^a)))$. Let $c_{r,s} = U_{\max}\sqrt{(2\ln r)/s}$. Let $M_t = a$ be the event that machine a is played at time t. For any machine corresponding to an action a, we find an upper bound on $T_a^x(n)$ for any sequence of plays. For an arbitrary positive integer ℓ, we have

$$\begin{aligned}
T_a^x(n) &= 1 + \sum_{t=|A(x)|+1}^{n} I\{M_t = a\} \\
&\le \ell + \sum_{t=|A(x)|+1}^{n} I\{M_t = a,\ T_a^x(t-1) \ge \ell\} \\
&\le \ell + \sum_{t=|A(x)|+1}^{n} I\big\{\tilde{Q}_{T^x_{a^*}(t-1)}(x,a^*) + c_{t-1,T^x_{a^*}(t-1)} \le \\
&\qquad \tilde{Q}_{T_a^x(t-1)}(x,a) + c_{t-1,T_a^x(t-1)},\ T_a^x(t-1) \ge \ell\big\} \\
&\le \ell + \sum_{t=|A(x)|+1}^{n} I\big\{ \min_{0<s<t}\big(\tilde{Q}_s(x,a^*) + c_{t-1,s}\big) \le \\
&\qquad \max_{\ell \le s_a < t}\big(\tilde{Q}_{s_a}(x,a) + c_{t-1,s_a}\big)\big\} \\
&\le \ell + \sum_{t=1}^{n}\sum_{s=1}^{t-1}\sum_{s_a=\ell}^{t-1} I\big\{\tilde{Q}_s(x,a^*) + c_{t,s} \le \tilde{Q}_{s_a}(x,a) + c_{t,s_a}\big\}. \qquad (2.10)
\end{aligned}$$

Next observe that if $I\big\{\tilde{Q}_s(x,a^*) + c_{t,s} \le \tilde{Q}_{s_a}(x,a) + c_{t,s_a}\big\} = 1$, then at least one of the following events must be true:

$$\tilde{Q}_s(x,a^*) \le V(x) - c_{t,s}, \qquad (2.11)$$

$$\tilde{Q}_{s_a}(x,a) \ge Q(x,a) + c_{t,s_a}, \qquad (2.12)$$

$$V(x) < Q\ (x,a) + 2c_{t,s_a}. \qquad (2.13)$$

By using the Hoeffding's inequality [72] we can bound the probability of events (2.11) and (2.12):

$$P\Big(\tilde{Q}_s(x,a^*) \le V(x) - c_{t,s}\Big) \le e^{-4lnt} = t^{-4},$$

$$P\Big(\tilde{Q}_{s_a}(x,a) \ge Q(x,a) + c_{t,s_a}\Big) \le e^{-4lnt} = t^{-4}.$$

Note that for $s_a \ge \lceil (8U_{\max}^2 \ln t)/\Delta_a^2 \rceil$, (2.13) cannot be true for any t, since

$$\begin{aligned}
V(x) - Q(x,a) - 2c_{t,s_a} &= V(x) - Q(x,a) - 2U_{\max}\sqrt{2\ln t / s_a} \\
&\ge V(x) - Q(x,a) - \Delta_a = 0.
\end{aligned}$$

Therefore, it follows that by taking $\ell = \left\lceil \frac{8U_{\max}^2 \ln n}{\Delta_a^2} \right\rceil$ in (2.10), we have

$$\begin{aligned} E[T_a^x(n)] &\le \ell + \sum_{t=1}^{n} \sum_{s=1}^{t-1} \sum_{s_a=\ell}^{t-1} \Big[P\Big(\tilde{Q}_s(x, a^*) \le V(x) - c_{t,s} \Big) \\ &\quad + P\Big(\tilde{Q}_{s_a}(x, a) \ge Q(x, a) + c_{t,s_a} \Big) \Big] \\ &\le \left\lceil \frac{8U_{\max}^2 \ln n}{\Delta_a^2} \right\rceil + \sum_{t=1}^{\infty} \sum_{s=1}^{t-1} \sum_{s_a=1}^{t-1} 2t^{-4} \\ &\le \frac{8U_{\max}^2 \ln n}{\Delta_a^2} + 1 + 2\sum_{t=1}^{\infty} t^{-2} \\ &\le \frac{8U_{\max}^2 \ln n}{(V(x) - Q(x, a))^2} + 1 + \frac{\pi^2}{3}. \end{aligned} \tag{2.14}$$

By the definition of $\rho(n)$, we have

$$\begin{aligned} \rho(n) &= V(x) \sum_{a=1}^{|A(x)|} T_a^x(n) - \sum_{a=1}^{|A(x)|} Q(x, a) E[T_a^x(n)] \\ &= \sum_{a=1}^{|A(x)|} E[T_a^x(n)] \left(V(x) - Q(x, a) \right) \\ &\le \sum_{a: Q(x,a) < V(x)} E[T_a^x(n)] \left(V(x) - Q(x, a) \right), \end{aligned}$$

and the proof is completed by applying the bound given by (2.14). □

Now let $\phi(x)$ be the set of nonoptimal actions at state x, given by $\phi(x) = \{a | Q(x, a) < V(x),\ a \in A(x)\}$, and whenever $\phi(x) \neq \emptyset$, we define the difference between the largest and the second largest expected bandit rewards by

$$\alpha(x) = \min_{a \in \phi(x)} \left(V(x) - Q(x, a) \right). \tag{2.15}$$

Throughout the analysis, we assume that $\alpha(x)$ satisfies the following condition.

Assumption 1. *There exists a constant $C > 0$ such that*

$$\inf_{x \in X} \alpha(x) \ge C.$$

Note that Assumption 1 is trivially satisfied if the state space X is finite.

The convergence of the OSA algorithm is summarized in the following lemma.

Lemma 2.2. *Given a stochastic value function U defined over X with finite $U_{\max}$, suppose we run OSA with the input n for any $x \in X$ with $A(x) > 1$. If Assumption 1 is satisfied, then*

$$E[\tilde{V}^n(x)] \to V(x) \text{ as } n \to \infty.$$

Proof. Observe that $\max_a(V(x) - Q(x,a)) \leq U_{\max}$ and $0 < \alpha(x) \leq U_{\max}$. Define

$$\tilde{V}(x) = \sum_{a=1}^{|A(x)|} \frac{T_a^x(n)}{n} Q(x,a).$$

Applying Theorem 2.1, we have

$$\begin{aligned} 0 \leq V(x) - E[\tilde{V}(x)] &= \frac{\rho(n)}{n} \\ &\leq \frac{8U_{\max}^2(|A(x)|-1)\ln n}{n\alpha(x)} + \left(1 + \frac{\pi^2}{3}\right)\frac{(|A(x)|-1)U_{\max}}{n} \\ &\leq \frac{C_1 \ln n}{n} + \frac{C_2}{n}, \end{aligned} \tag{2.16}$$

for some constants C_1 and C_2, where the last inequality follows from Assumption 1 and the fact that $\rho(n) = 0$ if $\phi(x) = \emptyset$. From the definition of $\tilde{V}^n(x)$ given by Equation (2.9), it follows that

$$\begin{aligned} V(x) - E[\tilde{V}^n(x)] &= V(x) - E[\tilde{V}(x) - \tilde{V}(x) + \tilde{V}^n(x)] \\ &= V(x) - E[\tilde{V}(x)] \\ &\quad + E\left[\sum_{a \in A(x)} \frac{T_a^x(n)}{n}\left(Q(x,a) - \tilde{Q}(x,a)\right)\right]. \end{aligned} \tag{2.17}$$

Letting $n \to \infty$, the first term $V(x) - E[\tilde{V}(x)]$ is bounded by zero from below with convergence rate of $O(\frac{\ln n}{n})$ by (2.16). We show now that the second expectation term is zero.

Note that for every finite n, $T_a^x(n) \leq n < \infty$ and the event $\{T_a^x(n) = k\}$ is independent of $\{w_{k+1}^a, \ldots\}$. Let $\mu_a(x) = E[R'(x,a,w_j^a) + \gamma U(f(x,a,w_j^a))]$. Then,

$$\begin{aligned} &E\left[\sum_{a \in A(x)} \frac{T_a^x(n)}{n}\left(Q(x,a) - \tilde{Q}(x,a)\right)\right] \\ &= E\Bigg[\sum_{a \in A(x)} \frac{T_a^x(n)}{n}\Bigg(\frac{1}{T_a^x(n)} \sum_{j=1}^{T_a^x(n)} \mu_a(x) \\ &\qquad - \frac{1}{T_a^x(n)} \sum_{j=1}^{T_a^x(n)} \left[R'(x,a,w_j^a) + \gamma U(f(x,a,w_j^a))\right]\Bigg)\Bigg] \end{aligned}$$

$$= \frac{1}{n}\Bigg(\sum_{a\in A(x)} E[T_a^x(n)]\mu_a(x) - \sum_{a\in A(x)} E\Bigg[\sum_{j=1}^{T_a^x(n)} \big[R'(x,a,w_j^a) + \gamma U(f(x,a,w_j^a))\big]\Bigg]\Bigg) \;=\; 0,$$

by applying a result analogous to Wald's equation.

Since

$$V(x) - E[\tilde{V}^n(x)] = V(x) - E[\tilde{V}(x)],$$

the convergence follows directly from Equation (2.17).

Therefore, because x was chosen arbitrarily, we have that for all $x \in X$,

$$E[\tilde{V}^n(x)] \to V(x) \text{ as } n \to \infty,$$

which concludes the proof of Lemma 2.2. □

We now state the main convergence theorem for the UCB sampling algorithm, whose proof is based upon an inductive application of Lemma 2.2.

Theorem 2.3. *Assume that $|A(x)| > 1$ for all $x \in X$. Suppose the UCB sampling algorithm is run with the input N_i for stage $i = 0, ..., H-1$, and an arbitrary initial state $x \in X$. If Assumption 1 is satisfied, then*

(i) $\lim_{N_0\to\infty} \lim_{N_1\to\infty} \cdots \lim_{N_{H-1}\to\infty} E[\hat{V}_0^{N_0}(x)] = V_0^*(x)$.

(ii) Moreover, the bias induced by the algorithm is bounded by a quantity that converges to zero at rate $O\big(\sum_{i=0}^{H-1} \frac{\ln N_i}{N_i}\big)$, i.e.,

$$V_0^*(x) - E[\hat{V}_0^{N_0}(x)] \le O\left(\sum_{i=0}^{H-1} \frac{\ln N_i}{N_i}\right),\ x \in X.$$

Proof. Part (i). From the definition of $\hat{V}_{H-1}^{N_{H-1}}$,

$$\hat{V}_{H-1}^{N_{H-1}}(x) = \sum_{a\in A(x)} \frac{1}{N_{H-1}} \sum_{j=1}^{N_a^{H-1}(x)} \Big(R'(x,a,w_j^a) + \gamma \hat{V}_H^{N_H}(f(x,a,w_j^a))\Big)$$
$$\le \sum_{a\in A(x)} \frac{N_a^{H-1}(x)}{N_{H-1}} (R_{\max} + \gamma \cdot 0) = R_{\max},\ x \in X.$$

Similarly for $\hat{V}_{H-2}^{N_{H-2}}$, we have that

$$\hat{V}_{H-2}^{N_{H-2}}(x) = \sum_{a\in A(x)} \frac{1}{N_{H-2}} \sum_{j=1}^{N_a^{H-2}(x)} \Big(R'(x,a,w_j^a) + \gamma \hat{V}_{H-1}^{N_{H-1}}(f(x,a,w_j^a))\Big)$$
$$\le \sum_{a\in A(x)} \frac{N_a^{H-2}(x)}{N_{H-2}} (R_{\max} + \gamma R_{\max}) = R_{\max}(1+\gamma),\ x \in X.$$

Continuing this backwards, we have for all $x \in X$ and $i = 0, ..., H-1$,

$$\hat{V}_i^{N_i}(x) \leq R_{\max} \sum_{j=0}^{H-i-1} \gamma^j \leq R_{\max}(H-i).$$

Therefore, from Lemma 2.2 with $U_{\max} = R_{\max}(H-i)$, we have for $i = 0, ..., H-1$, and for arbitrary $x \in X$,

$$E[\hat{V}_i^{N_i}(x)] \overset{N_i \to \infty}{\longrightarrow} \max_{a \in A(x)} \left(E[R'(x,a,w) + \gamma E[\hat{V}_{i+1}^{N_{i+1}}(f(x,a,w))]] \right).$$

But for arbitrary $x \in X$, because $\hat{V}_H^{N_H}(x) = V_H^*(x) = 0,\ x \in X$,

$$E[\hat{V}_{H-1}^{N_{H-1}}(x)] \overset{N_{H-1} \to \infty}{\longrightarrow} V_{H-1}^*(x),$$

which in turn leads to $E[\hat{V}_{H-2}^{N_{H-2}}(x)] \to V_{H-2}^*(x)$ as $N_{H-2} \to \infty$ for arbitrary $x \in X$, and by an inductive argument, we have that

$$\lim_{N_0 \to \infty} \lim_{N_1 \to \infty} \cdots \lim_{N_{H-1} \to \infty} E[\hat{V}_0^{N_0}(x)] = V_0^*(x) \text{ for all } x \in X,$$

which concludes the proof of the first part of Theorem 2.3.

Part (ii). We now argue that the bias of the optimal function estimator in the UCB sampling algorithm is bounded by a quantity that converges to zero at rate $O\left(\sum_{i=0}^{H-1} \frac{\ln N_i}{N_i}\right)$. Define $\Psi_i \in B(X)$ such that $\Psi_i(x) = E[\hat{V}_i^{N_i}(x)]$ for all $x \in X$ and $i = 0, ..., H-1$ and $\Psi_H(x) = V_H^*(x) = 0,\ x \in X$. In the proof of Lemma 2.2 (see Equation (2.17)), we showed that for $i = 0, ..., H-1$,

$$T(\Psi_{i+1})(x) - \Psi_i(x) \leq O\left(\frac{\ln N_i}{N_i}\right),\ x \in X,$$

where T is defined in Equation (1.18). Therefore, we have

$$T(\Psi_1)(x) - \Psi_0(x) \leq O\left(\frac{\ln N_0}{N_0}\right),\ x \in X. \tag{2.18}$$

and

$$\Psi_1(x) \geq T(\Psi_2)(x) - O\left(\frac{\ln N_1}{N_1}\right),\ x \in X. \tag{2.19}$$

Applying the T-operator to both sides of (2.19), and using the monotonicity property of T, we have

$$T(\Psi_1)(x) \geq T^2(\Psi_2)(x) - O\left(\frac{\ln N_1}{N_1}\right),\ x \in X. \tag{2.20}$$

Therefore, combining (2.18) and (2.20) yields

$$T^2(\Psi_2)(x) - \Psi_0(x) \leq O\left(\frac{\ln N_0}{N_0} + \frac{\ln N_1}{N_1}\right),\ x \in X.$$

Repeating this argument yields

$$T^H(\Psi_H)(x) - \Psi_0(x) \leq O\left(\sum_{i=0}^{H-1} \frac{\ln N_i}{N_i}\right),\ x \in X. \tag{2.21}$$

Observe that $T^H(\Psi_H)(x) = V_0^*(x),\ x \in X$. Rewriting (2.21), we finally have

$$V_0^*(x) - E[\hat{V}_0^{N_0}(x)] \leq O\left(\sum_{i=0}^{H-1} \frac{\ln N_i}{N_i}\right),\ x \in X,$$

and we know that $V_0^*(x) - E[\hat{V}_0^{N_0}(x)] \geq 0,\ x \in X$. Therefore, it implies that the worst possible bias is bounded by the quantity that converges to zero at rate $O\left(\sum_{i=0}^{H-1} \frac{\ln N_i}{N_i}\right)$. □

2.1.5 Numerical Example

To illustrate the algorithm, we consider some computational experiments on a finite-horizon inventory control problem with lost sales. The objective is to find the (nonstationary) policy to minimize expected costs, which comprise holding, order, and penalty costs. Demand is a discrete random variable. Given an inventory level, orders are placed and received, demand is realized, and the new inventory level for the period is calculated, on which costs are charged.

Let D_t denote the demand in period t, x_t the inventory level at the end of period t (which is the inventory at the beginning of period $t+1$), a_t the order amount in period t, p the per-period per-unit demand lost penalty cost, h the per-period per-unit inventory holding cost, K the fixed (set-up) cost per order, and M the maximum inventory level (storage capacity), i.e., $x_t \in \{0, 1, \ldots, M\}$. Then the state transition follows the dynamics:

$$x_{t+1} = (x_t + a_t - D_t)^+ .$$

The objective function is the expectation of the total cost given by

$$\sum_{t=0}^{H-1} \left[K \cdot I\{a_t > 0\} + h x_{t+1}^+ + p x_{t+1}^-\right],$$

where x_0 is the starting inventory level, H is the number of periods (time horizon). Note that we are ignoring per-unit order costs for simplicity.

We consider two versions: (i) fixed order amount q; (ii) any (integral) order amount (up to capacity). In both cases, if the order amount would bring the inventory level above the inventory capacity M, then that order cannot be placed, i.e., that order amount action is not feasible in that state. In case (i),

UCB Sampling Algorithm for Minimization Problems

Input: stage $i \neq H$, state $x \in X$, $N_i > \max_{x \in X} |A(x)|$.
(For $i = H$, $\hat{V}_H^{N_H}(x) = V_H^{N_H}(x) = 0$.)

Initialization: Simulate $w_1^a \sim U(0,1)$ for each $a \in A(x)$;
set $N_a^i(x) = 1 \ \forall a \in A(x)$, $\bar{n} = |A(x)|$, and

$$\hat{Q}_i^{N_i}(x,a) = R'(x,a,w_1^a) + \gamma \hat{V}_{i+1}^{N_{i+1}}(f(x,a,w_1^a)) \ \ \forall a \in A(x).$$

Loop until $\bar{n} = N_i$:

- Sample $w^{\hat{a}}_{N^i_{\hat{a}}(x)+1} \sim U(0,1)$ for current estimate of optimal action a^*:

$$\hat{a} \in \arg\min_{a \in A(x)} \left(\hat{Q}_i^{N_i}(x,a) - (H-i)\sqrt{\frac{2\ln \bar{n}}{N_a^i(x)}} \right),$$

  where

$$\hat{Q}_i^{N_i}(x,a) = \frac{1}{N_a^i(x)} \sum_{j=1}^{N_a^i(x)} \left[R'(x,a,w_j^a) + \gamma \hat{V}_{i+1}^{N_{i+1}}(f(x,a,w_j^a)) \right]. \quad (2.22)$$

- Update $\hat{Q}_i^{N_i}(x,\hat{a})$ estimate via (2.22) using simulated next state $f(x,\hat{a},w^{\hat{a}}_{T^{\hat{a}}_x(\bar{n})+1})$, with $N_{\hat{a}}^i(x) \leftarrow N_{\hat{a}}^i(x) + 1$.
- $\bar{n} \leftarrow \bar{n} + 1$.

Output:

$$\hat{V}_i^{N_i}(x) = \sum_{a \in A(x)} \frac{N_a^i(x)}{N_i} \hat{Q}_i^{N_i}(x,a). \quad (2.23)$$

Fig. 2.5. Modified UCB algorithm for minimization problems

there are just two actions (order or no order), whereas in case (ii), the number of actions depends on the capacity limit.

The examples presented here were chosen to be simple enough to allow the optimal solution to be determined by standard techniques once the distribution is given, so that the performance of the algorithms could be evaluated. However, the algorithms themselves use no knowledge of the underlying probability distributions driving the randomness in the systems, specifically in this case the demand distribution. Furthermore, there is no structural knowledge on the form of the optimal policy.

In actual implementation, a slight modification is required for this example, because it is a minimization problem, whereas the UCB sampling algorithm was written for a maximization problem. Conceptually, the most straightforward way would be to just take the reward as the negative of the

cost function. However, we instead leave the problem as a minimization, in which case we need to replace the "max" operator with the "min" operator and the addition with subtraction in (2.3):

$$\hat{a} \in \arg\min_{a \in A(x)} \left(\hat{Q}_i^{N_i}(x,a) - (H-i)\sqrt{\frac{2\ln \bar{n}}{N_a^i(x)}} \right),$$

where $R_{\max}$ has been replaced by 1, because empirical results indicated that this "unscaled" version exhibited better performance for this particular inventory control problem. The explicit modified UCB algorithm for minimization problems is given in Figure 2.5.

The alternative estimators would then be obtained by replacing the final estimator given by Equation (2.23) in Figure 2.5 by the following, corresponding to Equations (2.6) and (2.7), respectively:

$$\hat{V}_i^{N_i}(x) = \min_{a \in A(x)} \hat{Q}_i^{N_i}(x,a), \tag{2.24}$$

$$\hat{V}_i^{N_i}(x) = \min \left\{ \hat{Q}_i^{N_i}(x,\hat{a}), \sum_{a \in A(x)} \frac{N_a^i(x)}{N_i} \hat{Q}_i^{N_i}(x,a) \right\}, \tag{2.25}$$

where the operator in defining $\hat{a} \in \arg\max_a \{N_a^i(x)\}$ remains a maximization operation.

With $K = 0$ (no fixed order cost), the optimal order policy is easily solvable without dynamic programming, because the periods are decoupled, and the problem reduces to solving a single-period inventory optimization problem. In case (i), the optimal policy follows a threshold rule, in which an order is placed if the inventory is below a certain level; otherwise, no order is placed. The threshold (order point) is given by

$$s = \min_{x \geq 0} \{x : hE[(x+q-D)^+] + pE[(D-q-x)^+] \geq hE[(x-D)^+] + pE[(D-x)^+]\},$$

i.e., one orders in period t if $x_t < s$ (assuming that $x_t + q \leq M$; also, if the set is empty, then take $s = \infty$, i.e., an order will always be placed). In case (ii), the problem becomes a newsboy problem, with a base-stock (order up to) solution given by

$$S = F^{-1}(p/(p+h)),$$

i.e., one orders $(S - x_t)^+$ in period t (with the implicit assumption $S \leq M$).

For the $K > 0$ case (i), the optimal policy is again a threshold (order point) policy, but the order point is nonstationary, whereas in case (ii), the optimal policy is of the (s, S) type, again nonstationary. To obtain the true solutions, standard backwards induction was employed, using knowledge of the underlying demand distribution.

For the numerical experiments, we used the following parameter settings: horizon $H = 3$; capacity $M = 20$; initial inventory $x_0 = 5$; demand $D_t \sim$

$DU(0, 9)$ (discrete uniform); holding cost $h = 1$; penalty cost $p = 1$ and $p = 10$; fixed order cost $K = 0$ and $K = 5$; fixed order amount for case (i): $q = 10$. Note that since the order quantity is greater than the maximum demand for our values of the parameters, i.e., $q > D_t$ always, placing an order guarantees no lost sales.

Tables 2.1 and 2.2 give the performances of these estimators for each of the respective cases (i) and (ii), including the optimal value and policy parameters. Figures 2.6, 2.7, 2.8, and 2.9 show the convergence of the estimates as a function of the number of samples at each stage for each of the respective cases (i) and (ii) considered. In each table and figure, estimator 1 stands for the original estimator using Equation (2.23), and estimators 2 and 3 refer to the estimators using Equations (2.24) and (2.25) with $a^* \in \arg\max_a \{N_a^i(x)\}$ in place of Equation (2.23), respectively. The results indicate convergence of all three estimators, with the two alternative estimators providing superior empirical performance over the original estimator. We conjecture that this is due to the fact that the original estimator's use of a weighted average is too conservative, thus leading to unnecessarily slow convergence. We suspect this would be the case for the non-adaptive sampling version using a weighted average estimator, too.

Choosing an appropriate sample size is critical in practical applications. The empirical performance of the two alternative estimators indicates that a heuristic stopping rule for choosing the number of samples at each stage could be based on these two estimates, which showed rapid convergence in the numerical examples. This convergence implies that in Equation (2.7), the first term in the "max" operator dominates the second term (i.e., the original estimator), and the actions that have been sampled the most almost "always" yield the largest Q-function values; in other words, at this point, estimators 2 and 3 are "almost" the same, so if they are biased in opposite directions, they must have reached a sample size at which they are "nearly" unbiased. Once this is the case, it may be preferable to perform more independent replications at a particular action than to sample more actions (larger N).

Table 2.1. Value function estimate for the inventory control example case (i) as a function of the number of samples at each state: $H = 3, M = 20, x_0 = 5$, $D_t \sim DU(0, 9), q = 10, h = 1$, where each entry represents the mean based on 30 independent replications (standard error in parentheses)

(K, p)	optimal	N	estimator 1	estimator 2	estimator 3
	10.440	4	15.03 (0.29)	9.13 (0.21)	9.56 (0.32)
$K = 0$	$s = 0$	8	12.82 (0.16)	10.21 (0.10)	10.30 (0.10)
$p = 1$		16	11.75 (0.09)	10.33 (0.08)	10.38 (0.08)
		32	11.23 (0.06)	10.45 (0.06)	10.49 (0.06)
	24.745	4	30.45 (0.87)	19.98 (0.79)	20.48 (0.82)
$K = 0$	$s = 6$	8	28.84 (0.49)	23.09 (0.55)	23.68 (0.52)
$p = 10$		16	26.69 (0.38)	23.88 (0.44)	23.94 (0.45)
		32	26.12 (0.14)	24.73 (0.19)	24.74 (0.18)
	10.490	4	18.45 (0.29)	10.23 (0.21)	10.41 (0.22)
$K = 5$	$s_1 = 0$	8	14.45 (0.15)	10.59 (0.10)	10.62 (0.10)
$p = 1$	$s_2 = 0$	16	12.48 (0.10)	10.51 (0.10)	10.52 (0.10)
	$s_3 = 0$	32	11.47 (0.07)	10.46 (0.06)	10.46 (0.06)
	31.635	4	37.52 (0.98)	26.42 (0.88)	26.92 (0.89)
$K = 5$	$s_1 = 6$	8	36.17 (0.43)	30.13 (0.49)	30.41 (0.51)
$p = 10$	$s_2 = 6$	16	33.81 (0.40)	30.76 (0.43)	30.80 (0.43)
	$s_3 = 5$	32	33.11 (0.16)	31.62 (0.22)	31.64 (0.22)

Table 2.2. Value function estimate for the inventory control example case (ii) as a function of the number of samples at each state: $H = 3, M = 20, x_0 = 5$, $D_t \sim DU(0, 9), h = 1$, where each entry represents the mean based on 30 independent replications (standard error in parentheses)

(K, p)	optimal	N	estimator 1	estimator 2	estimator 3
	7.500	21	24.06 (0.16)	3.12 (0.17)	9.79 (0.21)
$K = 0$	$S = 4$	25	22.05 (0.12)	5.06 (0.12)	6.28 (0.19)
$p = 1$		30	20.36 (0.11)	5.91 (0.09)	6.47 (0.09)
		35	18.82 (0.11)	6.26 (0.10)	6.62 (0.11)
	13.500	21	29.17 (0.21)	6.04 (0.30)	13.69 (0.46)
$K = 0$	$S = 9$	25	28.08 (0.21)	9.28 (0.23)	12.06 (0.29)
$p = 10$		30	27.30 (0.19)	11.40 (0.20)	13.28 (0.23)
		35	26.06 (0.16)	12.23 (0.18)	13.07 (0.16)
	10.490	21	33.05 (0.12)	8.73 (0.21)	18.62 (0.44)
$K = 5$	$s_1 = 0, S_1 = 0$	25	29.99 (0.10)	10.96 (0.11)	11.79 (0.16)
$p = 1$	$s_2 = 0, S_2 = 0$	30	27.45 (0.10)	11.22 (0.05)	11.52 (0.07)
	$s_3 = 0, S_3 = 0$	35	25.33 (0.09)	10.96 (0.06)	11.12 (0.07)
	25.785	21	39.97 (0.22)	17.78 (0.49)	26.76 (0.52)
$K = 5$	$s_1 = 6, S_1 = 9$	25	39.01 (0.19)	22.68 (0.26)	25.09 (0.33)
$p = 10$	$s_2 = 6, S_2 = 9$	30	38.03 (0.16)	24.35 (0.17)	25.45 (0.27)
	$s_3 = 6, S_3 = 9$	35	36.89 (0.12)	24.71 (0.23)	25.51 (0.28)

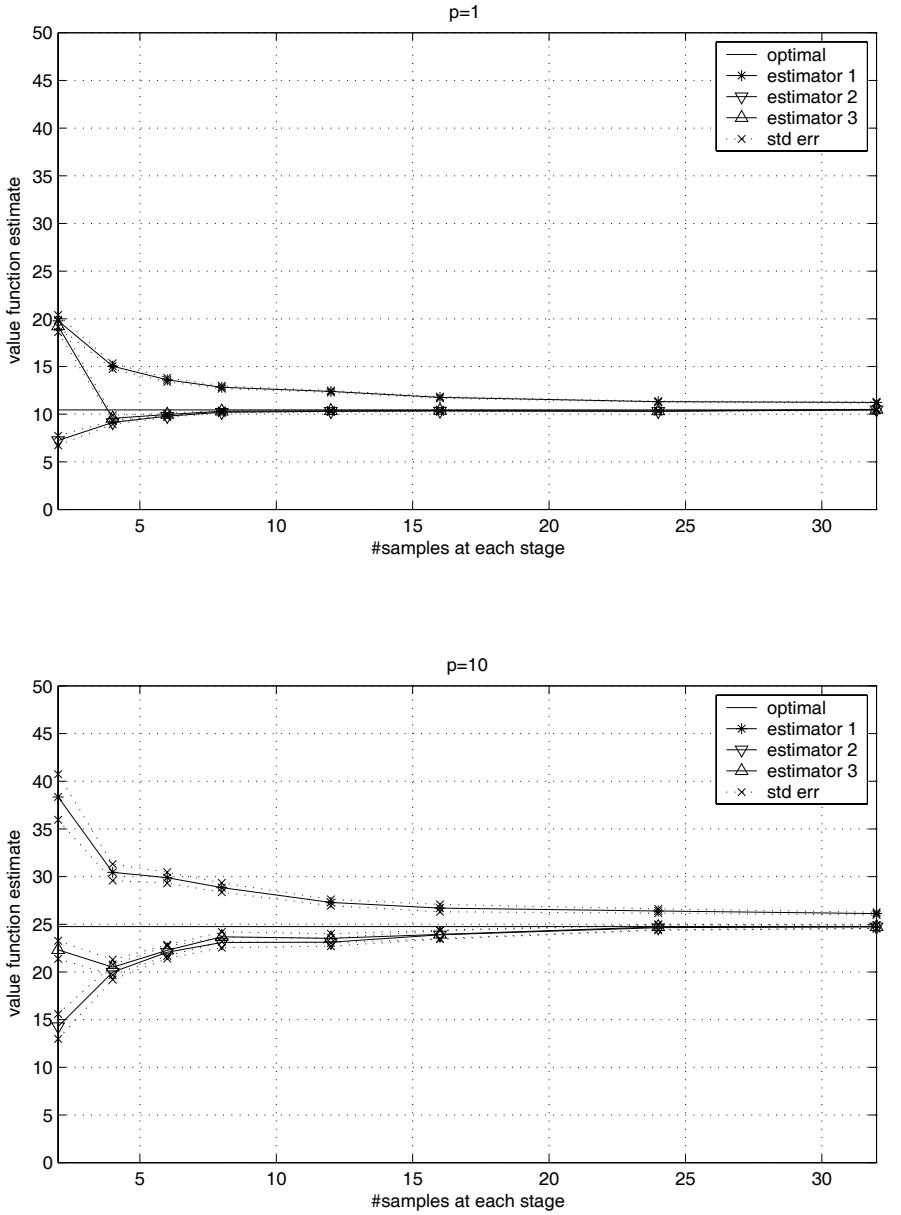

Fig. 2.6. Convergence of value function estimate for the inventory control example case (i) q=10 as a function of the number of samples at each state:
$H = 3, M = 20, x_0 = 5, D_t \sim DU(0, 9), h = 1, K = 0$

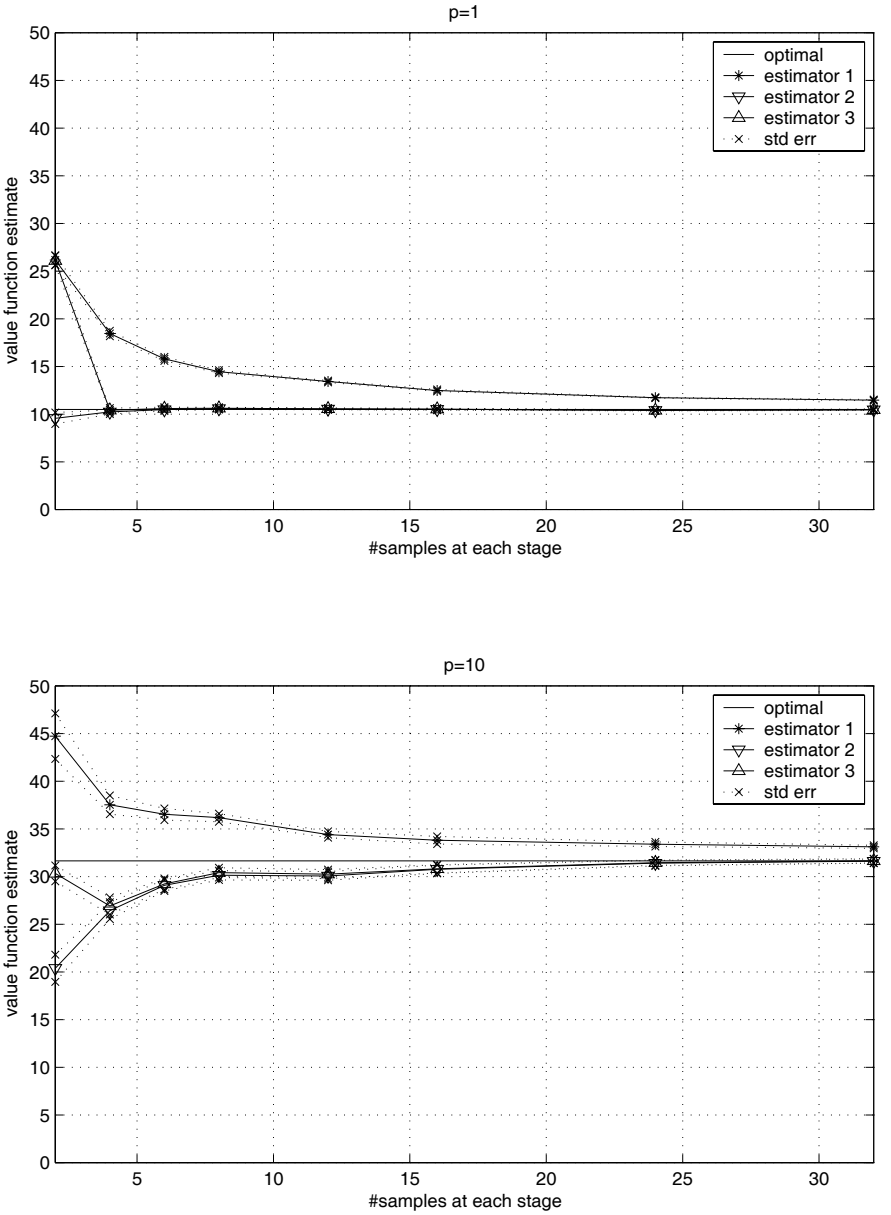

Fig. 2.7. Convergence of value function estimate for the inventory control example case (i) q=10 as a function of the number of samples at each state: $H = 3, M = 20, x_0 = 5, D_t \sim DU(0, 9), h = 1, K = 5$

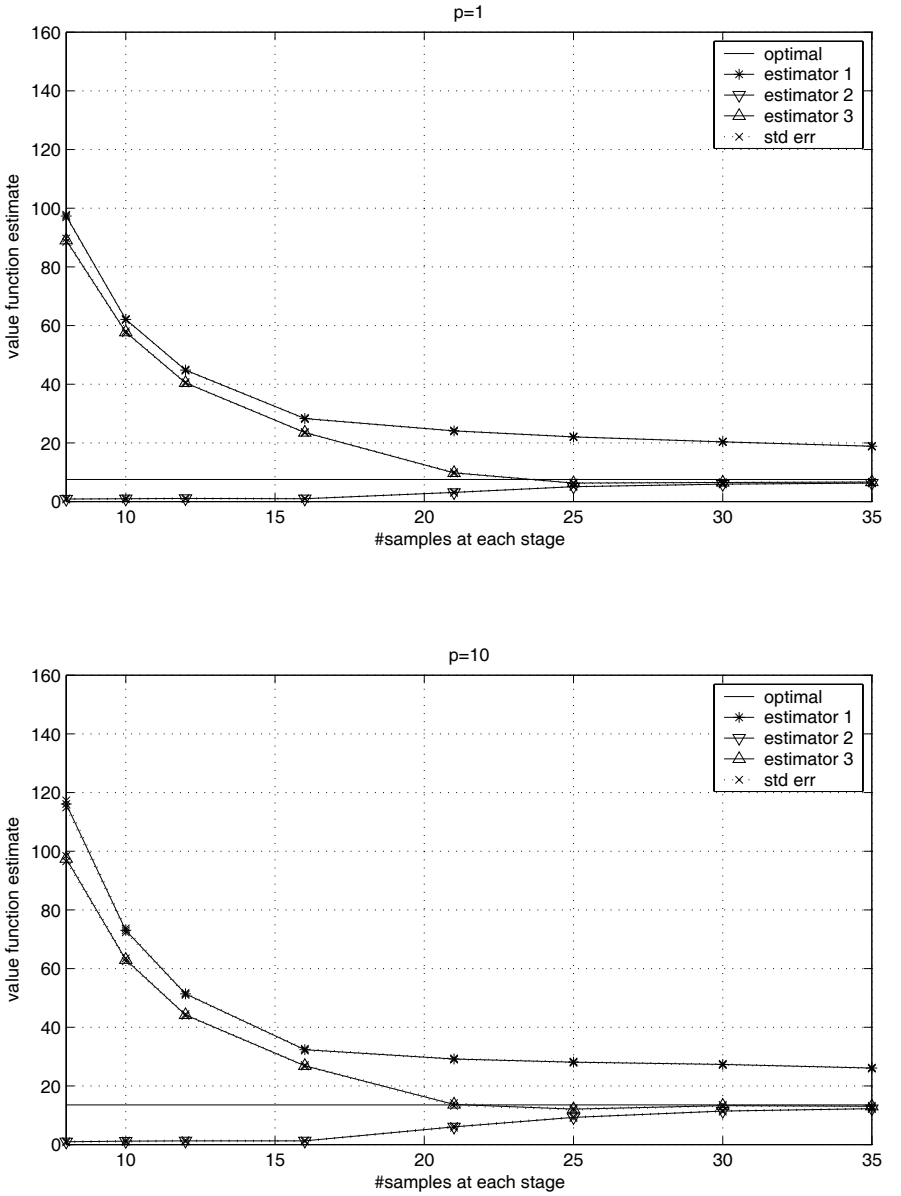

Fig. 2.8. Convergence of value function estimate for the inventory control example case (ii) as a function of the number of samples at each state:
$H = 3, M = 20, x_0 = 5, D_t \sim DU(0, 9), h = 1, K = 0$

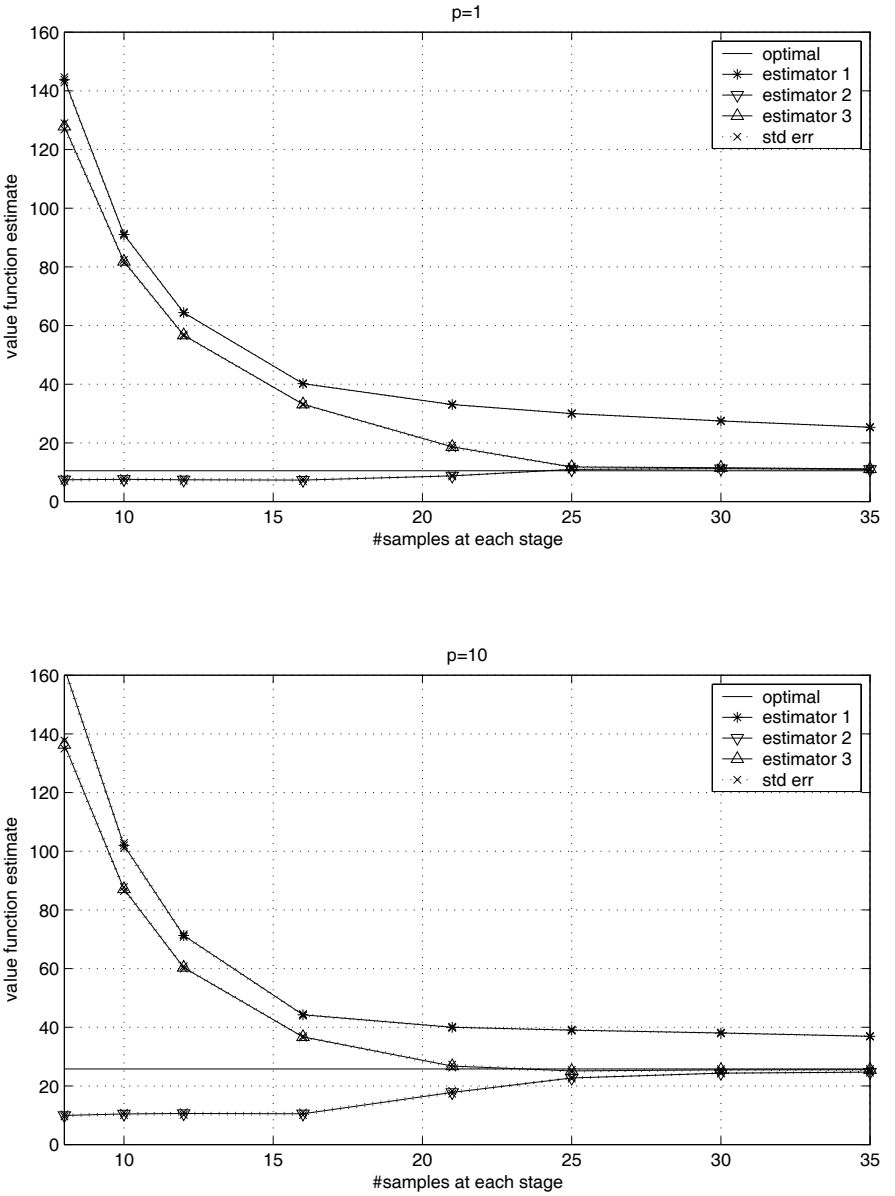

Fig. 2.9. Convergence of value function estimate for the inventory control example case (ii) as a function of the number of samples at each state:
$H = 3, M = 20, x_0 = 5, D_t \sim DU(0, 9), h = 1, K = 5$

2.2 Pursuit Learning Automata Sampling

The second algorithm in the chapter is the pursuit learning automata (PLA) sampling algorithm. We analyze the finite-time behavior of the PLA sampling algorithm, providing a bound on the probability that a given initial state takes the optimal action, and a bound on the probability that the difference between the optimal value and the estimate of it exceeds a given error. Similar to the UCB algorithm, the PLA sampling algorithm constructs a sampled tree in a recursive manner to estimate the optimal value at an initial state and incorporates an adaptive sampling mechanism for selecting which action to simulate at each branch in the tree. In the PLA algorithm, the action is determined by sampling from a *probability distribution*, which is iteratively updated based on a probability estimate for the optimal action. We also discuss how to apply the PLA sampling algorithm in the direct context of partially observable MDPs (POMDPs).

The PLA sampling algorithm extends in a recursive manner (for MDPs) the pursuit algorithm from learning automata that is designed to solve (non-sequential) stochastic optimization problems. A learning automaton is associated with a finite set of actions (candidate solutions) and updates a probability distribution over the set by iterative interaction with an environment and takes (samples) an action according to the newly updated distribution. The environment provides a certain reaction (reward) to the action taken by the automaton, where the reaction is random and the distribution is unknown to the automaton. The automaton's aim is to learn to choose the action that yields the highest average reward. In the pursuit algorithm, the automaton *pursues* the current best action, which is estimated using sample average rewards, by increasing the probability of selecting that action while decreasing the probability of selecting all other actions.

Since learning automata are well-known adaptive decision-making devices operating in unknown random environments, the PLA sampling algorithm's sampling process of taking an action is adaptive at each stage. At each given state in a given stage, a fixed sampling budget is allocated among feasible actions as in the UCB sampling algorithm, and the budget is used with the current probability estimate for the optimal action. A simulated state corresponds to an automaton and updates certain functions (including the probability distribution over the action space) at each iteration of the algorithm.

Based on the finite-time analysis of the pursuit algorithm, we analyze the finite-time behavior of the PLA sampling algorithm, providing:

(i) a bound on the probability that the initial state at stage 0 takes the optimal action, in terms of sampling parameters of the PLA sampling algorithm, and

(ii) a bound on the probability that the difference between the estimate of $V_0^*(x_0)$ and $V_0^*(x_0)$ exceeds a given error.

2.2.1 Algorithm Description

Figure 2.10 presents the PLA sampling algorithm for estimating $V_i^*(x)$ for a given state x. The inputs to the algorithm are similar to the UCB algorithm: a state $x \in X$ and the stage i, plus sampling parameters $N_i > 0$ and $\mu_i \in (0,1)$, where the latter is particular to the PLA sampling algorithm and the former does not require sampling every action at least once, as in the UCB algorithm. The output is the same as in the UCB algorithm: $\hat{V}_i^{N_i}(x)$, an estimate of $V_i^*(x)$, the optimal reward-to-go value for state x and stage i, where $\hat{V}_H^{N_H}(x) = V_H^{N_H}(x) = 0 \ \forall N_H, x \in X$, but it is estimated using the Q-function value at the estimated optimal action (cf. Equation (2.29)), somewhat analogous to the UCB algorithm alternative estimator given by Equation (2.7). As in the UCB sampling algorithm, whenever $\hat{V}_{i'}^{N_{i'}}(y)$ (for future periods $i' > i$ and simulated next states y) is encountered in the **Loop** portion of the algorithm at (2.26), a recursive call is required. The initial call to the algorithm is done with stage $i = 0$, the initial state x_0, N_0, and μ_0, and every sampling is independent of previous samplings.

As in the UCB sampling algorithm, the PLA sampling algorithm builds a sampled tree of depth H, with the root node being the initial state x_0 at stage 0 and a branching factor of N_i at each level i (level 0 corresponds to the root). The root node x_0 initializes the probability distribution over the action space P_{x_0} as the uniform distribution (see the **Initialization** step in the PLA sampling algorithm). At each iteration in the **Loop** step, an action is sampled from the probability distribution $P_{x_0}(k)$ and a random number w_k is generated independently (an action and a random number together corresponding to an edge in the tree). For the sampled action $a(k) \in A(x_0)$, the Q-function estimate is updated using the simulated reward $R'(x_0, a(k), w_k)$ and next state $f(x_0, a(k), w_k)$, and the count variable $N_{a(k)}^0(x_0)$ is incremented, where a recursive call is made to estimate $\hat{V}_1^{N_1}$ at the simulated next state. This is followed by updating the estimate of the optimal action – an action that achieves the current best Q-function value (cf. (2.27)) – and then updating the probability distribution $P_{x_0}(k)$ in the direction of the current estimate of the optimal action $\hat{a}$ (cf. (2.28)) by adding μ_i to its probability mass and subtracting a proportional amount from all other actions. This "pursuit" of the current best action gives the original algorithm its name in its nonrecursive one-stage original version. After N_0 iterations, the algorithm estimates the optimal value $V_0^*(x_0)$ by the Q-function value at the currently estimated optimal action via Equation (2.29), where

$$\hat{Q}_0^{N_0}(x_0, a) = \frac{1}{N_a^0(x_0)} \sum_{j:a(j)=a} \left[R'(x_0, a, w_j) + \hat{V}_1^{N_1}(f(x_0, a, w_j)) \right],$$

$\sum_{a \in A(x_0)} N_a^0(x_0) = N_0$. Note that here for notational simplicity we have not associated the random number streams $\{w_j\}$ with actions, as in the UCB sampling algorithm, where we used $\{w_j^a\}, a \in A(x)$.

Pursuit Learning Automata (PLA) Sampling Algorithm

Input: stage $i < H$, state $x \in X$, $N_i > 0$, $\mu_i \in (0,1)$.
(For $i = H$, $\hat{V}_H^{N_H}(x) = V_H^{N_H}(x) = 0$.)

Initialization: Set $P_x(0)(a) = 1/|A(x)|, N_a^i(x) = 0, M_i(x,a) = 0 \ \forall a \in A(x)$; $k = 0$.

Loop until $k = N_i$:

- Sample $a(k) \sim P_x(k)$, $w_k \sim U(0,1)$.
- Update Q-function estimate for $a = a(k)$ only:

$$\begin{aligned} M_i(x,a(k)) &\leftarrow M_i(x,a(k)) \\ &\quad + R'(x,a(k),w_k) + \hat{V}_{i+1}^{N_{i+1}}(f(x,a(k),w_k)), \quad (2.26) \\ N_{a(k)}^i(x) &\leftarrow N_{a(k)}^i(x) + 1, \\ \hat{Q}_i^{N_i}(x,a(k)) &\leftarrow \frac{M_i^{N_i}(x,a(k))}{N_{a(k)}^i(x)}. \end{aligned}$$

- Update optimal action estimate: (ties broken arbitrarily)

$$\hat{a} \in \arg\max_{a \in A(x)} \hat{Q}_i^{N_i}(x,a). \quad (2.27)$$

- Update probability distribution over action space:

$$P_x(k+1)(a) \leftarrow (1-\mu_i)P_x(k)(a) + \mu_i I\{\hat{a} = a\} \quad \forall a \in A(x). \quad (2.28)$$

- $k \leftarrow k+1$.

Output:

$$\hat{V}_i^{N_i}(x) = \hat{Q}_i^{N_i}(x,\hat{a}). \quad (2.29)$$

Fig. 2.10. Pursuit learning automata (PLA) sampling algorithm description

Analogous to the UCB sampling algorithm, the running-time complexity of the PLA sampling algorithm is $O(N^H)$ with $N = \max_i N_i$, independent of the state space size. (For some performance guarantees, the value N depends on the size of the action space; see the next section.)

2.2.2 Convergence Analysis

All of the estimated optimal value and Q-values in the current section refer to the values from the **Output** step of the algorithm. The following lemma provides a probability bound on the estimate of the Q-value relative to the true Q-value when the estimate of the Q-value is obtained under the assumption that the optimal value for the remaining horizon is known (so that the recursive call is not required).

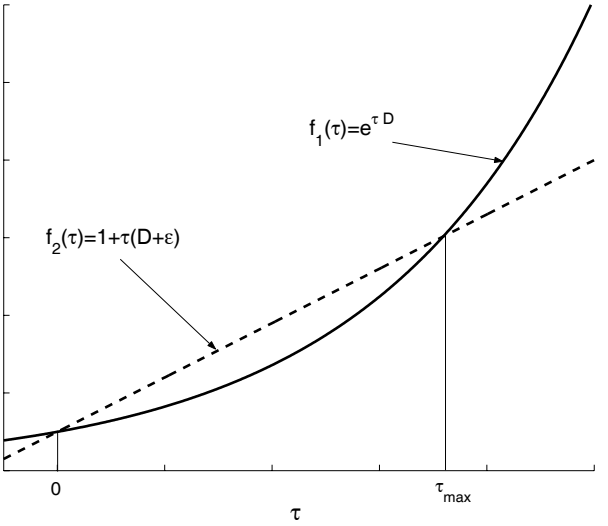

Fig. 2.11. Sketch of functions $f_1(\tau) = e^{\tau D}$ and $f_2(\tau) = 1 + \tau(D + \epsilon)$

Lemma 2.4. *(cf. [119, Lemma 3.1]) Given $\delta \in (0, 1)$ and positive integer K such that $6 \leq K < \infty$, consider running the one-stage nonrecursive PLA sampling algorithm obtained by replacing (2.26) by*

$$M_i^{N_i}(x, a) \leftarrow M_i^{N_i}(x, a) + R'(x, a, w_k) + V_{i+1}^*(f(x, a, w_k)) \tag{2.30}$$

with $N_i > \bar{\lambda}(K, \delta)$ and $0 < \mu_i < \bar{\mu}_i(K, \delta)$, where

$$\bar{\lambda}(K, \delta) = \left\lceil \frac{2K}{\ln l} \ln \left\lceil \frac{Kl}{\ln l} \left(\frac{K}{\delta} \right)^{\frac{1}{K}} \right\rceil \right\rceil, \quad \bar{\mu}_i(K, \delta) = 1 - 2^{-1/\bar{\lambda}(K,\delta)},$$

and $l = \frac{2|A(x)|}{2|A(x)|-1}$. Then for each action $a \in A(x)$, we have

$$P\left(\sum_{j=0}^{N_i} I\{a(j) = a\} \leq K \right) < \delta.$$

Theorem 2.5. *Let $\{X_i,\ i = 1, 2, \ldots\}$ be a sequence of i.i.d. non-negative uniformly bounded random variables, with $0 \leq X_i \leq D$ and $E[X_i] = \mu\ \forall\ i$, and let $M \in \mathcal{Z}^+$ be a positive integer-valued random variable bounded by L. Then for any given $\epsilon > 0$ and $n \in \mathcal{Z}^+$, we have*

$$P\left(\left| \frac{1}{M} \sum_{i=1}^{M} X_i - \mu \right| \geq \epsilon, M \geq n \right) \leq 2e^{-n\left(\frac{D+\epsilon}{D} \ln \frac{D+\epsilon}{D} - \frac{\epsilon}{D} \right)}.$$

Proof. Define $\Lambda_D(\tau) := \frac{e^{D\tau}-1-\tau D}{D^2}$, and let $\tau_{\max}$ be a constant satisfying $\tau_{\max} \neq 0$ and $1 + (D+\epsilon)\tau_{\max} - e^{D\tau_{\max}} = 0$ (see Figure 2.11).

Let $Y_k = \sum_{i=1}^{k}(X_i - \mu)$. It is easy to see that the sequence $\{Y_k\}$ forms a martingale w.r.t. $\{\mathcal{F}_k\}$, where $\mathcal{F}_j$ is the σ-field generated by $\{Y_1, \ldots, Y_j\}$. Therefore, for any $\tau > 0$,

$$\begin{aligned} P\Big(\frac{1}{M}\sum_{i=1}^{M} X_i - \mu \geq \epsilon, M \geq n\Big) &= P(Y_M \geq M\epsilon, M \geq n) \\ &= P(\tau Y_M - \Lambda_D(\tau)\langle Y\rangle_M \geq \tau M\epsilon - \Lambda_D(\tau)\langle Y\rangle_M, M \geq n), \end{aligned}$$

where

$$\langle Y\rangle_n = \sum_{j=1}^{n} E[(\Delta Y_j)^2|\mathcal{F}_{j-1}],\ \Delta Y_j = Y_j - Y_{j-1}.$$

Now for any $\tau \in (0, \tau_{\max})$, and for any $n_1 \geq n_0$, where $n_0,\ n_1 \in \mathcal{Z}^+$,

$$\begin{aligned} \tau(n_1 - n_0)\epsilon &\geq \frac{e^{D\tau}-1-\tau D}{D^2}(n_1 - n_0)D^2 \\ &\geq \Lambda_D(\tau)\Big[\sum_{j=1}^{n_1} E[(\Delta Y_j)^2|\mathcal{F}_{j-1}] - \sum_{j=1}^{n_0} E[(\Delta Y_j)^2|\mathcal{F}_{j-1}]\Big], \end{aligned}$$

which implies that

$$\tau n_1\epsilon - \Lambda_D(\tau)\langle Y\rangle_{n_1} \geq \tau n_0\epsilon - \Lambda_D(\tau)\langle Y\rangle_{n_0} \ \ \forall\ \tau \in (0, \tau_{\max}).$$

Thus for all $\tau \in (0, \tau_{\max})$,

$$\begin{aligned} &P\Big(\frac{1}{M}\sum_{i=1}^{M} X_i - \mu \geq \epsilon, M \geq n\Big) \\ &\quad\leq P(\tau Y_M - \Lambda_D(\tau)\langle Y\rangle_M \geq \tau n\epsilon - \Lambda_D(\tau)\langle Y\rangle_n, M \geq n) \\ &\quad\leq P(\tau Y_M - \Lambda_D(\tau)\langle Y\rangle_M \geq \tau n\epsilon - \Lambda_D(\tau)nD^2, M \geq n) \\ &\quad= P(e^{\tau Y_M - \Lambda_D(\tau)\langle Y\rangle_M} \geq e^{\tau n\epsilon - n\Lambda_D(\tau)D^2}, M \geq n). \end{aligned}$$

It can be shown that (cf. Lemma 1 in [133, p.505]) the sequence $\{Z_t(\tau) = e^{\tau Y_t - \Lambda_D(\tau)\langle Y\rangle_t},\ t \geq 1\}$ with $Z_0(\tau) = 1$ forms a non-negative supermartingale. From the above inequality, it follows that

$$\begin{aligned} &P\Big(\frac{1}{M}\sum_{i=1}^{M} X_i - \mu \geq \epsilon, M \geq n\Big) \\ &\quad\leq P(e^{\tau Y_M - \Lambda_D(\tau)\langle Y\rangle_M} \geq e^{\tau n\epsilon - n\Lambda_D(\tau)D^2}) \\ &\quad\leq P\left(\sup_{0\leq t\leq L} Z_t(\tau) \geq e^{\tau n\epsilon - n\Lambda_D(\tau)D^2}\right) \\ &\quad\leq \frac{E[Z_0(\tau)]}{e^{\tau n\epsilon - n\Lambda_D(\tau)D^2}} \ \text{ by maximal inequality for supermartingales [133]} \\ &\quad= e^{-n(\tau\epsilon - \Lambda_D(\tau)D^2)}. \end{aligned} \tag{2.31}$$

By using a similar argument, we can also show that

$$P\Big(\frac{1}{M}\sum_{i=1}^{M} X_i - \mu \le -\epsilon, M \ge n\Big) \le e^{-n(\tau\epsilon - \Lambda_D(\tau)D^2)}. \tag{2.32}$$

Thus by combining (2.31) and (2.32), we have

$$P\Big(\Big|\frac{1}{M}\sum_{i=1}^{M} X_i - \mu\Big| \ge \epsilon, M \ge n\Big) \le 2e^{-n(\tau\epsilon - \Lambda_D(\tau)D^2)}. \tag{2.33}$$

Finally, we optimize the right-hand side of (2.33) over τ. It is easy to verify that the optimal τ^* is given by $\tau^* = \frac{1}{D}\ln\frac{D+\epsilon}{D} \in (0, \tau_{\max})$ and

$$\tau^*\epsilon - \Lambda_D(\tau^*)D^2 = \frac{D+\epsilon}{D}\ln\frac{D+\epsilon}{D} - \frac{\epsilon}{D} > 0.$$

Hence Theorem 2.5 follows. □

Lemma 2.6. *Consider the nonrecursive PLA sampling algorithm obtained by replacing (2.26) by*

$$M_i^{N_i}(x,a) \leftarrow M_i^{N_i}(x,a) + R'(x,a,w_k) + V_{i+1}^*(f(x,a,w_k)). \tag{2.34}$$

Assume $N_i > \lambda(\epsilon,\delta)$ *and* $0 < \mu_i < \mu_i^*(\epsilon,\delta)$, *where*

$$\lambda(\epsilon,\delta) = \left\lceil \frac{2M_{\epsilon,\delta}}{\ln l} \ln\left[\frac{lM_{\epsilon,\delta}}{\ln l}\left(\frac{2M_{\epsilon,\delta}}{\delta}\right)^{1/M_{\epsilon,\delta}}\right]\right\rceil, \tag{2.35}$$

with

$$M_{\epsilon,\delta} = \max\left\{6, \left\lceil \frac{R_{\max}H\ln(4/\delta)}{(R_{\max}H+\epsilon)\ln((R_{\max}H+\epsilon)/R_{\max}H) - \epsilon}\right\rceil\right\},$$

$l = 2|A(x)|/(2|A(x)|-1)$, *and* $\mu_i^*(\epsilon,\delta) = 1 - 2^{-1/N_i}$. *Consider a fixed* i, $x \in X$, $\epsilon > 0$, *and* $\delta \in (0,1)$. *Then, for all* $a \in A(x)$ *at the* **Output** *step,*

$$P\left(|\hat{Q}_i^{N_i}(x,a) - Q_i^*(x,a)| \ge \epsilon\right) < \delta.$$

Proof. For any action $a \in A(x)$, let $I_j(a)$ be the iteration at which action a is chosen for the jth time, let $\hat{Q}_{i,k}^{N_i}(x,a)$ be the current estimate of $Q_i^*(x,a)$ at the kth iteration, and let $N_a^{i,k}(x)$ be the number of times action a is sampled up to the kth iteration at x, i.e., $N_a^{i,k}(x) = \sum_{j=0}^{k} I\{a(j) = a\}$. By the PLA sampling algorithm, the estimation $\hat{Q}_{i,k}^{N_i}(x,a)$ is given by (cf. (2.34))

$$\hat{Q}_{i,k}^{N_i}(x,a) = \frac{1}{N_a^{i,k}(x)} \sum_{j=1}^{N_a^{i,k}(x)} \big(R'(x,a,w_{I_j(a)}) + V_{i+1}^*(f(x,a,w_{I_j(a)}))\big). \tag{2.36}$$

Since the sequence of random variables $\{w_{I_j(a)},\ j \geq 1\}$ is i.i.d., a straightforward application of Theorem 2.5 yields

$$P\Big(\big|\hat{Q}_{i,k}^{N_i}(x,a) - Q_i^*(x,a)\big| \geq \epsilon,\ N_a^{i,k}(x) \geq K\Big) \leq 2e^{-K\left(\frac{R_{\max}H+\epsilon}{R_{\max}H}\ln\frac{R_{\max}H+\epsilon}{R_{\max}H} - \frac{\epsilon}{R_{\max}H}\right)}. \tag{2.37}$$

Define the events

$$\mathcal{A}_k = \big\{\big|\hat{Q}_{i,k}^{N_i}(x,a) - Q_i^*(x,a)\big| \geq \epsilon\big\} \ \text{ and } \ \mathcal{B}_k = \big\{N_a^{i,k}(x) \geq K\big\}.$$

By the law of total probability,

$$P(\mathcal{A}_k) = P(\mathcal{A}_k \cap \mathcal{B}_k) + P(\mathcal{A}_k|\mathcal{B}_k^c)P(\mathcal{B}_k^c) \leq P(\mathcal{A}_k \cap \mathcal{B}_k) + P(\mathcal{B}_k^c).$$

Taking $K = \left\lceil \frac{R_{\max}H\ln(4/\delta)}{(R_{\max}H+\epsilon)\ln((R_{\max}H+\epsilon)/R_{\max}H)-\epsilon} \right\rceil$, we get from (2.37) that $P(\mathcal{A}_k \cap \mathcal{B}_k) \leq \delta/2$. On the other hand, by Lemma 2.4

$$P(\mathcal{B}_k^c) = P(N_a^{i,k}(x) < K) < \frac{\delta}{2} \ \text{ for } k > \bar{\lambda}(K, \delta/2) \text{ and } 0 < \mu_i < 1 - 2^{-\frac{1}{\bar{\lambda}(K,\frac{\delta}{2})}}.$$

Therefore $P(\mathcal{A}_{N_i}) = P\left(|\hat{Q}_i^{N_i}(x,a) - Q_i^*(x,a)| \geq \epsilon\right) < \delta$ for $N_i > \lambda(\epsilon,\delta)$ and $0 < \mu_i < \mu_i^*(\epsilon,\delta)$, where

$$\begin{aligned}
\lambda(\epsilon,\delta) &= \left\lceil \frac{2M_{\epsilon,\delta}}{\ln l} \ln\left\lceil \frac{lM_{\epsilon,\delta}}{\ln l}\left(\frac{2M_{\epsilon,\delta}}{\delta}\right)^{1/M_{\epsilon,\delta}}\right\rceil\right\rceil, \\
M_{\epsilon,\delta} &= \max\left\{6, \left\lceil \frac{R_{\max}H\ln(4/\delta)}{(R_{\max}H+\epsilon)\ln\big((R_{\max}H+\epsilon)/R_{\max}H\big)-\epsilon}\right\rceil\right\}, \\
\mu_i^*(\epsilon,\delta) &= 1 - 2^{-\frac{1}{N_i}} < 1 - 2^{-\frac{1}{\lambda(\epsilon,\delta)}}.
\end{aligned}$$

Since $a \in A(x)$ is arbitrary, the proof is complete. □

We now make an assumption *for the purpose of the analysis*. The assumption states that at each stage, the optimal action is unique at each state. In other words, the given MDP has a unique optimal policy. We will give a remark on this at the end of this section.

Assumption 2. *For all $x \in X$ and $i = 0, 1, ..., H-1$,*

$$\theta_i(x) := Q_i^*(x,a^*) - \max_{a \neq a^*} Q_i^*(x,a) > 0,$$

where $V_i^*(x) = Q_i^*(x,a^*)$.

Define $\theta := \inf_{x\in X, i=0,\ldots,H-1} \theta_i(x)$. Given $\delta_i \in (0,1), i = 0, ..., H-1$, define

$$\rho := (1-\delta_0)\prod_{i=1}^{H-1}(1-\delta_i)^{\prod_{j=1}^{i} N_j}. \tag{2.38}$$

Lemma 2.7. *Assume that Assumption 2 holds. Select* $N_i > \lambda(\frac{\theta}{2^{i+2}}, \delta_i)$ *(see Equation (2.35)) and* $0 < \mu_i < \mu_i^* = 1 - 2^{-\frac{1}{N_i}}$ *for a given* $\delta_i \in (0,1), i = 0, ..., H-1$. *Then under the PLA sampling algorithm,*

$$P\left(\left|\hat{V}_0^{N_0}(x_0) - V_0^*(x_0)\right| > \frac{\theta}{2}\right) < 1 - \rho,$$

where ρ *is given by Equation (2.38).*

Proof. Let X_s^i be the set of sampled states in X by the algorithm at stage i. Suppose for a moment that for all $x \in X_s^{i+1}$, with some N_{i+1}, μ_{i+1}, and a given $\delta_{i+1} \in (0,1)$,

$$P\left(\left|\hat{V}_{i+1}^{N_{i+1}}(x) - V_{i+1}^*(x)\right| > \frac{\theta}{2^{i+2}}\right) < \delta_{i+1}. \tag{2.39}$$

Consider for $x \in X_s^i$,

$$\tilde{Q}_i^{N_i}(x,a) = \frac{1}{N_a^i(x)} \sum_{j=1}^{N_a^i(x)} \left[R'(x,a,w_j^a) + V_{i+1}^*(f(x,a,w_j^a))\right],$$

where $\{w_j^a\}, j = 1, ..., N_a^i(x)$ refers to the sampled random number sequence for the sample execution of the action a in the algorithm. We have that for any sampled $x \in X_s^i$ at stage i,

$$\begin{aligned}\hat{Q}_i^{N_i}(x,a) - \tilde{Q}_i^{N_i}(x,a) \\ = \frac{1}{N_a^i(x)} \sum_{j=1}^{N_a^i(x)} \left(\hat{V}_{i+1}^{N_{i+1}}(f(x,a,w_j^a)) - V_{i+1}^*(f(x,a,w_j^a))\right).\end{aligned}$$

Then under the assumption that (2.39) holds, for all $a \in A(x)$ at any sampled $x \in X_s^i$ at stage i,

$$\begin{aligned}P\left(\left|\hat{Q}_i^{N_i}(x,a) - \tilde{Q}_i^{N_i}(x,a)\right| \le \frac{\theta}{2^{i+2}}\right) \\ \ge (1-\delta_{i+1})^{N_a^i(x)} \ge (1-\delta_{i+1})^{N_{i+1}}.\end{aligned} \tag{2.40}$$

This is because if for all w_j^a's, $j = 1, ..., N_a^i(x)$,

$$|V_{i+1}^{N_{i+1}}(f(x,a,w_j^a)) - V_{i+1}^*(f(x,a,w_j^a))| \le \epsilon$$

for $\epsilon > 0$, then

$$\frac{1}{N_a^i(x)} \sum_{j=1}^{N_a^i(x)} \left|V_{i+1}^{N_{i+1}}(f(x,a,w_j^a)) - V_{i+1}^*(f(x,a,w_j^a))\right| \le \epsilon,$$

which further implies

$$\frac{1}{N_a^i(x)}\left|\sum_{j=1}^{N_a^i(x)}\left[V_{i+1}^{N_{i+1}}(f(x,a,w_j^a))-V_{i+1}^*(f(x,a,w_j^a))\right]\right|\le\epsilon,$$

and therefore

$$P\left(\left|\hat{Q}_i^{N_i}(x,a)-\tilde{Q}_i^{N_i}(x,a)\right|\le\epsilon\right)$$
$$\ge\prod_{j=1}^{N_a^i(x)}P\left(\left|V_{i+1}^{N_{i+1}}(f(x,a,w_j^a))-V_{i+1}^*(f(x,a,w_j^a))\right|\le\epsilon\right).$$

From Lemma 2.6, for all $a\in A(x)$, with $N_i>\lambda(\theta/2^{i+2},\delta_i)$ and $\mu_i\in(0,1-2^{-1/N_i})$ for $\delta_i\in(0,1)$,

$$P\left(\left|\tilde{Q}_i^{N_i}(x,a)-Q_i^*(x,a)\right|>\frac{\theta}{2^{i+2}}\right)<\delta_i,\ x\in X_s^i. \tag{2.41}$$

Combining (2.40) and (2.41),

$$P\left(\left|\hat{Q}_i^{N_i}(x,a)-Q_i^*(x,a)\right|\le\frac{\theta}{2^{i+2}}+\frac{\theta}{2^{i+2}}\right)\ge(1-\delta_i)(1-\delta_{i+1})^{N_{i+1}},$$

and this yields that under the supposition of (2.39), for any $x\in X_s^i$,

$$P\left(\left|\hat{Q}_i^{N_i}(x,a)-Q_i^*(x,a)\right|\le\frac{\theta}{2^{i+1}}\right)\ge(1-\delta_i)(1-\delta_{i+1})^{N_{i+1}}.$$

This implies that at the **Output** step,

$$P\left(\max_{a\in A}\left|\hat{Q}_i^{N_i}(x,a)-Q_i^*(x,a)\right|<\frac{\theta}{2}\right)\ge(1-\delta_i)(1-\delta_{i+1})^{N_{i+1}},x\in X_s^i. \tag{2.42}$$

From the definition of θ, if

$$\max_{a\in A}\left|\hat{Q}_i^{N_i}(x,a)-Q_i^*(x,a)\right|<\theta/2,$$

then $\hat{Q}_i^{N_i}(x,a^*)>\hat{Q}_i^{N_i}(x,a)$ for all $a\neq a^*$ with $a^*=\arg\max_{a\in A}Q_i^*(x,a)$ (cf. the proof of Theorem 3.1 in [119]). Therefore, by the definition of $\hat{V}_i^{N_i}(x)$, ($\hat{V}_i^{N_i}(x)=\max_{a\in A}\hat{Q}_i^{N_i}(x,a)=\hat{Q}_i^{N_i}(x,a^*)$ and $V_i^*(x)=Q_i^*(x,a^*)$), with our choice of $N_i>\lambda(\frac{\theta}{2^{i+2}},\delta_i)$ and $\mu_i\in(0,1-2^{-\frac{1}{N_i}})$, we have that

$$P\left(\left|\hat{V}_i^{N_i}(x)-V_i^*(x)\right|>\frac{\theta}{2^{i+1}}\right)<1-(1-\delta_i)(1-\delta_{i+1})^{N_{i+1}}$$

if for all $x\in X_s^{i+1}$, with some N_{i+1},μ_{i+1}, and a given $\delta_{i+1}\in(0,1)$,

$$P\left(\left|\hat{V}_{i+1}^{N_{i+1}}(x)-V_{i+1}^{*}(x)\right|>\frac{\theta}{2^{i+2}}\right)<\delta_{i+1}.$$

Now apply an inductive argument: since $\hat{V}_H^{N_H}(x)=V_H^*(x)=0,\ x\in X$, with $N_{H-1}>\lambda(\theta/2^{H+1},\delta_{H-1})\geq\lambda(\theta/2^H,\delta_{H-1})$ and $\mu_{H-1}\in(0,1-2^{-1/N_{H-1}})$,

$$P\left(\left|\hat{V}_{H-1}^{N_{H-1}}(x)-V_{H-1}^{*}(x)\right|>\frac{\theta}{2^H}\right)<\delta_{H-1},\ x\in X_s^{H-1}.$$

It follows that with $N_{H-2}>\lambda(\theta/2^H,\delta_{H-2})$ and $\mu_{H-2}\in(0,1-2^{-1/N_{H-2}})$,

$$P\left(\left|\hat{V}_{H-2}^{N_{H-2}}(x)-V_{H-2}^{*}(x)\right|>\theta/2^{H-1}\right)<1-(1-\delta_{H-2})(1-\delta_{H-1})^{N_{H-1}}$$

for $x\in X_s^{H-2}$ and further follows that with $N_{H-3}>\lambda(\theta/2^{H-1},\delta_{H-3})$ and $\mu_{H-3}\in(0,1-2^{-1/N_{H-3}})$,

$$P\left(\left|\hat{V}_{H-3}^{N_{H-3}}(x)-V_{H-3}^{*}(x)\right|>\frac{\theta}{2^{H-2}}\right)$$
$$<1-(1-\delta_{H-3})(1-\delta_{H-2})^{N_{H-2}}(1-\delta_{H-1})^{N_{H-2}N_{H-1}}$$

for $x\in X_s^{H-3}$. Continuing this way, we have that

$$P\left(\left|\hat{V}_{1}^{N_1}(x)-V_{1}^{*}(x)\right|>\frac{\theta}{2^2}\right)$$
$$<1-(1-\delta_1)(1-\delta_2)^{N_2}(1-\delta_3)^{N_2N_3}\times\cdots\times(1-\delta_{H-1})^{N_2\cdots N_{H-1}}$$

for $x\in X_s^1$. Finally, with $N_0>\lambda(\theta/4,\delta_0)$ and $\mu_0\in(0,1-2^{-1/N_0})$,

$$P\left(\left|\hat{V}_{0}^{N_0}(x_0)-V_{0}^{*}(x_0)\right|>\frac{\theta}{2}\right)$$
$$<1-(1-\delta_0)(1-\delta_1)^{N_1}\times\cdots\times(1-\delta_{H-1})^{N_1\cdots N_{H-1}},$$

which completes the proof. □

Theorem 2.8. *Assume that Assumption 2 holds. Given* $\delta_i\in(0,1), i=0,...,H-1$, *select* $N_i>\lambda(\theta/2^{i+2},\delta_i)$ *and* $0<\mu_i<\mu_i^*=1-2^{-1/N_i}, i=1,...,H-1$. *If*

$$N_0>\lambda(\theta/4,\delta_0)+\left\lceil\frac{\ln\frac{1}{\epsilon}}{\ln\frac{1}{1-\mu_0^*}}\right\rceil$$

and $0<\mu_0<\mu_0^*=1-2^{-1/\lambda(\theta/4,\delta_0)}$, *then under the PLA sampling algorithm with* ρ *in Equation (2.38), for all* $\epsilon\in(0,1)$,

$$P\left(P_{x_0}(N_0)(a^*)>1-\epsilon\right)>\rho,$$

where $a^*\in\arg\max_{a\in A(x_0)}Q_0^*(x_0,a)$.

Proof. Define the event

$$E'(k) = \{P_{x_0}(k)(a^*) > 1 - \epsilon\},$$

where $a^* = \arg\max_{a \in A} Q_0^*(x_0, a)$. Let $\lambda(\theta/4, \delta_0) = K$. Then,

$$P(E'(\kappa + K)) \geq P(E'(\kappa + K)|E(K))P(E(K)), \kappa = 1, 2, ...,$$

where the event $E(K)$ is given as $\left\{\max_{a \in A} \left|\hat{Q}_i^{N_i}(x, a) - Q_i^*(x, a)\right| < \theta/2\right\}$ at iteration $k = K$.

By selecting $N_0 > K = \lambda(\frac{\theta}{4}, \delta_0)$ and $N_i > \lambda(\frac{\theta}{2^{i+2}}, \delta_i), i = 1, ..., H-1$ and μ_i's for $\delta_i \in (0, 1)$, $P(E(K)) \geq \rho$ by Lemma 2.7. We will obtain l such that $P(E'(\kappa + K)|E(K)) = 1$ if $\kappa > l$, proving the statement of the theorem.

From the choice of $K = \lambda(\theta/4, \delta_0)$, at iteration $N_0 > K$, for each non-optimal action $a \neq a^*$, $P_{x_0}(N_0)(a)$ is decremented by $(1 - \mu_0)$. Therefore, $P_{x_0}(\kappa + K)(a^*) = 1 - \sum_{a \neq a^*} P_{x_0}(K)(a)(1 - \mu_0)^\kappa$ and $\sum_{a \neq a^*} P_{x_0}(K)(a)(1 - \mu_0)^\kappa < \epsilon$ is satisfied if $\kappa > l = \lceil \frac{\ln \epsilon}{\ln(1-\mu_0^*)} \rceil$. □

Based on the proof of Lemma 2.7, the following result follows immediately. We skip the details.

Theorem 2.9. *Assume that Assumption 2 holds. Given* $\delta_i \in (0, 1), i = 0, ..., H-1$ *and* $\epsilon \in (0, \theta]$, *select* $N_i > \lambda(\frac{\epsilon}{2^{i+2}}, \delta_i), 0 < \mu_i < \mu_i^* = 1 - 2^{-\frac{1}{N_i}}, i = 0, ..., H-1$. *Then under the PLA sampling algorithm with* ρ *in Equation (2.38),*

$$P\left(\left|\hat{V}_0^{N_0}(x_0) - V_0^*(x_0)\right| > \frac{\epsilon}{2}\right) < 1 - \rho.$$

From the statements of Lemma 2.7 and Theorems 2.8 and 2.9, the performance of the PLA sampling algorithm depends on the value of θ. If $\theta_i(x)$ is very small or even 0 (failing to satisfy Assumption 2) for some $x \in X$, the PLA sampling algorithm requires a very high sampling complexity to distinguish between the optimal action and the second best action or multiple optimal actions *if* x *is in the sampled tree of* the PLA sampling algorithm. In general, the larger θ is, the more effective the algorithm will be (the smaller the sampling complexity). Therefore, in the actual implementation of the PLA sampling algorithm, if multiple actions' performances are very close after "enough" iterations in the **Loop** portion, it would be advisable to keep only one action among the competitive actions (transferring the probability mass). The parameter θ can thus be viewed as a measure of problem difficulty.

Furthermore, to achieve a certain approximation guarantee at the root level of the sampled tree (i.e., the quality of $\hat{V}_0^{N_0}(x_0)$), we need a *geometric* increase in the accuracies of the optimal reward-to-go values for the sampled states at the lower levels, making it necessary that the total number of samples at the lower levels increases geometrically (N_i depends on $2^{i+2}/\theta$). This is

because the estimate error of $V_i^*(x_i)$ for some $x_i \in X$ affects the estimate of the sampled states in the higher levels in a recursive manner (the error in a level "adds up recursively").

However, the probability bounds in Theorem 2.8 and 2.9 are obtained with coarse estimation of various parameters/terms. For example, we used the worst-case values of $\theta_i(x), x \in X, i = 0, ..., H-1$ and $(R_{\max}H)^2$ for bounding $\sup_{x \in X} V_i^*(x), i = 0, ..., H-1$, and used conservative bounds in (2.40) and in relating the probability bounds for the estimates at the two adjacent levels. Considering this, the performance of the PLA sampling algorithm should probably be more effective in practice than the analysis indicates here.

2.2.3 Application to POMDPs

The simulation model we consider in this chapter covers the dynamics of partially observable MDPs (POMDPs) with finite state, action, and observation spaces, as such a POMDP can be reduced to the equivalent model of an *information-state* MDP, where the state space is the set of all possible probability distributions over the state space of the corresponding POMDP.

Consider a POMDP model parameterized as follows: X is a finite set of states, $A(x)$ is a finite set of admissible actions for each $x \in X$, O is a finite set of observations that provide incomplete state information, and I_0 is the initial information-state, i.e., a probability distribution over X ($I_0(x), x \in X$ denotes the probability of being in state $x \in X$). At stage i, the system is in x_i (where this state information is unknown to the decision maker). The decision maker takes an action a_i, the system makes a transition to x_{i+1} by the probability $P(x_{i+1}|x_i, a_i)$, I_i represents the decision maker's knowledge of x_i, and the decision maker obtains the reward of $r(x_i, a_i, x_{i+1})$. At stage $i+1$, the decision maker observes an observation generated with the probability $O(o_{i+1}|x_{i+1}, a_i)$. The decision maker updates its information-state by

$$I_{i+1}(y) = \eta O(o_{i+1}|y, a_i) \sum_{x \in X} P(y|x, a_i) I_i(x), y \in X,$$

where η is the normalizing constant. From this information-state update procedure, we can induce the probability $P(I_{i+1}|I_i, a_i)$ and map this into a next-state function $h : \Psi \times [0,1] \rightarrow X_I$, where $\Psi = \{(x, a)|x \in X_I, a \in A(x)\}$ and X_I is the set of all possible information-states. The reward function $R_I : \Psi \times [0,1] \rightarrow \Re^+$ is similarly induced. Once the equivalent information-state MDP is constructed, the PLA sampling algorithm can be applied to the information-state MDP. Figure 2.12 presents the modification of the PLA algorithm (cf. Figure 2.10) applied to the unreduced POMDP model.

PLA Sampling Algorithm for POMDPs

Input: stage $i < H$, information state $I_i \in X_i$, $N_i > 0$, $\mu_i \in (0,1)$. (For $i = H$, $\hat{V}_H^{N_H}(x) = V_H^{N_H}(x) = 0$.)

Initialization: Set $P_x(0)(a) = 1/|A(x)|, N_a^i(x) = 0, M_i(x,a) = 0\ \forall a \in A(x)$; $k = 0$.

Loop until $k = N_i$:

- Sample $a(k) \sim P_x(k), y \sim I_i, z \sim P(\cdot|y,a(k)), o \sim O(\cdot|z,a(k))$.
- Obtain the information-state I_{i+1}^k: for $y \in X$,

$$I_{i+1}^k(y) = \eta O(o|y,a(k)) \sum_{y' \in X} P(y|y',a(k)) I_i(y').$$

- Update Q-function estimate for $a = a(k)$ only:

$$\begin{aligned} M_i(x,a(k)) &\leftarrow M_i(x,a(k)) \\ &\quad + \sum_{z \in X} r(x,a(k),z) I_{i+1}^k(z) + \hat{V}_{i+1}^{N_{i+1}}(I_{i+1}^k), \\ N_{a(k)}^i(x) &\leftarrow N_{a(k)}^i(x) + 1, \\ \hat{Q}_i^{N_i}(x,a(k)) &\leftarrow \frac{M_i^{N_i}(x,a(k))}{N_{a(k)}^i(x)}. \end{aligned}$$

- Update optimal action estimate: (ties broken arbitrarily)

$$\hat{a} \in \arg\max_{a \in A(x)} \hat{Q}_i^{N_i}(x,a).$$

- Update probability distribution over action space:

$$P_x(k+1)(a) \leftarrow (1-\mu_i) P_x(k)(a) + \mu_i I\{\hat{a} = a\} \quad \forall a \in A(x).$$

- $k \leftarrow k+1$.

Output: Return $\hat{V}_i^{N_i}(x) = \hat{Q}_i^{N_i}(x,\hat{a})$.

Fig. 2.12. Modified PLA sampling algorithm description for POMDPs

2.2.4 Numerical Example

In this section, we compare the performance of the PLA sampling algorithm with UCB sampling and with the non-adaptive multi-stage sampling (NMS) algorithm in [83], using the inventory control problem of Section 2.1.5. The numerical results for UCB sampling are based on the alternative estimator 2 in Section 2.1.5 given by (2.25). Similar to the PLA and UCB algorithms, NMS is also a simulation-tree based method and estimates the value function at each visited state by taking the minimum of the Q-value estimates. However, the difference between these algorithms is in the way the actions are sampled at each decision period: both the PLA and UCB algorithms sample actions in an adaptive manner, whereas NMS simply samples each action for a fixed number of times.

In the simulation experiments, we consider two cases for the action space, which contains the possible order amounts to be placed: (i) $a_t \in \{0, 5, 10\}$, and (ii) $a_t \in \{0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20\}$, $t = 0, \ldots, H-1$. All other parameter values remain the same as in the examples of Section 2.1.5. For simplicity, the number of samples at each stage, N_i, is taken to be the same for all $i = 0, \ldots, H-1$, and this quantity is denoted by N. Thus, the input parameter μ_i in the PLA algorithm is chosen to be $\mu_i = 1 - 2^{-\frac{1}{N}}$, independent of stage i. In NMS, whenever a state x is visited, each admissible action at x is sampled $\lceil N/|A(x)| \rceil$ times.

The results, based on 30 independent simulation runs for each algorithm, are reported in Tables 2.3 and 2.4. Figures 2.13, 2.14, 2.15, and 2.16 plot the (averaged) value function estimates of the algorithms as a function of the total number of periods simulated. These results indicate that the PLA and UCB algorithms have comparable performance, and both outperform NMS in almost all test cases considered. Moreover, both the PLA and UCB estimates also show a significant reduction in the standard error over the NMS estimate.

Table 2.3. Value function estimates of the PLA, UCB, and NMS algorithms for the inventory control example case (i) as a function of the number of samples at each state: $H = 3, M = 20, x_0 = 5$, $D_t \sim DU(0, 9), h = 1$, where each entry represents the mean based on 30 independent replications (standard error in parentheses)

(K, p)	optimal	N	PLA	UCB	NMS
$K = 0$ $p = 1$	7.700	4	7.61 (0.28)	7.08 (0.29)	6.97 (0.30)
		10	7.57 (0.12)	7.64 (0.10)	7.36 (0.18)
		15	7.63 (0.09)	7.64 (0.08)	7.46 (0.14)
		25	7.70 (0.08)	7.68 (0.08)	7.66 (0.11)
$K = 0$ $p = 10$	16.318	4	14.40 (0.44)	13.13 (0.77)	12.98 (0.50)
		10	16.15 (0.23)	16.58 (0.23)	14.30 (0.35)
		15	16.17 (0.24)	16.34 (0.14)	14.69 (0.35)
		25	16.26 (0.16)	16.45 (0.15)	15.86 (0.20)
$K = 5$ $p = 1$	10.490	4	10.46 (0.27)	10.84 (0.36)	10.05 (0.29)
		10	10.72 (0.10)	10.94 (0.13)	10.50 (0.22)
		15	10.52 (0.09)	10.80 (0.09)	10.70 (0.16)
		25	10.66 (0.07)	10.70 (0.05)	10.54 (0.12)
$K = 5$ $p = 10$	27.322	4	24.48 (0.51)	22.19 (0.76)	21.97 (0.72)
		10	26.25 (0.31)	27.00 (0.24)	24.28 (0.49)
		15	26.55 (0.25)	26.85 (0.23)	25.22 (0.42)
		25	27.19 (0.08)	27.48 (0.08)	26.23 (0.33)

Table 2.4. Value function estimates of the PLA, UCB, and NMS algorithms for the inventory control example case (ii) as a function of the number of samples at each state: $H = 3, M = 20, x_0 = 5$, $D_t \sim DU(0, 9), h = 1$, where each entry represents the mean based on 30 independent replications (standard error in parentheses)

(K, p)	optimal	N	PLA	UCB	NMS
$K = 0$ $p = 1$	7.500	10	6.20 (0.19)	4.20 (0.30)	3.56 (0.28)
		20	6.67 (0.14)	6.99 (0.12)	5.16 (0.18)
		30	7.14 (0.09)	7.32 (0.07)	5.57 (0.16)
		40	7.20 (0.06)	7.34 (0.05)	6.01 (0.16)
$K = 0$ $p = 10$	13.605	10	11.34 (0.28)	6.46 (0.45)	6.57 (0.56)
		20	12.88 (0.26)	13.27 (0.24)	9.48 (0.54)
		30	13.32 (0.17)	13.92 (0.15)	10.02 (0.34)
		40	13.57 (0.14)	14.04 (0.14)	11.53 (0.20)
$K = 5$ $p = 1$	10.490	10	10.98 (0.20)	9.33 (0.32)	9.14 (0.40)
		20	10.98 (0.10)	11.12 (0.09)	10.32 (0.20)
		30	10.86 (0.11)	10.87 (0.05)	9.95 (0.19)
		40	10.80 (0.07)	10.85 (0.05)	10.36 (0.18)
$K = 5$ $p = 10$	25.998	10	23.48 (0.37)	16.29 (0.71)	16.75 (0.74)
		20	24.53 (0.19)	25.68 (0.16)	21.01 (0.47)
		30	25.12 (0.13)	26.19 (0.15)	21.87 (0.34)
		40	25.30 (0.14)	26.17 (0.10)	23.89 (0.22)

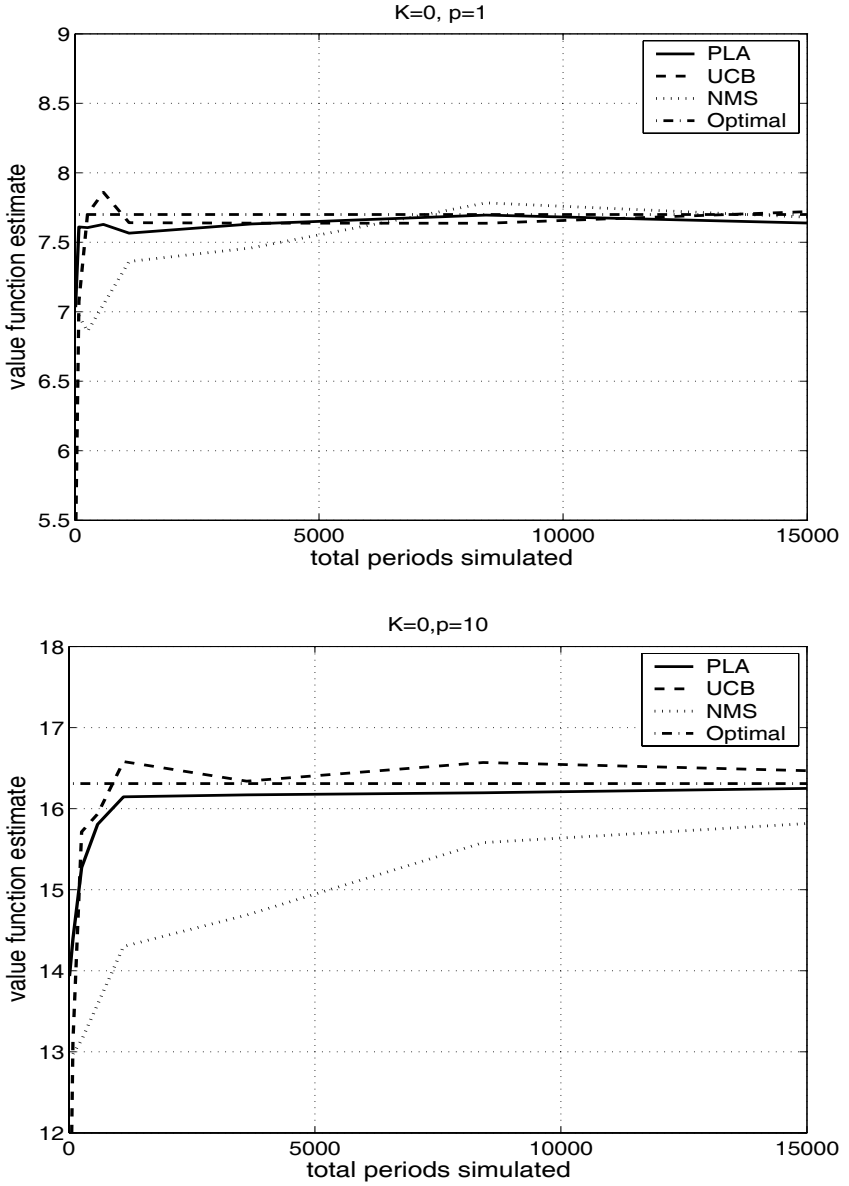

Fig. 2.13. Value function estimates (mean of 30 simulation replications) of the PLA, UCB, and NMS algorithms for the inventory control example case (i) as a function of the number of samples at each state:
$H = 3, M = 20, x_0 = 5, D_t \sim DU(0, 9), h = 1, K = 0$

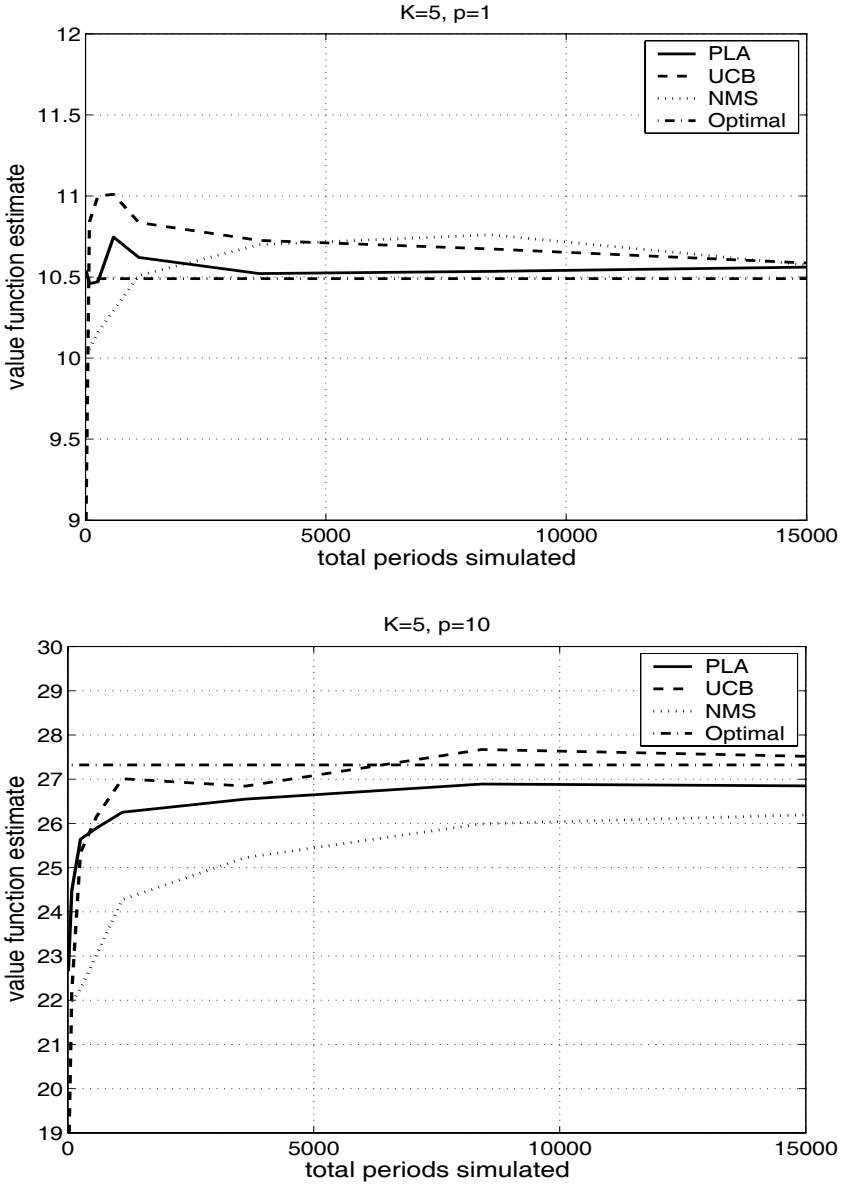

Fig. 2.14. Value function estimates (mean of 30 simulation replications) of the PLA, UCB, and NMS algorithms for the inventory control example case (i) as a function of the number of samples at each state:
$H = 3, M = 20, x_0 = 5, D_t \sim DU(0, 9), h = 1, K = 5$

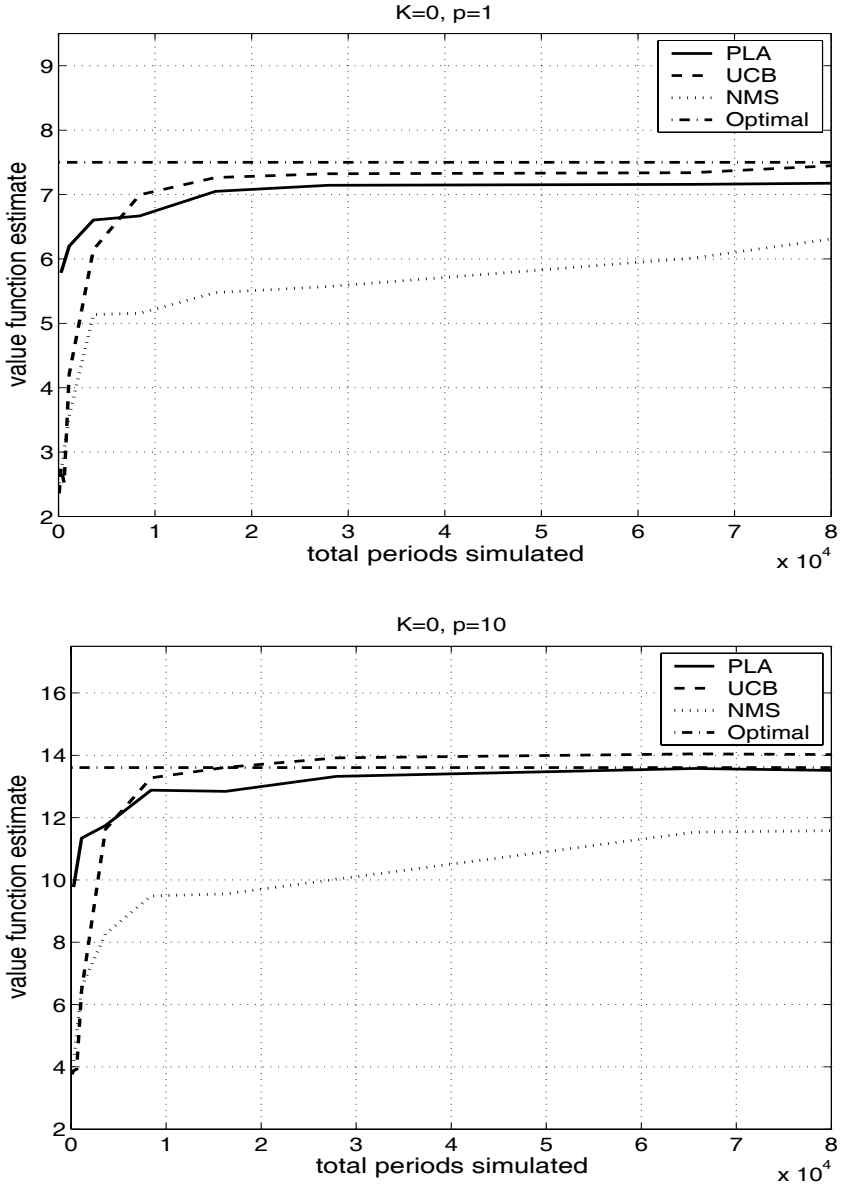

Fig. 2.15. Value function estimates (mean of 30 simulation replications) of the PLA, UCB, and NMS algorithms for the inventory control example case (ii) as a function of the number of samples at each state:
$H = 3, M = 20, x_0 = 5, D_t \sim DU(0, 9), h = 1, K = 0$

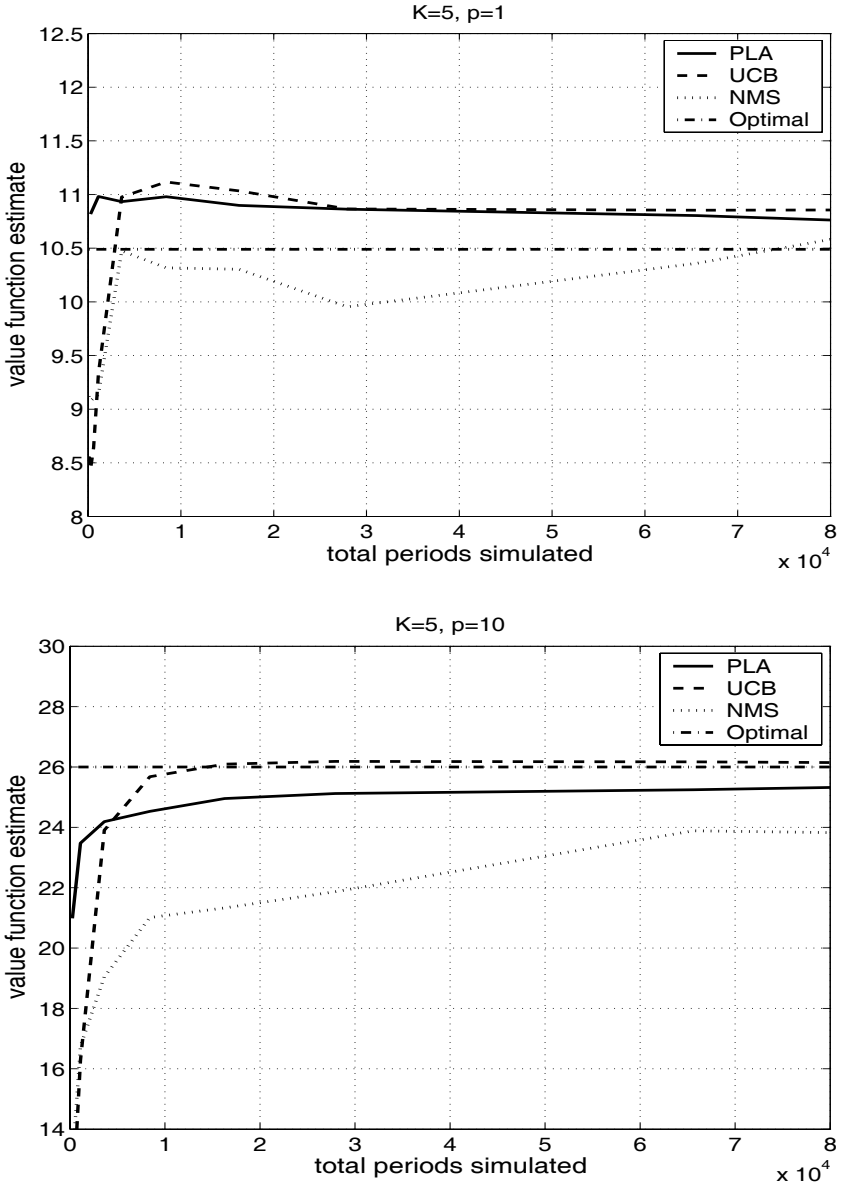

Fig. 2.16. Value function estimates (mean of 30 simulation replications) of the PLA, UCB, and NMS algorithms for the inventory control example case (ii) as a function of the number of samples at each state:
$H = 3, M = 20, x_0 = 5, D_t \sim DU(0, 9), h = 1, K = 5$

2.3 Notes

The expected regret analysis for multi-armed bandit models motivating the UCB sampling algorithm goes back to [93]. The specific index-based policy used here was first proposed in [1], and the finite-time bounds are based on the analysis in [4]. The assumption of bounded rewards can be relaxed by using a result in [1], but the uniform logarithmic bound is not preserved.

The pursuit algorithm designed with learning automata that motivated the PLA sampling algorithm presented here for MDPs was introduced in [140] (see also [110] and [126]). The finite-time analysis of the pursuit algorithm is based on [119], where bounds on the number of iterations and the parameter of the learning algorithm for a given accuracy of performance are provided. General introductory material on learning automata can be found in the book [107] and in the overview survey article [141], whereas application of learning automata for solving controlled (ergodic) Markov chains in a model-free reinforcement learning (RL) framework for a loss function defined on the chains can be found in the books [116], [117]. Controlling ergodic Markov chains for the infinite-horizon average reward within a similar RL framework is considered in [150]. A *uniform* bound on the empirical performance of policies within a simulation model of (partially observable) MDPs is provided in [79]. Reducing a POMDP to an equivalent information-state MDP model can be found in [3].

The UCB sampling algorithm was called the adaptive multi-stage sampling (AMS) algorithm when first introduced in [29]; we chose to change the name in the presentation here, because both of the algorithms in this chapter are multi-stage algorithms with adaptive sampling. The PLA sampling algorithm was originally called the recursive automata sampling algorithm (RASA) in [31]. Again, since both algorithms in this chapter are recursive, we chose the more descriptive "pursuit learning automata" (PLA) label.

Springer

http://www.springer.com/978-1-84628-689-6

Simulation-based Algorithms for Markov Decision Processes

Chang, H.S.; Fu, M.C.; Hu, J.; Marcus, S.I.

2007, XVIII, 189 p. 38 illus., Hardcover
ISBN: 978-1-84628-689-6