Communications and Control Engineering

Hyeong Soo Chang
Jiaqiao Hu
Michael C. Fu
Steven I. Marcus

Simulation-Based Algorithms for Markov Decision Processes

Second Edition

Springer

Communications and Control Engineering

For further volumes:
www.springer.com/series/61

Hyeong Soo Chang • Jiaqiao Hu • Michael C. Fu •
Steven I. Marcus

Simulation-Based Algorithms for Markov Decision Processes

Second Edition

Springer

Hyeong Soo Chang
Dept. of Computer Science and Engineering
Sogang University
Seoul, South Korea

Michael C. Fu
Smith School of Business
University of Maryland
College Park, MD, USA

Jiaqiao Hu
Dept. Applied Mathematics & Statistics
State University of New York
Stony Brook, NY, USA

Steven I. Marcus
Dept. Electrical & Computer Engineering
University of Maryland
College Park, MD, USA

ISSN 0178-5354 Communications and Control Engineering
ISBN 978-1-4471-5021-3 ISBN 978-1-4471-5022-0 (eBook)
DOI 10.1007/978-1-4471-5022-0
Springer London Heidelberg New York Dordrecht

Library of Congress Control Number: 2013933558

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

To Jung Won and three little rascals, Won, Kyeong & Min, who changed my days into a whole world of wonders and joys
– H.S. Chang
To my family – J. Hu
To my mother, for continuous support, and to Lara & David, for mixtures of joy & laughter
– M.C. Fu
To Shelley, Jeremy, and Tobin – S. Marcus

Preface to the 2nd Edition

Markov decision process (MDP) models are widely used for modeling sequential decision-making problems that arise in engineering, computer science, operations research, economics, and other social sciences. However, it is well known that many real-world problems modeled by MDPs have huge state and/or action spaces, leading to the well-known curse of dimensionality, which makes solution of the resulting models intractable. In other cases, the system of interest is complex enough that it is not feasible to explicitly specify some of the MDP model parameters, but simulated sample paths can be readily generated (e.g., for random state transitions and rewards), albeit at a non-trivial computational cost. For these settings, we have developed various sampling and population-based numerical algorithms to overcome the computational difficulties of computing an optimal solution in terms of a policy and/or value function. Specific approaches include multi-stage adaptive sampling, evolutionary policy iteration and random policy search, and model reference adaptive search. The first edition of this book brought together these algorithms and presented them in a unified manner accessible to researchers with varying interests and background. In addition to providing numerous specific algorithms, the exposition included both illustrative numerical examples and rigorous theoretical convergence results. This book reflects the latest developments of the theories and the relevant algorithms developed by the authors in the MDP field, integrating them into the first edition, and presents an updated account of the topics that have emerged since the publication of the first edition over six years ago. Specifically, novel approaches include a stochastic approximation framework for a class of simulation-based optimization algorithms and applications into MDPs and a population-based on-line simulation-based algorithm called approximation stochastic annealing. These simulation-based approaches are distinct from but complementary to those computational approaches for solving MDPs based on explicit state-space reduction, such as neuro-dynamic programming or reinforcement learning; in fact, the computational gains achieved through approximations and parameterizations to reduce the size of the state space can be incorporated into most of the algorithms in this book.

Our focus is on *computational* approaches for calculating or estimating optimal value functions and finding optimal policies (possibly in a restricted policy space). As a consequence, our treatment does not include the following topics found in most books on MDPs:

(i) characterization of fundamental *theoretical* properties of MDPs, such as existence of optimal policies and uniqueness of the optimal value function;
(ii) paradigms for *modeling* complex real-world problems using MDPs.

In particular, we eschew the technical mathematics associated with defining continuous state and action space MDP models. However, we do provide a rigorous theoretical treatment of convergence properties of the algorithms. Thus, this book is aimed at researchers in MDPs and applied probability modeling with an interest in numerical computation. The mathematical prerequisites are relatively mild: mainly a strong grounding in calculus-based probability theory and some familiarity with Markov decision processes or stochastic dynamic programming; as a result, this book is meant to be accessible to graduate students, particularly those in control, operations research, computer science, and economics.

We begin with a formal description of the discounted reward MDP framework in Chap. 1, including both the finite- and infinite-horizon settings and summarizing the associated optimality equations. We then present the well-known exact solution algorithms, value iteration and policy iteration, and outline a framework of rolling-horizon control (also called receding-horizon control) as an approximate solution methodology for solving MDPs, in conjunction with simulation-based approaches covered later in the book. We conclude with a brief survey of other recently proposed MDP solution techniques designed to break the curse of dimensionality.

In Chap. 2, we present simulation-based algorithms for estimating the optimal value function in finite-horizon MDPs with large (possibly uncountable) state spaces, where the usual techniques of policy iteration and value iteration are either computationally impractical or infeasible to implement. We present two adaptive sampling algorithms that estimate the optimal value function by choosing actions to sample in each state visited on a finite-horizon simulated sample path. The first approach builds upon the expected regret analysis of multi-armed bandit models and uses upper confidence bounds to determine which action to sample next, whereas the second approach uses ideas from learning automata to determine the next sampled action. The first approach is also the predecessor of a closely related approach in artificial intelligence (AI) called Monte Carlo tree search that led to a breakthrough in developing the current best computer Go-playing programs (see Sect. 2.3 Notes).

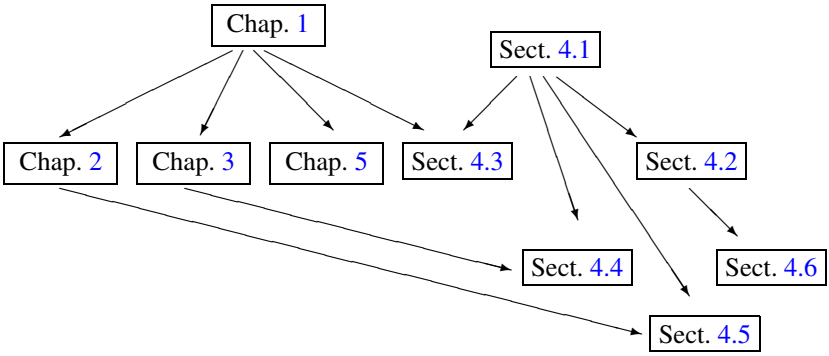
Chapter 3 considers infinite-horizon problems and presents evolutionary approaches for finding an optimal policy. The algorithms in this chapter work with a population of policies—in contrast to the usual policy iteration approach, which updates a single policy—and are targeted at problems with large action spaces (again

possibly uncountable) and relatively small state spaces. Although the algorithms are presented for the case where the distributions on state transitions and rewards are known explicitly, extension to the setting when this is not the case is also discussed, where finite-horizon simulated sample paths would be used to estimate the value function for each policy in the population.

In Chap. 4, we consider a global optimization approach called model reference adaptive search (MRAS), which provides a broad framework for updating a probability distribution over the solution space in a way that ensures convergence to an optimal solution. After introducing the theory and convergence results in a general optimization problem setting, we apply the MRAS approach to various MDP settings. For the finite- and infinite-horizon settings, we show how the approach can be used to perform optimization in policy space. In the setting of Chap. 3, we show how MRAS can be incorporated to further improve the exploration step in the evolutionary algorithms presented there. Moreover, for the finite-horizon setting with both large state and action spaces, we combine the approaches of Chaps. 2 and 4 and propose a method for sampling the state and action spaces. Finally, we present a stochastic approximation framework for studying a class of simulation- and sampling-based optimization algorithms. We illustrate the framework through an algorithm instantiation called model-based annealing random search (MARS) and discuss its application to finite-horizon MDPs.

In Chap. 5, we consider an approximate rolling-horizon control framework for solving infinite-horizon MDPs with large state/action spaces in an on-line manner by simulation. Specifically, we consider policies in which the system (either the actual system itself or a simulation model of the system) evolves to a particular state that is observed, and the action to be taken in that particular state is then computed on-line at the decision time, with a particular emphasis on the use of simulation. We first present an updating scheme involving multiplicative weights for updating a probability distribution over a restricted set of policies; this scheme can be used to estimate the optimal value function over this restricted set by sampling on the (restricted) policy space. The lower-bound estimate of the optimal value function is used for constructing on-line control policies, called (simulated) policy switching and parallel rollout. We also discuss an upper-bound based method, called hindsight optimization. Finally, we present an algorithm, called approximate stochastic annealing, which combines Q-learning with the MARS algorithm of Section 4.6.1 to directly search the policy space.

The relationship between the chapters and/or sections of the book is shown below. After reading Chap. 1, Chaps. 2, 3, and 5 can pretty much be read independently, although Chap. 5 does allude to algorithms in each of the previous chapters, and the numerical example in Sect. 5.1 is taken from Sect. 2.1. The first two sections of Chap. 4 present a general global optimization approach, which is then applied to MDPs in the subsequent Sects. 4.3, 4.4 and 4.5, where the latter two build upon work in Chaps. 3 and 2, respectively. The last section of Chap. 4 deals with a stochastic approximation framework for a class of optimization algorithms and its applications to MDPs.



Finally, we acknowledge the financial support of several US Federal funding agencies for this work: the National Science Foundation (under Grants DMI-9988867, DMI-0323220, CMMI-0900332, CNS-0926194, CMMI-0856256, EECS-0901543, and CMMI-1130761), the Air Force Office of Scientific Research (under Grants F496200110161, FA95500410210, and FA95501010340), and the Department of Defense.

Seoul, South Korea — Hyeong Soo Chang
Stony Brook, NY, USA — Jiaqiao Hu
College Park, MD, USA — Michael Fu
College Park, MD, USA — Steve Marcus

Contents

Selected Notation and Abbreviations[1]

$\Re$ ($\Re^+$)	set of (non-negative) real numbers
$\mathcal{Z}$ ($\mathcal{Z}^+$)	set of (positive) integers
H	horizon length (number of stages or periods)
X	state space
A	action space
$A(x)$	admissible action space in state x
$P(x, a)(y)$	probability of transitioning to state y from state x when taking action a
$f(x, a, u)$	next state reached from state x when taking action a for random number u
$R(x, a)$	non-negative bounded reward obtained in state x when taking action a
$C(x, a)$	non-negative bounded cost obtained in state x when taking action a
$R'(x, a, w)$	non-negative bounded reward obtained in state x when taking action a for random number w
$R_{\max}$	upper bound on one-period reward
γ	discount factor $\in (0, 1]$
π	policy (a sequence of mappings prescribing an action to take for each state)
$\pi_i(x)$	action prescribed for state x in stage i under policy π
$\pi(x)$	action prescribed for state x (under stationary policy π)
π^*	an optimal policy
$\hat{\pi}^k$	an estimated optimal policy at kth iteration
Π	set of all non-stationary Markovian policies
Π_s	set of all stationary Markovian policies: (1.10)
$V_i^*(x)$	optimal reward-to-go value from stage i in state x: (1.5)

[1] Notation specific to a particular chapter is noted parenthetically. Equation numbers indicate where the quantity is defined.

V_i^*	optimal reward-to-go value function from stage i
$\hat{V}_i^{N_i}$	estimated optimal reward-to-go value function from stage i based on N_i simulation replications in that stage
$V^*(x)$	optimal value for starting state x: (1.2)
V^*	optimal value function
V_i^π	reward-to-go value function for policy π from stage i: (1.6)
V^π	value function for policy π: (1.11)
$\mathcal{V}_H^\pi(x)$	expected total discounted reward over horizon length H under policy π, starting from state x $(=V_0^\pi(x))$
$Q_i^*(x,a)$	Q-function value giving expected reward for taking action a from state x in stage i, plus expected total discounted optimal reward-to-go value from next state reached in stage $i+1$: (1.9)
$Q^*(x,a)$	infinite-horizon Q-function value: (1.14)
$\hat{Q}_i^{N_i}(x,a)$	estimate for $Q_i^*(x,a)$ based on N_i samples
$\hat{Q}(x,a)$	estimate for $Q^*(x,a)$
$\mathcal{P}_x$	action selection distribution over $A(x)$
a.s.	almost sure(ly)
c.d.f.	cumulative distribution function
i.i.d.	independent and identically distributed
p.d.f.	probability density function
p.m.f.	probability mass function
s.t.	such that (or subject to)
w.p.	with probability
w.r.t.	with respect to
$U(a,b)$	(continuous) uniform distribution with support on $[a,b]$
$DU(a,b)$	discrete uniform distribution on $\{a, a+1, \ldots, b-1, b\}$
$N(\mu, \sigma^2)$	normal (Gaussian) distribution with mean (vector) μ and variance σ^2 (covariance matrix Σ)
E_f	expectation under p.d.f. f (Chap. 4)
E_θ, P_θ	expectation/probability under p.d.f./p.m.f. $f(\cdot,\theta)$ (Chap. 4)
$\tilde{E}_\theta$, $\tilde{P}_\theta$	expectation/probability under p.d.f./p.m.f. $\tilde{f}(\cdot,\theta)$ (Chap. 4)
$\forall$	for all
$\exists$	there exists
$\mathcal{D}(\cdot,\cdot)$	Kullback–Leibler (KL) divergence between two p.d.f.s/p.m.f.s (Chaps. 4, 5)
$d(\cdot,\cdot)$	distance metric (Chap. 3)
$d_\infty(\cdot,\cdot)$	infinity-norm distance between two policies (Chap. 3)
$d_T(\cdot,\cdot)$	total variation distance between two p.m.f.s (Chap. 5)
NEF	natural exponential family (Chap. 4)
$:=$	equal by definition
$\stackrel{d}{=}$	equal in distribution
$\Longleftrightarrow$	if and only if
$\Longrightarrow$	implies (or weak convergence)
$I\{\cdot\}$	indicator function of the set $\{\cdot\}$

$\lvert X\rvert$	cardinality (number of elements) of set X
$\lVert\cdot\rVert$	norm of a function or vector, or induced norm of a matrix
$x \vee y$	$\max(x, y)$
$x \wedge y$	$\min(x, y)$
x^+	$\max(x, 0)$
x^-	$\min(-x, 0)$
$\lceil x \rceil$	least integer greater than or equal to x
$\lfloor x \rfloor$	greatest integer less than or equal to x
$f(n) = O(g(n))$	$\limsup_{n\to\infty} \frac{f(n)}{g(n)} < \infty$
$f(n) = \Theta(g(n))$	$f(n) = O(g(n))$ and $g(n) = O(f(n))$

Chapter 1
Markov Decision Processes

Define a Markov decision process (MDP) by the five-tuple $(X, A, A(\cdot), P, R)$, where X denotes the state space, A denotes the action space, $A(x) \subseteq A$ is the set of admissible actions in state x, $P(x,a)(y)$ is the probability of transitioning from state $x \in X$ to state $y \in X$ when action $a \in A(x)$ is taken, and $R(x,a)$ is the reward obtained when in state $x \in X$ and action $a \in A(x)$ is taken. We will assume throughout the book that the reward is non-negative and bounded, i.e., $0 \leq R(x,a) \leq R_{\max}$ for all $x \in X, a \in A(x)$. More generally, $R(x,a)$ may itself be a random variable, or viewed as the (conditioned on x and a) expectation of an underlying random reward. For simplicity and mathematical rigor, we will usually assume that X is a countable set, but the discussion and notation can be generalized to uncountable state spaces. We have assumed that the components of the model are stationary (not explicitly time-dependent); the nonstationary case can be incorporated into this model by augmenting the state with a time variable. Note that an equivalent model description is done with a cost function C such that $C(x,a)$ is the cost obtained when in state $x \in X$ and action $a \in A(x)$ is taken, in which case a minimum/infimum operator needs to replace a maximum/supremum operator in appropriate places below.

The evolution of the system is as follows (see Fig. 1.1). Let x_t denote the state at time (*stage* or *period*) $t \in \{0, 1, \ldots\}$ and a_t the action chosen at that time. If $x_t = x \in X$ and $a_t = a \in A(x)$, then the system transitions from state x to state $x_{t+1} = y \in X$ with probability $P(x,a)(y)$, and a reward of $R(x,a)$ is obtained. Once the transition to the next state has occurred, a new action is chosen, and the process is repeated.

Let Π be the set of non-stationary Markovian policies $\pi = \{\pi_t, t = 0, 1, \ldots\}$, where $\pi_t : X \to A$ is a function such that $\pi_t(x) \in A(x)$ for each $x \in X$. The goal is to find a policy π that maximizes the *expected total discounted reward* given by

$$V^{\pi}(x) = E\left[\sum_{t=0}^{H-1} \gamma^t R\big(x_t, \pi_t(x_t)\big) \,\middle|\, x_0 = x\right], \tag{1.1}$$

for some given initial state $x \in X$, where $0 < \gamma \leq 1$ is the discount factor, and H may be infinite, in which case we require $\gamma < 1$. The *optimal value function* is

H.S. Chang et al., *Simulation-Based Algorithms for Markov Decision Processes*, Communications and Control Engineering,
DOI 10.1007/978-1-4471-5022-0_1,

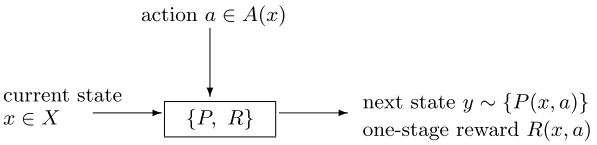


Fig. 1.1 MDP "standard" model

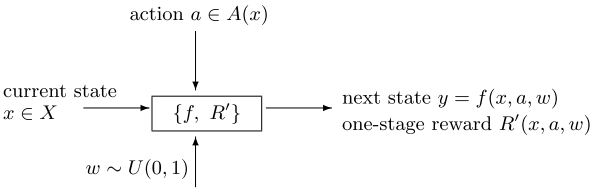


Fig. 1.2 MDP simulation model

denoted by $V^* : X \to \Re^+$, where the optimal value for a given state $x \in X$ is given by

$$V^*(x) = \sup_{\pi \in \Pi} V^\pi(x), \tag{1.2}$$

and a corresponding optimal policy yielding that optimal value function will be denoted by π^*, where

$$V^*(x) = V^{\pi^*}(x), \quad x \in X. \tag{1.3}$$

We will also describe an MDP using a *simulation model*, denoted by $(X, A, A(\cdot), f, R')$, where f is the next-state transition function such that the system dynamics are given by

$$x_{t+1} = f(x_t, a_t, w_t) \quad \text{for } t = 0, 1, \ldots, H-1, \tag{1.4}$$

and $R'(x_t, a_t, w_t) \leq R_{\max}$ is the associated non-negative reward, where $x_t \in X$, $a_t \in A(x)$, and $\{w_t\}$ is an i.i.d. (random number) sequence distributed $U(0, 1)$, representing the uncertainty in the system (see Fig. 1.2). Thus, the simulation model assumes a single random number for both the reward and next-state transition in each period. The expected discounted reward to be maximized is given by (1.1) with R replaced by R' and the expectation taken over the random sequence $\{w_t, t = 0, 1, \ldots\}$, and the optimal value function is still given by (1.2), with a corresponding optimal policy satisfying (1.3). Note that any simulation model $(X, A, A(\cdot), f, R')$ with dynamics (1.4) can be transformed into a model $(X, A, A(\cdot), P, R)$ with state transition function P. Conversely a standard MDP model $(X, A, A(\cdot), P, R)$ can be represented as a simulation model $(X, A, A(\cdot), f, R')$.

1.1 Optimality Equations

For the finite-horizon problem ($H < \infty$), we define the *optimal reward-to-go value* for state $x \in X$ in stage i by

$$V_i^*(x) = \sup_{\pi \in \Pi} V_i^\pi(x), \tag{1.5}$$

where the *reward-to-go value for policy* π for state x in stage i is defined by

$$V_i^\pi(x) = E\left[\sum_{t=i}^{H-1} \gamma^{t-i} R\big(x_t, \pi_t(x_t)\big) \middle| x_i = x\right], \tag{1.6}$$

$i = 0, \ldots, H-1$, with $V_H^*(x) = 0$ for all $x \in X$. Note that $V^\pi(x) = V_0^\pi(x)$ and $V^*(x) = V_0^*(x)$, where V^π and V^* are the *value function for* π and the optimal value function, respectively. It is well known that V_i^* can be written recursively as follows: for all $x \in X$ and $i = 0, \ldots, H-1$,

$$V_i^*(x) = \sup_{a \in A(x)} \left\{ R(x,a) + \gamma \sum_{y \in X} P(x,a)(y) V_{i+1}^*(y) \right\}, \tag{1.7}$$

or, equivalently, by defining the Q-function,

$$V_i^*(x) = \sup_{a \in A(x)} Q_i^*(x,a), \tag{1.8}$$

$$Q_i^*(x,a) = R(x,a) + \gamma \sum_{y \in X} P(x,a)(y) V_{i+1}^*(y). \tag{1.9}$$

The solution of these optimality equations is usually referred to as (stochastic) dynamic programming, which yields the optimal value as defined by Eq. (1.2) for a given initial state x_0:

$$V_0^*(x_0) = \sup_{\pi \in \Pi} V_0^\pi(x_0).$$

Simulation-based methods for estimating this optimal value for a given initial state are the focus of Chap. 2, where simulation will be required to estimate $Q_i^*(x,a)$ as expressed by the simulation model equivalent of Eq. (1.9) given by Eq. (1.17) below, and an adaptive sampling procedure will be used to determine which actions to simulate to estimate $V_i^*(x)$.

For an infinite-horizon MDP ($H = \infty$), we consider the set $\Pi_s \subseteq \Pi$ of all stationary Markovian policies such that

$$\Pi_s = \left\{ \pi \in \Pi \mid \pi_t = \pi_{t'} \; \forall t, t' \right\}, \tag{1.10}$$

since under mild regularity conditions, an optimal policy always exists in Π_s for the infinite-horizon problem. In a slight abuse of notation, we use π for the policy $\{\pi, \pi, \ldots, \}$ for the infinite-horizon problem, and we define the *optimal value*

associated with an initial state $x \in X$: $V^*(x) = \sup_{\pi \in \Pi_s} V^\pi(x), x \in X$, where for $x \in X, 0 < \gamma < 1, \pi \in \Pi_s$,

$$V^\pi(x) = E\left[\sum_{t=0}^{\infty} \gamma^t R\big(x_t, \pi(x_t)\big) \middle| x_0 = x\right], \tag{1.11}$$

for which the well-known Bellman optimality principle holds as follows. For all $x \in X$,

$$V^*(x) = \sup_{a \in A(x)} \left\{ R(x,a) + \gamma \sum_{y \in X} P(x,a)(y) V^*(y) \right\}, \tag{1.12}$$

where $V^*(x), x \in X$, is unique, and there exists an optimal policy $\pi^* \in \Pi_s$ satisfying

$$\pi^*(x) \in \arg\sup_{a \in A(x)} \left\{ R(x,a) + \gamma \sum_{y \in X} P(x,a)(y) V^*(y) \right\}, \quad x \in X, \tag{1.13}$$

and $V^{\pi^*}(x) = V^*(x)$ for all $x \in X$.

In order to simplify the notation, we use V^* and V^π to denote the optimal value function and value function for policy π, respectively, in both the finite and infinite-horizon settings.

Define

$$Q^*(x,a) = R(x,a) + \gamma \sum_{y \in X} P(x,a)(y) V^*(y), \quad x \in X,\ a \in A(x). \tag{1.14}$$

Then it immediately follows that

$$\sup_{a \in A(x)} Q^*(x,a) = V^*(x), \quad x \in X,$$

and that Q^* satisfies the following fixed-point equation: for $x \in X,\ a \in A(x)$,

$$Q^*(x,a) = R(x,a) + \gamma \sum_{y \in X} P(x,a)(y) \sup_{a' \in A(y)} Q^*\big(y, a'\big). \tag{1.15}$$

Our goal for infinite-horizon problems is to find an (approximate) optimal policy $\pi^* \in \Pi_s$ that achieves the (approximate) optimal value for any given initial state.

For a simulation model $(X, A, A(\cdot), f, R')$ with dynamics (1.4), the reward-to-go value for policy π for state x in stage i over a horizon H corresponding to (1.6) is given by

$$V_i^\pi(x) = E\left[\sum_{t=i}^{H-1} \gamma^{t-i} R'\big(x_t, \pi_t(x_t), w_t\big) \middle| x_i = x\right], \tag{1.16}$$

where $x \in X$, $x_t = f(x_{t-1}, \pi_{t-1}(x_{t-1}), w_{t-1})$ is a random variable denoting the state at stage t following policy π, and $w_i, \ldots, w_{H-1}$ are i.i.d. $U(0,1)$. The corresponding optimal reward-to-go value V_i^* is defined by (1.5), satisfying

$$V_i^*(x) = \sup_{a \in A(x)} \{E[R'(x,a,U)] + \gamma E[V_{i+1}^*(f(x,a,U))]\}, \quad U \sim U(0,1),$$

which can be expressed as in (1.8) in terms of the Q-function defined analogously to (1.9) as follows:

$$Q_i^*(x,a) = E[R'(x,a,U)] + \gamma E[V_{i+1}^*(f(x,a,U))], \quad U \sim U(0,1). \tag{1.17}$$

For notational simplification, we will often drop the explicit dependence on U or w_j whenever there is an expectation involved, e.g., we would simply write Eq. (1.17) as

$$Q_i^*(x,a) = E[R'(x,a)] + \gamma E[V_{i+1}^*(f(x,a))],$$

where the expectation is understood to be with respect to the randomness in the one-stage reward(s) and next-state transition(s). Using this notation, we write the corresponding infinite-horizon relationships for the simulation model:

$$\begin{aligned}
V^*(x) &= \sup_{a \in A(x)} E[R'(x,a) + \gamma V^*(f(x,a))] = \sup_{a \in A(x)} Q^*(x,a), \\
\pi^*(x) &\in \arg\sup_{a \in A(x)} E[R'(x,a) + \gamma V^*(f(x,a))] = \arg\sup_{a \in A(x)} Q^*(x,a), \\
Q^*(x,a) &= E[R'(x,a)] + \gamma E[V^*(f(x,a))], \\
&= E[R'(x,a)] + \gamma E\Big[\sup_{a' \in A(f(x,a))} Q^*(f(x,a),a')\Big].
\end{aligned}$$

In the remainder of the chapter, we include the expressions for both the MDP standard and simulation models.

1.2 Policy Iteration and Value Iteration

Policy iteration and value iteration are the two most well-known techniques for determining the optimal value function V^* and/or a corresponding optimal policy π^* for infinite-horizon problems. Before presenting each, we introduce some notation. Let $B(X)$ be the space of bounded real-valued functions on X. For $\Phi \in B(X)$, $x \in X$, we define an operator $T : B(X) \to B(X)$ by

$$T(\Phi)(x) = \sup_{a \in A(x)} \left\{ R(x,a) + \gamma \sum_{y \in X} P(x,a)(y)\Phi(y) \right\}, \tag{1.18}$$

$$T(\Phi)(x) = \sup_{a \in A(x)} E[R'(x,a) + \gamma \Phi(f(x,a))], \tag{1.19}$$

for the standard and simulation models, respectively. Similarly, we define an operator $T_\pi : B(X) \to B(X)$ for $\pi \in \Pi_s$ by

$$T_\pi(\Phi)(x) = R\big(x, \pi(x)\big) + \gamma \sum_{y \in X} P\big(x, \pi(x)\big)(y)\Phi(y), \tag{1.20}$$

$$T_\pi(\Phi)(x) = E\big[R'\big(x, \pi(x)\big)\big] + \gamma E\big[\Phi\big(f\big(x, \pi(x)\big)\big)\big]. \tag{1.21}$$

We begin with policy iteration. Each step of policy iteration consists of two parts: policy evaluation and policy improvement. Each iteration preserves monotonicity in terms of the policy performance.

Policy evaluation is based on the result that for any policy $\pi \in \Pi_s$, there exists a corresponding unique $\Phi \in B(X)$ such that for $x \in X$, $T_\pi(\Phi)(x) = \Phi(x)$ and $\Phi(x) = V^\pi(x)$. The policy evaluation step obtains V^π for a given $\pi \in \Pi_s$ by solving the corresponding fixed-point functional equation over all $x \in X$:

$$V^\pi(x) = R\big(x, \pi(x)\big) + \gamma \sum_{y \in X} P\big(x, \pi(x)\big)(y)V^\pi(y), \tag{1.22}$$

$$V^\pi(x) = E\big[R'\big(x, \pi(x)\big)\big] + \gamma E\big[V^\pi\big(f\big(x, \pi(x)\big)\big)\big], \tag{1.23}$$

which, for finite X, is just a set of $|X|$ linear equations in $|X|$ unknowns.

The policy improvement step takes a given policy π and obtains a new policy $\hat{\pi}$ by satisfying the condition $T(V^\pi)(x) = T_{\hat{\pi}}(V^\pi)(x)$, $x \in X$, i.e., for each $x \in X$, by taking the action

$$\hat{\pi}(x) \in \arg\sup_{a \in A(x)} \left\{ R(x, a) + \gamma \sum_{y \in X} P(x, a)(y)V^\pi(y) \right\}, \tag{1.24}$$

$$\hat{\pi}(x) \in \arg\sup_{a \in A(x)} \big\{ E\big[R'(x, a)\big] + \gamma E\big[V^\pi\big(f(x, a)\big)\big] \big\}. \tag{1.25}$$

The policy improvement step ensures that the value function of $\hat{\pi}$ is no worse than that of π, i.e.,

$$V^{\hat{\pi}}(x) \geq V^\pi(x) \quad \forall x \in X.$$

Starting with an arbitrary policy $\pi_0 \in \Pi_s$, at each iteration $k \geq 1$, policy iteration applies the policy evaluation and policy improvement steps alternately until $V^{\pi_k}(x) = V^{\pi_{k-1}}(x)$ $\forall x \in X$, in which case an optimal policy has been found. For finite policy spaces, and thus in particular for finite state and action spaces, policy iteration guarantees convergence to an optimal solution in a finite number of steps.

Value iteration iteratively updates a given value function by applying the operator T successively, i.e., for $v \in B(X)$, a new value function is obtained by computing for each $x \in X$,

$$\hat{v}(x) = \sup_{a \in A(x)} \left\{ R(x, a) + \gamma \sum_{y \in X} P(x, a)(y)v(y) \right\},$$

$$\hat{v}(x) = \sup_{a \in A(x)} \left\{ E\left[R'(x,a)\right] + \gamma E\left[v\left(f(x,a)\right)\right] \right\}.$$

Let $\{v_n\}$ be the sequence of value iteration functions defined by $v_n = T(v_{n-1})$, where $n = 1, 2, \ldots$ and $v_0 \in B(X)$ is arbitrary. Then for any $n = 0, 1, \ldots$, the value iteration function v_n satisfies $\|v_n - V^*\| \leq \gamma^n \|v_0 - V^*\|$, i.e., T is a contraction mapping and successive applications of T will lead to v_n converging to V^* by Banach's fixed-point theorem. Thus, value iteration is often called the method of *successive approximations*. In particular, taking $v_0 = 0$, v_n is equal to the optimal reward-to-go value function V^*_{H-n} for the finite-horizon problem, where this procedure is called "backward induction." Unlike policy iteration, however, value iteration may require an infinite number of iterations to converge, even when the state and action spaces are finite.

The running-time complexity of value iteration is polynomial in $|X|$, $|A|$, $1/(1-\gamma)$; in particular, one iteration is $O(|X|^2|A|)$ in the size of the state and action spaces. Even though the single iteration running-time complexity $O(|X|^2|A|)$ of value iteration is smaller than the corresponding $O(|X|^2|A| + |X|^3)$ single-iteration time complexity of policy iteration, the number of iterations required for value iteration can be very large—possibly infinite, as just mentioned.

1.3 Rolling-Horizon Control

In this section, we consider an approximation framework for solving infinite-horizon MDP problems. This rolling-horizon control (also called *receding*-horizon control) framework will be discussed together with simulation-based approaches in Chap. 5. The idea of rolling-horizon control can be used to solve problems in an on-line manner, where an optimal exact solution with respect to a fixed-length moving horizon at each decision time is obtained and its initial action is applied to the system. The intuition behind the approach is that if the horizon is sufficiently long so as to provide a good estimate of the stationary behavior of the system, the moving-horizon control should perform well. Indeed, the value of the rolling-horizon policy converges geometrically to the optimal value, uniformly in the initial state, as the length of the moving horizon increases, where the convergence rate is characterized by the discount factor (cf. Theorem 1.1 below).

Furthermore, under mild conditions, there always exists a minimal finite horizon H^* such that the rolling-H^*-horizon control prescribes exactly the same action as the policy that achieves the optimal infinite-horizon rewards at every state.

A rolling-H-horizon control policy π_{rh} is a stationary policy for the infinite-horizon problem that is obtained from an optimal non-stationary policy $\{\pi^*_0, \ldots, \pi^*_{H-1}\}$ for the finite-horizon problem of length $H < \infty$, by taking $\pi_{\text{rh}} = \pi^*_0$, i.e., for a given starting state $x \in X$, it satisfies the initial stage optimality equation

$$V^*_0(x) = R\left(x, \pi_{\text{rh}}(x)\right) + \gamma \sum_{y \in X} P\left(x, \pi_{\text{rh}}(x)\right)(y) V^*_1(y),$$

$$V_0^*(x) = E\big[R'\big(x, \pi_{\rm rh}(x)\big)\big] + \gamma E\big[V_1^*\big(f\big(x, \pi_{\rm rh}(x)\big)\big)\big],$$

or using the notation of the previous section,

$$T_{\pi_{\rm rh}}\big(V_1^*\big)(x) = T\big(V_1^*\big)(x), x \in X,$$

where V_1^* is the optimal reward-to-go function for the finite-H-horizon MDP beginning in stage one. Although the rolling-H-horizon policy $\pi_{\rm rh}$ is determined using the finite-horizon MDP model, it is applied in the infinite-horizon setting. The following result bounds the error between the true (infinite-horizon) optimal value function and the value function associated with the rolling-horizon policy, providing an explicit characterization of the geometric convergence rate in the discount factor with respect to the horizon length.

Theorem 1.1 (Hernández-Lerma and Lasserre [84])

$$0 \le V^*(x) - V^{\pi_{\rm rh}}(x) \le \frac{R_{\max}}{1-\gamma} \cdot \gamma^H, \quad x \in X.$$

Again, we reiterate that here V^* and $V^{\pi_{\rm rh}}$ denote *infinite*-horizon value functions, whereas what is used to determine the stationary policy $\pi_{\rm rh}$ is a finite-horizon optimal reward-to-go function V_1^*. Unfortunately, a large state space makes it very difficult to solve such MDPs in practice even with a relatively small rolling horizon. Motivated by this, we provide in Chap. 5 an error bound for approximate rolling-horizon control defined from an estimate of V_1^*. In addition, in Chap. 2, we present adaptive sampling simulation-based algorithms that estimate V_1^*, and in Chap. 5, we study two approximate rolling-horizon controls via lower and upper bounds to V_1^*, both implemented in numerical examples by simulation.

1.4 Survey of Previous Work on Computational Methods

While an optimal policy can, in principle, be obtained by the methods of dynamic programming, policy iteration, and value iteration, such computations are often prohibitively time-consuming. In particular, the size of the state space grows exponentially with the number of *state variables*, a phenomenon referred to by Bellman as the *curse of dimensionality*. Similarly, the size of the action space can also lead to computational intractability. Lastly, the transition function/probabilities (f or P) and/or random rewards may not be explicitly known, but a simulation model may be available for producing sample paths, which means that traditional approaches cannot be applied. These diverse computational challenges have given rise to a number of approaches intended to result in more tractable computations for estimating the optimal value function and finding optimal or good suboptimal policies. Some of these approaches can be categorized as follows:

1. structural analysis and proof of structural properties;

2. approximating the problem with a simpler problem;
3. approximating the dynamic programming equations or the value function;
4. algorithms in policy space.

The first approach can be exact, and involves the use of structural properties of the problem or the solution, such as monotonicity, convexity, modularity, or factored representations, to facilitate the process of finding an optimal solution or policy.

The remaining approaches all involve approximations or suboptimal policies. The second class of approaches can involve (i) approximation of the model with a simpler model (e.g., via state aggregation, linearization, or discretization, or (ii) restricting the structure of the policies (e.g., linear policies, certainty equivalent policies, or open-loop feedback-control policies). The third approach is to approximate the value function and/or the dynamic programming equations using techniques such as state aggregation, basis function representations, and feature extraction. The fourth class includes algorithms that work in policy space like policy iteration, but are intended to provide more tractable algorithms than policy iteration. The algorithms presented in this book use randomization, sampling, or simulation in the context of the third and fourth approaches listed above.

To put the approaches of this book in context, we briefly compare them with some other important randomized/simulation-based methods. Most of this work has involved approximate solution of the dynamic programming equations or approximation of value functions, and is referred to as *reinforcement learning* or *neuro-dynamic programming*.

Q-learning, perhaps the most well-known example of reinforcement learning, is a stochastic-approximation-based solution approach to solving (1.15). It is a model-free approach that works for the case in which the parameters of the transition function f (or transition probabilities P) and one-stage reward function R are unknown. In asynchronous Q-learning, a sequence of estimates $\{\hat{Q}\}$ of Q^* is constructed as follows. At time t, the decision maker observes state x_t and takes an action $a_t \in A(x_t)$ chosen according to a *randomized* policy (a randomized policy is a generalized type of policy, in which, for an observed state x_t, an action is chosen randomly from a probability distribution over $A(x_t)$). The decision maker receives the reward $R'(x_t, a_t, w_t)$, moves to state $f(x_t, a_t, w_t)$, where $w_t \sim U(0, 1)$, and updates the Q-value estimate at (x_t, a_t) by

$$\hat{Q}(x_t, a_t) \leftarrow \hat{Q}(x_t, a_t) + \alpha_t(x_t, a_t)\Big[R'(x_t, a_t, w_t) + \gamma \sup_{a' \in A(f(x_t, a_t, w_t))} \hat{Q}\big(f(x_t, a_t, w_t), a'\big) - \hat{Q}(x_t, a_t)\Big],$$

where $\alpha_t(x_t, a_t)$ is a non-negative stepsize coefficient. Note that at each step, only a single value of the Q-function estimate is updated.

Under fairly general conditions, $\{\hat{Q}\}$ will converge to the function Q^* for finite state and action MDPs. A key requirement is that the randomized policy should ensure that each state is visited infinitely often and every action is taken (explored)

in every state infinitely often. Only limited results exist for the rate of convergence of Q-learning, although it is well known that the convergence of stochastic-approximation-based algorithms for solving MDPs can be quite slow. Furthermore, because Q-learning is implemented with a lookup table of size $|X| \times |A|$, it suffers from the curse of dimensionality.

Another important aspect of the work involves approximating the optimal value function V^* using, for example, neural networks and/or simulation. $V^*(x), x \in X$, is replaced with a suitable function approximation $\tilde{V}(x, r)$, called a "scoring function," where r is a vector of parameters, and an approximate optimal policy is obtained by taking an action in

$$\arg\sup_{a \in A(x)} E\big[R'(x,a) + \gamma \tilde{V}\big(f(x,a), r\big)\big]$$

in state x. The functional form of $\tilde{V}$ is selected such that the evaluation of $\tilde{V}(x, r)$ is simple once the vector r is determined. A scoring function with a small number of parameters can thus compactly represent a large state space. For example, $\tilde{V}(x, r)$ may be the output of some neural network in response to the input x, and r is the associated vector of weights or parameters of the neural network. Alternatively, features or basis functions can be selected to represent states, in which case r is the associated vector of relative weights of the features or basis functions. Once the architecture of scoring functions is selected, the main computational burden involves "learning" the parameter vector r that most closely approximates the optimal value. The success of the approach depends heavily on the choice of a good architecture, which is generally problem dependent. Furthermore, the quality of the approximation is often difficult to gauge in terms of useful theoretical error bounds.

Up to now, the majority of the solution methods have concentrated on reducing the size of the state space to address the state space "curse of dimensionality." The key idea throughout is to avoid enumerating the entire state space. However, most of the above approaches generally require the ability to search the entire action space in order to choose the best action at each step of the iteration procedure; thus problems with very large action spaces may still pose a computational challenge. The approach proposed in Chap. 3 is meant to complement these highly successful techniques. In particular, there we focus on MDPs where the state space is relatively small but the action space is very large, so that enumerating the entire action space becomes practically inefficient. From a more general point of view, if one of the aforementioned state space reduction techniques is considered, for instance, state aggregation, then MDPs with small state spaces and large action spaces can also be regarded as the outcomes resulting from the aggregation of MDPs with large state and action spaces.

1.5 Simulation

In this book, simulation will mean *stochastic* (or Monte Carlo) simulation, as opposed to numerical approximations of (deterministic) differential equations, e.g., by

the Runge–Kutta method. Specifically, simulation is used to generate realizations of the system dynamics in the MDP simulation model described by (1.4). The context that we most frequently have in mind is where f is not known explicitly but for which the output of f can be easily generated, given the state, action, and input random number. For example, in a capacity planning model in manufacturing, the transitions and cost/rewards in the MDP model might correspond to outputs from a run of a large simulation model of a complex semiconductor fabrication facility, the action might be a choice of whether or not to add long-term capacity by purchasing an expensive new piece of machinery, the current state is the existing capacity and other relevant system information, and the input "random number" could represent a starting seed for the simulation model. Here, we outline some important basic aspects connected with performing such simulations, but because this is not the focus of the work in this book, the discussion will be brief. Specifically, we touch upon the following:

- random number generation;
- random variate generation;
- input analysis;
- output analysis;
- verification and validation;
- variance reduction techniques.

The fundamental inputs driving the stochastics in Monte Carlo simulation are random number streams. A random number stream is by definition a sequence of i.i.d. $U(0, 1)$ random variables, the realizations of which are called random "variates" in simulation terminology. An algorithm or procedure to generate such a sequence is usually called a pseudo-random number generator, and sometimes the resulting output may also retain the "pseudo-" prefix (viz., pseudo-random number). Most of the older common pseudo-random number generators are linear congruential generators (LCGs) based on the iteration:

$$x_n = (ax_{n-1} + c) \pmod{m}, \quad n = 1, 2, \ldots,$$

where m is the modulus (an integer), a is the multiplier, and c is the increment (the latter two both integers between 1 and $m - 1$). The starting point x_0 is called the seed. A prime modulus multiplicative linear congruential generator takes $c = 0$ and m prime. Clearly, one can iterate the recurrence to obtain

$$x_n = \left[a^n x_0 + \frac{c(a^n - 1)}{a - 1} \right] \pmod{m}, \quad n = 1, 2, \ldots,$$

so that any x_n can be found in a deterministic manner just from the values of x_0, m, a, and c. The random numbers are then generated from the sequence of $\{x_n\}$ via

$$u_n = x_n/m. \tag{1.26}$$

Commercial random number generators improve upon the basic LCGs by employing more complicated forms of the recursion. A multiple recursive generator (MRG) of order k is based on the following kth-order linear recurrence:

$$x_n = (a_1 x_{n-1} + \cdots + a_k x_{n-k}) \mod m, \tag{1.27}$$

where m and k are positive integers, a_i are integers of $0, 1, \ldots, m-1$, and again the actual random number sequence is generated via (1.26). In order to obtain generators with large periods in an efficient manner, instead of using (1.27) directly with a single large modulus, one constructs an equivalent generator by combining smaller modulus MRGs based on (1.27).

An alternative to pseudo-random numbers are quasi-Monte Carlo sequences (also known as low-discrepancy sequences), which do not attempt to preserve the independence between members of the sequence, but rather try to spread the numbers out so as to most uniformly cover the $[0, 1]^d$ hypercube, for a d-dimensional problem. Examples of such sequences include Faure, Halton, Sobol, Hammersley, and Niederreiter. These sequences lead to a deterministic $O((\log N)^d/N)$ error bound for numerical integration, as opposed to the usual $O(1/\sqrt{N})$ convergence rate associated with Monte Carlo integration, where N is the number of points sampled.

The form of the system dynamics in the MDP simulation model described by (1.4) masks two fundamental steps in carrying out the mechanics of stochastic simulation. The first is the transformation from random number sequences to input stochastic processes. The second is the transformation from input stochastic processes to output stochastic processes, which leads to the state transformation implied by (1.4).

The basic methodology for generating input processes usually involves an algorithm for going from a random number to a random variate, given a target probability distribution, which may be continuous or discrete. For example, to generate sample paths associated with Brownian motion, Gaussian random variates need to be generated. If the input process involves dependencies, this is an additional step that must be included. Random variate generation is done through a number of means, primarily consisting of some combination of the following:

- Inverse Transform Method, which uses the c.d.f.;
- Acceptance–Rejection Method, which uses the p.d.f.;
- Composition Method, which takes a convex combination of distribution and uses one of the two procedures above;
- Convolution Method, which takes the sum of r.v.'s and uses one of the first two procedures above;
- specialized routines for a given distribution (e.g., normal/Gaussian).

The transformation from input processes to output processes usually constitutes the bulk of a simulation model, in terms of implementation. For example, a semiconductor fabrication facility simulation model is commonly based on a discrete-event dynamic system model, which involves the mechanics of next-event scheduling. In terms of model building, two fundamental aspects in implementing a simulation

model are *verification*, which is to make sure that the model is working as desired (e.g., debugging the program properly), and *validation*, which is to make sure that the model represents the real system closely enough to make it useful for the target decision making or modeling goals. These two issues are quite different, but both are critical.

Input analysis and *output analysis* refer to the use of statistical inference on data. Input analysis takes actual "real-world" data to build the probability distributions that drive the input processes to the simulation model. Output analysis takes output data from the simulation model (i.e., simulated data) in order to make meaningful statistical statements, generally in the form of point estimation and interval estimation with confidence intervals. A key element of the Monte Carlo method is the availability of confidence intervals, which provide a measure of precision for the estimators of simulation output.

Because simulation can be quite expensive in terms of computational cost, an important aspect has to do with efficiency of the estimation in the output analysis. Methodologies for improving this aspect are called variance reduction techniques or efficiency improvement techniques, and can lead to orders of magnitude reduction in computation. Among the most effective of these are the following:

- control variates—exploiting correlation between simulation processes with known distributional properties (usually the mean) and the target output performance measure;
- importance sampling ("change of measure")—changing the parameters (e.g., mean) of input distributions with an appropriate reweighting of the target output performance measure;
- stratified sampling—dividing the sampling procedure into subsets such that each has much reduced variability in the target output performance measure, and carrying out conditional sampling on the subsets;
- conditional Monte Carlo—conditioning on certain processes in the simulation to derive a conditional expectation estimator of the target output performance measure;
- common random numbers—exploiting positive correlation to reduce variance when comparing different systems or the same system at different parameter settings (e.g., an MDP sample path using different actions from the same state).

Variance reduction techniques such as these can dramatically improve the performance of simulation-based algorithms for solving MDPs, but this is an area on which there has been scant research, so there is clearly untapped potential for progress on this front.

1.6 Preview of Coming Attractions

Table 1.1 provides a summary of the various settings considered, based on various characteristics of the MDP model. The term "analytical" means that f or P is

Table 1.1 Taxonomy of problem settings and solution approaches

	Chapter			
	2	3	4	5
finite horizon	√		4.3, 4.5, 4.6	
infinite horizon		√	4.3, 4.4,	√
simulation-based	√		4.3, 4.5, 4.6	√
analytical		√	4.4	
sampling	√		√	√
population		√	4.4, 4.6	5.4
large state spaces	√		4.3, 4.5	√
large action spaces		√	√	√

known explicitly, and the resulting optimality (or policy evaluation) equations will be solved directly. As described in the previous section, the term "simulation" will indicate realized states and/or rewards resulting in a "sample path" of length H for the finite-horizon setting. On the other hand, "sampling" will be reserved to indicate a means by which the next action or policy is chosen to be simulated. Chaps. 2, 4, and 5 all contain simulation-based sampling algorithms (Sect. 3.4 also includes a brief discussion of simulation-based algorithms), which become the method of choice in settings where

(i) either the transition function/probabilities are *not explicitly known* or it is computationally infeasible to use them, due to the size of the state space, or
(ii) the one-stage reward is stochastic with its distribution *not explicitly known*.

For example, in many complex systems, it is often the case that a simulation model is available that is essentially a black box that captures detailed stochastic interactions in the system, e.g., the semiconductor fabrication facility simulation model described earlier. In this setting, a state-action pair produces a simulated visited state or one-stage reward, or both in the case where both assumptions hold. An underlying implicit assumption is that the cost of simulation is relatively expensive in terms of computational burden.

1.7 Notes

Texts on Markov decision processes include [12, 145], and [114], in which the standard results summarized here can be found. More advanced treatments, including rigorous discussion of MDPs with uncountable (e.g., Borel) state spaces and unbounded rewards, can be found in [16, 82] and [85]; see also [61]. For the relationship between the simulation model and the standard MDP model, see [23] or [85, Sect. 2.3]. For a recent summary of analysis and solution methods for finite state and action MDPs, see [102]. It can be shown that policy iteration converges faster to the optimal value than value iteration in terms of the number of iterations if

both algorithms begin with the same value [145], and policy iteration often outperforms value iteration in practical applications [22, 101]. In particular, for small-scale problems (state space size less than 10,000), policy iteration performs considerably better than value iteration, provided the discount factor is close to 1 [153]. See [123] or [22] for a detailed discussion of the complexity of the two approaches, including the state and action space-dependent time complexity of the linear programming approach for solving MDPs. For a discussion of conditions under which there exists a stationary optimal policy for infinite-horizon MDPs, see [3, 24, 85].

The geometric convergence of the rolling-horizon control to the optimal value can be found in [84]. Existence of a minimal finite horizon H^* such that the rolling-H^*-horizon control prescribes exactly the same action as the policy that achieves the optimal infinite-horizon rewards at every state can be found in [18] for the discounted case and [83] for the average case.

The idea of rolling-horizon control has been applied to many interesting problems in various contexts to solve the problems in an on-line manner, including planning problems (e.g., inventory control) that can be modeled as linear programs [76] and that can be represented as a shortest path problem in an acyclic network (see [60] for example problems and references therein), routing problems in communication networks by formulating the problem as a non-linear optimal control problem [5], dynamic games [178], aircraft tracking [139], the stabilization of non-linear time-varying systems [105, 129, 130] in the model predictive control literature, and macroplanning in economics [100]. For a survey relating rolling-horizon control, approximate dynamic programming, and other suboptimal control methods, see [13], where the former is referred to as receding-horizon control; for a bibliography of applications in operations management problems, see [29].

One of the earliest works employing randomization to break the curse of dimensionality used random successive approximations and random multigrid algorithms [154]. Classical references on reinforcement learning are [101, 171]. Recent work on approximate dynamic programming and simulation-based methods includes [75, 99, 142, 164]. Approximate dynamic programming has come to mean mainly value function approximation, with the term neuro-dynamic programming coined by [17], because neural networks represent one of the most commonly used approaches for representing the value function or Q-function.

Q-learning was introduced by Watkins [180]; see also [17, 177]. Some results exist on the convergence rate of Q-learning are found in [57]. For a recent survey on research in neuro-dynamic programming, see [179].

Representative examples on the use of structural properties include [141] and [166] for general approaches; [68, 160, 170], [145, Sect. 4.7], and [62] for monotonicity; [24] for convexity; [2, 181], and [107, Chap. 5] for modularity; [159] for approximating sequences; and [110] for factored representations. Work on approximating the value function includes [71] and [14] via state aggregation, [52] on using basis functions with a linear programming approach, and [17] on feature extraction.

In parameterized policy space, a simulation-based method for solving average-cost MDPs by iteratively estimating the performance gradient of a policy and updating the policy parameters in a direction of improvement is proposed in [127].

Drawbacks of the approach include potentially large variance of the gradient estimator and the discarding of past gradient information. Additional related work includes [128] and [185]. Actor-critic algorithms [9] use an approximation architecture to learn a value function via simulation, and the value function is used to update the policy parameters in a direction of performance improvement. Work employing importance sampling in actor-critic algorithms includes [186]. A convergence proof of some actor-critic algorithms under linearly parameterized approximations of the value function for average-cost MDPs is provided in [111], but theoretical understanding has been limited to the case of lookup table representations of policies and value functions.

Another approach for solving average-reward MDPs is simulation-based policy iteration, which employs a simulation for policy evaluation at each iteration and applies policy improvement with the approximate solutions to the average evaluation equations. In [48], three simulation estimators are analyzed for policy evaluation, and conditions derived on the simulation runlengths that guarantee almost-sure convergence of the algorithm. Chang [37] presents a simulation-based algorithm for average MDPs based on the work by Garcia et al. [28, 70] of a *decentralized* approach to discrete optimization via the "fictitious play" algorithm applied to games with identical payoffs. A given MDP is basically formulated as an identical payoff game where a player is associated with each state and each player plays selecting an action in his action set with the goal of minimizing the identical payoff, which is the average cost of following the policy constructed from each player's action selection. This identical payoff game is iteratively solved with a simulation-based variant of fictitious play in an off-line manner to find a pure Nash-equilibrium. If there exists a unique optimal policy, the sequence of probability distributions over the policy space generated by the algorithm converges to a distribution concentrated only on the unique optimal policy with probability one.

On-line estimation of the "performance potential" of a policy by a single sample-path simulation combined with gradient-based stochastic approximation simulation-based policy iteration algorithm is presented in [59]. A "temporal-difference" learning for evaluating a policy in a similar context to simulation-based policy iteration can be found in [80].

Some related models with MDPs have been studied by White and Eldeib [184], and Satia and Lave [156], under the rubric of MDPs with "imprecisely known transition probabilities," and Givan et al. [71] under "bounded parameter Markov Decision Processes." All of these models can be viewed within the framework of "controlled Markov set-chain" by Kurano et al. [115], even though the notion of "Pareto-optimality" defined by Kurano et al. was not dealt with in any of these efforts. Chang [36] develops a VI-type algorithm for solving controlled Markov set-chains and analyze its finite-step error bounds and also develops PI-type algorithms in [38] and establish their convergence. See [136] for various types of uncertainty model for transition probability distributions, including the "entropy" model and the interval model of Kurano et al., and related computational algorithms. Kalyanasundaram et al. [103] study continuous-time MDPs with unknown transition rates and average reward criteria, and develop a PI-type algorithm based on single-policy improvement, for obtaining robust ("max-min") policies.

The material on stochastic simulation in this chapter merely touches upon some basic ideas. Two standard texts are [63] and [120]; see also [64] for a more recent textbook. Another classical but more eclectic text is [25]. An excellent state-of-the-art reference to current simulation research is [81]; see also [7]. Recent research advances in stochastic simulation research are reported at the annual Winter Simulation Conference, whose proceedings are freely available online at http://www.informs-cs.org/wscpapers.html. A classic on random variate generation is [54], which is available online for free download at http://luc.devroye.org/rnbookindex.html, and a well-known reference on quasi-Monte Carlo is [135]; see also http://www.mcqmc.org/.

Chapter 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 non-stationary 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 Chap. 5.

Letting $\hat{V}_i^{N_i}(x)$ denote the estimate of the optimal reward-to-go function, $V_i^*(x)$, defined by Eq. (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 Eq. (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 turn determines the estimate $\hat{V}_i^{N_i}(x)$, albeit not necessarily using Eq. (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.

H.S. Chang et al., *Simulation-Based Algorithms for Markov Decision Processes*, Communications and Control Engineering,
DOI 10.1007/978-1-4471-5022-0_2, © Springer-Verlag London 2013

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

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:

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

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 $\{w_j^a,\ j = 1, \ldots, N_a^i(x)\}$ 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 $N = \max_{i=0,\ldots,H-1} N_i$, which is independent of the state space size. One approach is to select "optimal" values of $N_a^i(x)$ for $i = 0, \ldots, H-1$, $a \in A(x)$, 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 Sect. 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 $i = 0, \ldots, H-1$, $a \in A(x)$, and $x \in X$ such that the expected difference is bounded as a function of $N_a^i(x)$ and N_i, $i = 0, \ldots, H-1$, and such that the bound (from above and from below) goes to zero as N_i, $i = 0, \ldots, H-1$, 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., $E[\hat{V}_0^{N_0}(x)] \to V_0^*(x)$ as $N_i \to \infty, \forall\ i = 0, \ldots, H-1$), 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 $O((|A| \max_{i=0,\ldots,H-1} N_i)^H)$, 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 $1 \le i \le M$ and $j \ge 1$, where successive plays of machine i yield "rewards" $\eta_{i,1}, \eta_{i,2}, \ldots$, 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

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

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 Sect. 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 Fig. 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

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

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

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

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

  where

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

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

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

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

Output:

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

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

then iterate as follows (see **Loop** in Fig. 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

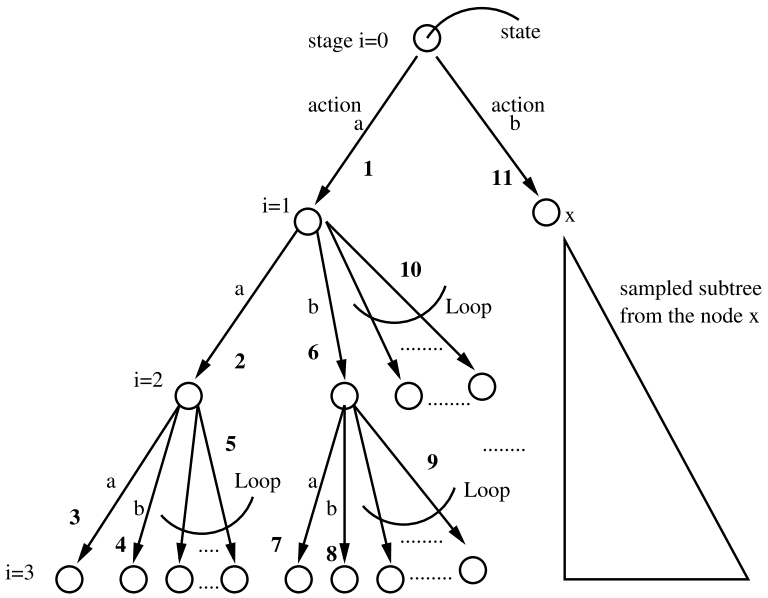


Fig. 2.3 Graphical illustration of a sequence of recursive calls made in **Initialization** of the UCB sampling algorithm, where each *circle* corresponds to a simulated state, each *arrow* with associated action signifies a sampling for the action (and a recursive call), and the *boldface number near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

sample mean of the immediate reward plus V_1^*-values (multiplied by the discount factor) at all of the simulated next states (cf. Eq. (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., $\{N_a^0(x)/N_0\}_{a \in A(x)}$ 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. Eq. (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 Eq. (2.5) in the UCB sampling algorithm. First, consider the estimator that replaces the weighted sum of the Q-function estimates in Eq. (2.5) by the maximum of the estimates, i.e., for $i < H$,

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

For the non-adaptive 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:

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

which would again replace Eq. (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 Eq. (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 non-adaptive versions of these estimators, but proving these results in our setting and characterizing the convergence rate of the estimator given by Eq. (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

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

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

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 $w^{\hat{a}^*}_{T_{\hat{a}}^x(\bar{n})+1} \sim U(0,1)$ 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

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

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

the algorithm is asymptotically unbiased, and the bias can be shown to be bounded by a quantity that converges to zero at rate $O(\sum_{i=0}^{H-1} \frac{\ln N_i}{N_i})$.

We start with a convergence result for the one-stage approximation. Consider the following one-stage sampling algorithm (OSA) in Fig. 2.4 with a *stochastic value function* U defined over X, where $U(x)$ for $x \in X$ is a *non-negative 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

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

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

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

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.

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

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

where

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

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

$$\begin{aligned}
T_a^x(n) &= 1 + \sum_{t=|A(x)|+1}^{n} I\{M_t = a\} \\
&\le \ell + \sum_{t=|A(x)|+1}^{n} I\big\{M_t = a,\ T_a^x(t-1) \ge \ell\big\} \\
&\le \ell + \sum_{t=|A(x)|+1}^{n} I\big\{\tilde{Q}_{T_{a^*}^x(t-1)}\big(x,a^*\big) + c_{t-1,T_{a^*}^x(t-1)} \\
&\qquad\qquad \le \tilde{Q}_{T_a^x(t-1)}(x,a) + c_{t-1,T_a^x(t-1)}, T_a^x(t-1) \ge \ell\big\}
\end{aligned}$$

$$\leq \ell + \sum_{t=|A(x)|+1}^{n} I\Big\{ \min_{0<s<t} \big(\tilde{Q}_s(x,a^*) + c_{t-1,s}\big) \leq \max_{\ell \leq s_a < t} \big(\tilde{Q}_{s_a}(x,a) + c_{t-1,s_a}\big)\Big\}$$

$$\leq \ell + \sum_{t=1}^{n}\sum_{s=1}^{t-1}\sum_{s_a=\ell}^{t-1} I\big\{\tilde{Q}_s(x,a^*) + c_{t,s} \leq \tilde{Q}_{s_a}(x,a) + c_{t,s_a}\big\}. \tag{2.10}$$

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

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

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

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

By using Hoeffding's inequality [86] we can bound the probability of events (2.11) and (2.12):

$$P\big(\tilde{Q}_s(x,a^*) \leq V(x) - c_{t,s}\big) \leq e^{-4\ln t} = t^{-4},$$
$$P\big(\tilde{Q}_{s_a}(x,a) \geq Q(x,a) + c_{t,s_a}\big) \leq e^{-4\ln t} = t^{-4}.$$

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

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

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

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

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

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

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

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

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

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*

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

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

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

Applying Theorem 2.1, we have

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

$$\leq \frac{C_1 \ln n}{n} + \frac{C_2}{n}, \tag{2.16}$$

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 Eq. (2.9), it follows that

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

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,

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

by applying a result analogous to Wald's equation.

Since

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

the convergence follows directly from Eq. (2.17).

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

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

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* $i = 0, \ldots, H-1$, *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* $O(\sum_{i=0}^{H-1} \frac{\ln N_i}{N_i})$, *i.e.*,

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

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

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

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

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

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

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

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

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

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

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

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

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

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 $O(\sum_{i=0}^{H-1} \frac{\ln N_i}{N_i})$. Define $\Psi_i \in B(X)$ such that $\Psi_i(x) = E[\hat{V}_i^{N_i}(x)]$ for all $x \in X$ and $i = 0, \ldots, H-1$ and $\Psi_H(x) = V_H^*(x) = 0,\ x \in X$. In the proof of Lemma 2.2 (see Eq. (2.17)), we showed that for $i = 0, \ldots, H-1$,

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

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

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

and

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

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

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

Therefore, combining (2.18) and (2.20) yields

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

Repeating this argument yields

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

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

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

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

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 (non-stationary) 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

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

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), 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

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

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

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

- Sample $w_{N_a^i(x)+1}^{\hat{a}} \sim U(0,1)$ 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

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

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

Output:

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

Fig. 2.5 Modified UCB algorithm for minimization problems

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 Fig. 2.5.

The alternative estimators would then be obtained by replacing the final estimator given by Eq. (2.23) in Fig. 2.5 by the following, corresponding to Eqs. (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

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

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

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

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 non-stationary, whereas in case (ii), the optimal policy is of the (s, S) type, again non-stationary. 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

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
$K = 0$	10.440	4	15.03 (0.29)	9.13 (0.21)	9.56 (0.32)
$p = 1$	$s = 0$	8	12.82 (0.16)	10.21 (0.10)	10.30 (0.10)
		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)
$K = 0$	24.745	4	30.45 (0.87)	19.98 (0.79)	20.48 (0.82)
$p = 10$	$s = 6$	8	28.84 (0.49)	23.09 (0.55)	23.68 (0.52)
		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)
$K = 5$	10.490	4	18.45 (0.29)	10.23 (0.21)	10.41 (0.22)
$p = 1$	$s_1 = 0$	8	14.45 (0.15)	10.59 (0.10)	10.62 (0.10)
	$s_2 = 0$	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)
$K = 5$	31.635	4	37.52 (0.98)	26.42 (0.88)	26.92 (0.89)
$p = 10$	$s_1 = 6$	8	36.17 (0.43)	30.13 (0.49)	30.41 (0.51)
	$s_2 = 6$	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)

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 Eq. (2.23), and estimators 2 and 3 refer to the estimators using Eqs. (2.24) and (2.25) with $a^* \in \arg\max_a \{N_a^i(x)\}$ in place of Eq. (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 Eq. (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.

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
$K = 0$	7.500	21	24.06 (0.16)	3.12 (0.17)	9.79 (0.21)
$p = 1$	$S = 4$	25	22.05 (0.12)	5.06 (0.12)	6.28 (0.19)
		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)
$K = 0$	13.500	21	29.17 (0.21)	6.04 (0.30)	13.69 (0.46)
$p = 10$	$S = 9$	25	28.08 (0.21)	9.28 (0.23)	12.06 (0.29)
		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)
$K = 5$	10.490	21	33.05 (0.12)	8.73 (0.21)	18.62 (0.44)
$p = 1$	$s_1 = 0, S_1 = 0$	25	29.99 (0.10)	10.96 (0.11)	11.79 (0.16)
	$s_2 = 0, S_2 = 0$	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)
$K = 5$	25.785	21	39.97 (0.22)	17.78 (0.49)	26.76 (0.52)
$p = 10$	$s_1 = 6, S_1 = 9$	25	39.01 (0.19)	22.68 (0.26)	25.09 (0.33)
	$s_2 = 6, S_2 = 9$	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)

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).

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)

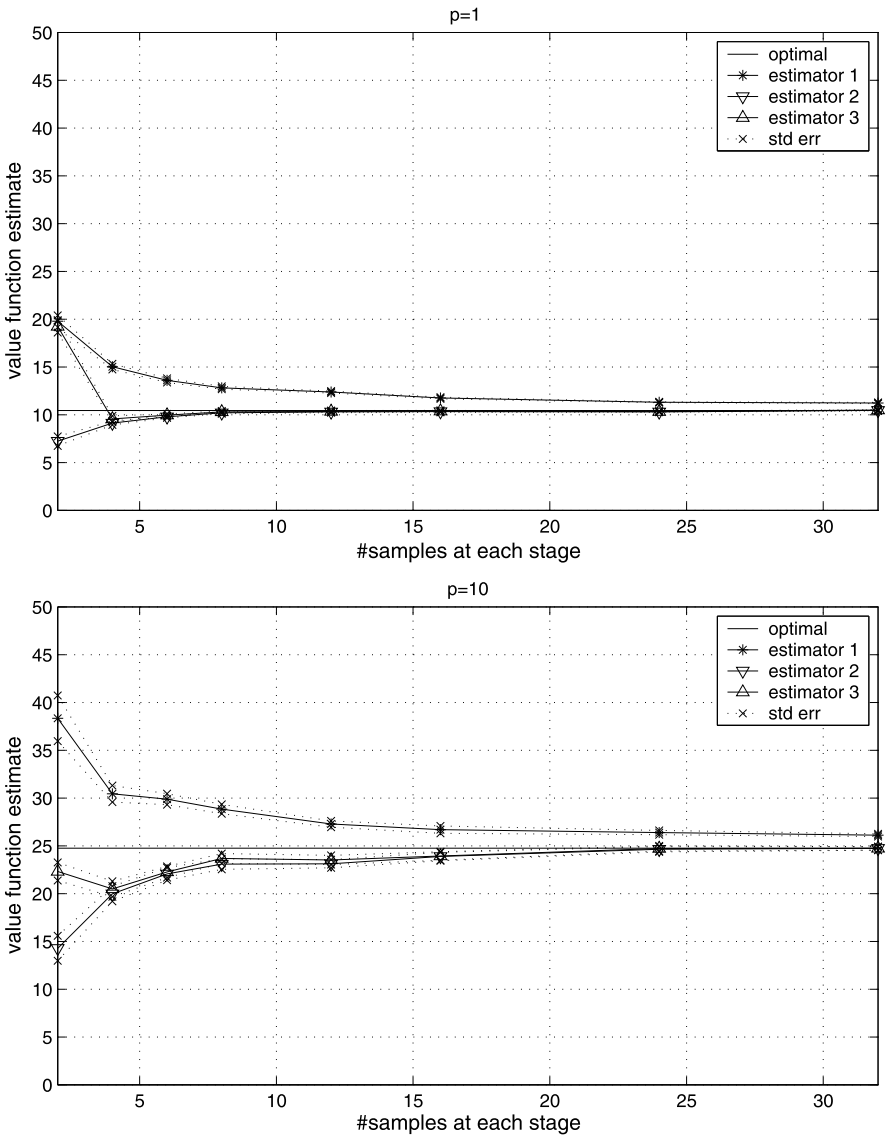


Fig. 2.6 Convergence of value function estimate for the inventory control example case (i) $q = 10$ as a function of the number of samples at each state: $H = 3, M = 20, x_0 = 5, D_t \sim DU(0, 9)$, $h = 1, K = 0$

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

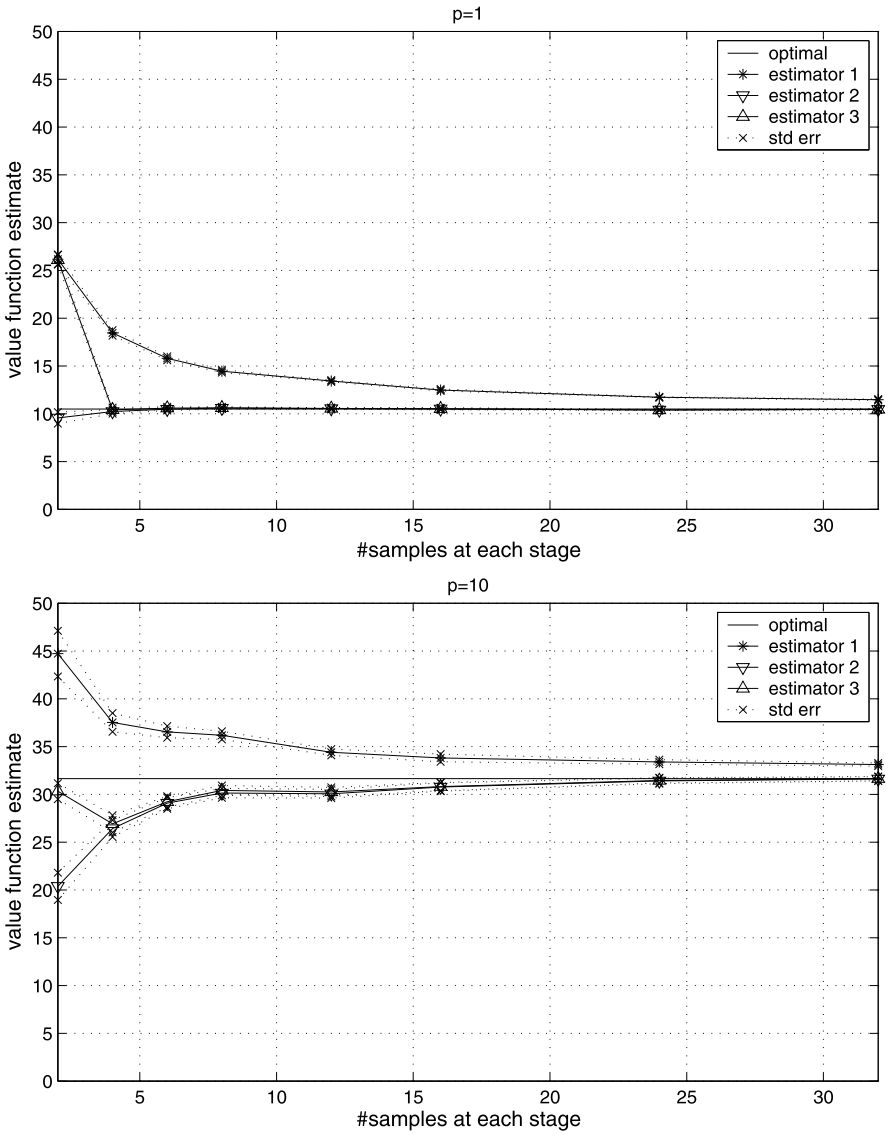


Fig. 2.7 Convergence of value function estimate for the inventory control example case (i) $q = 10$ as a function of the number of samples at each state: $H = 3, M = 20, x_0 = 5, D_t \sim DU(0, 9)$, $h = 1, K = 5$

(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

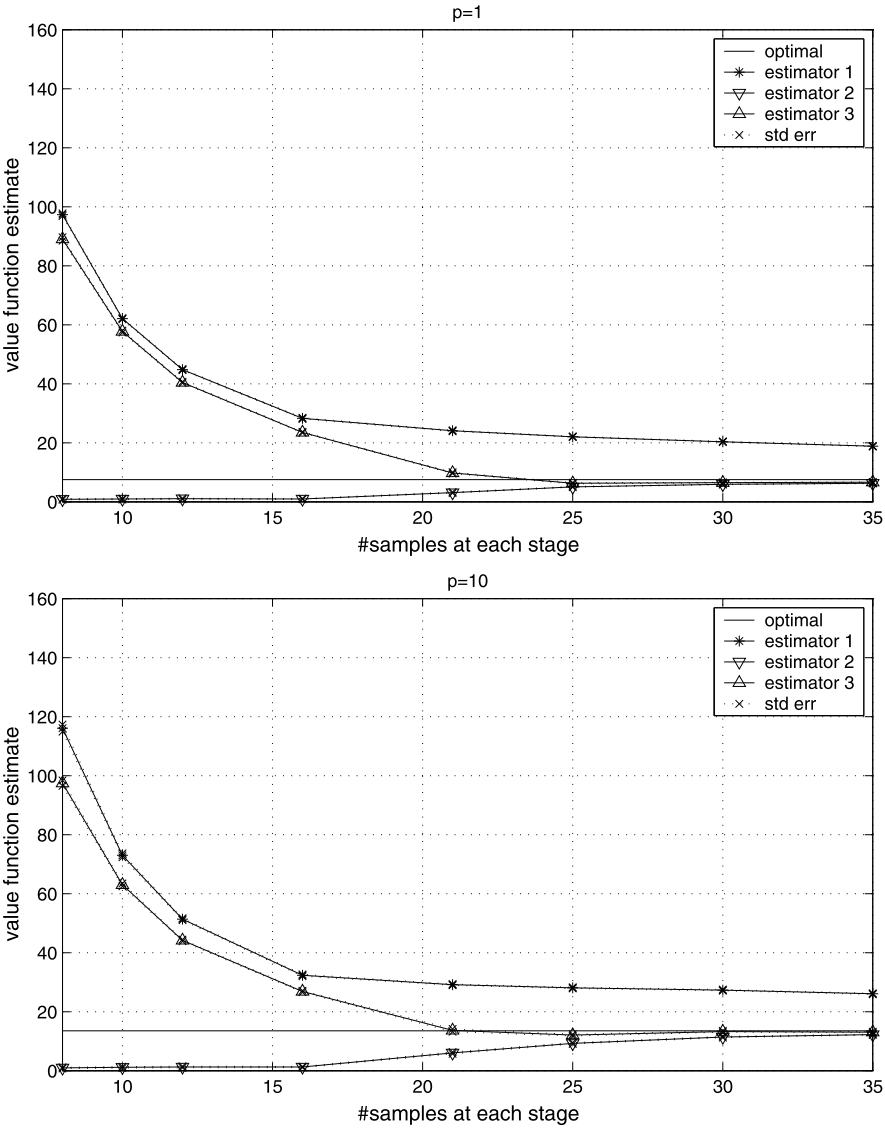


Fig. 2.8 Convergence of value function estimate for the inventory control example case (ii) as a function of the number of samples at each state: $H = 3, M = 20, x_0 = 5, D_t \sim DU(0, 9), h = 1, K = 0$

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

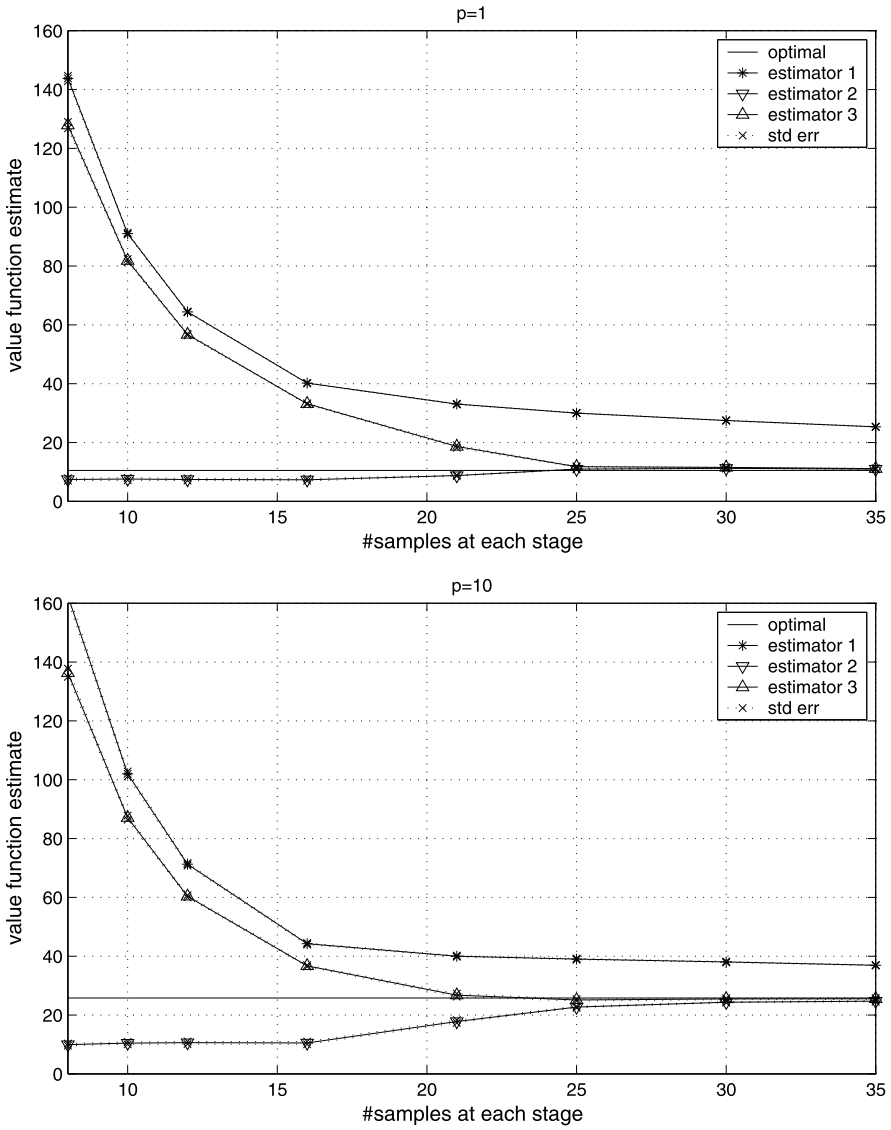


Fig. 2.9 Convergence of value function estimate for the inventory control example case (ii) as a function of the number of samples at each state: $H = 3, M = 20, x_0 = 5, D_t \sim DU(0, 9)$, $h = 1, K = 5$

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. Eq. (2.29)), somewhat analogous to the UCB algorithm alternative estimator given by Eq. (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 non-recursive one-stage original version. After N_0 iterations, the algorithm estimates the optimal value $V_0^*(x_0)$ by the Q-function

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:

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

$$N_{a(k)}^i(x) \leftarrow N_{a(k)}^i(x) + 1,$$

$$\hat{Q}_i^{N_i}\big(x, a(k)\big) \leftarrow \frac{M_i^{N_i}(x, a(k))}{N_{a(k)}^i(x)}.$$

- Update optimal action estimate: (ties broken arbitrarily)

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

- Update probability distribution over action space:

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

- $k \leftarrow k + 1$.

Output:

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

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

value at the currently estimated optimal action via Eq. (2.29), where

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

$\sum_{a \in A(x_0)} N_a^0(x_0) = N_0$. 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 $\{w_j^a\}$, $a \in A(x)$.

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).

Lemma 2.4 (Cf. [146, Lemma 3.1]) *Given $\delta \in (0,1)$ and positive integer K such that $6 \le K < \infty$, consider running the one-stage non-recursive PLA sampling algorithm obtained by replacing (2.26) by*

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

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

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

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

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

Theorem 2.5 *Let $\{X_i,\ i = 1,2,\ldots\}$ be a sequence of i.i.d. non-negative uniformly bounded random variables, with $0 \le X_i \le D$ 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*

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

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

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

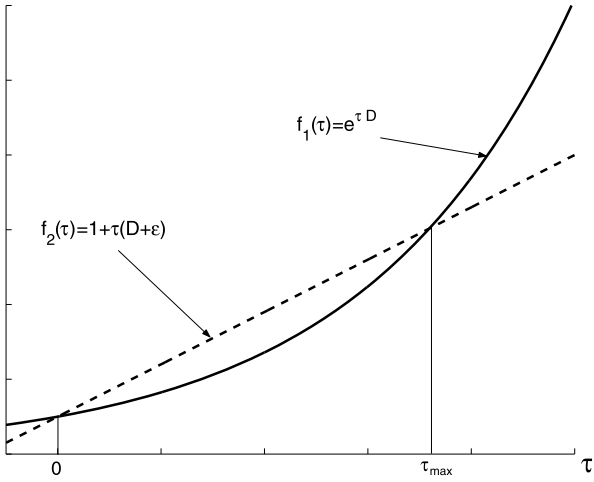


Fig. 2.11 Sketch of functions $f_1(\tau) = e^{\tau D}$ and $f_2(\tau) = 1 + \tau(D + \epsilon)$

where

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

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

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

which implies that

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

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

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

It can be shown that (cf. Lemma 1 in [163, 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

$$\begin{aligned}
&P\left(\frac{1}{M}\sum_{i=1}^{M} X_i - \mu \geq \epsilon, M \geq n\right) \\
&\leq P\left(e^{\tau Y_M - \Lambda_D(\tau)\langle Y\rangle_M} \geq e^{\tau n\epsilon - n\Lambda_D(\tau)D^2}\right) \\
&\leq P\left(\sup_{0\leq t\leq L} Z_t(\tau) \geq e^{\tau n\epsilon - n\Lambda_D(\tau)D^2}\right) \\
&\leq \frac{E[Z_0(\tau)]}{e^{\tau n\epsilon - n\Lambda_D(\tau)D^2}} \quad \text{by maximal inequality for supermartingales [163]} \\
&= e^{-n(\tau\epsilon - \Lambda_D(\tau)D^2)}. \qquad (2.31)
\end{aligned}$$

By using a similar argument, we can also show that

$$P\left(\frac{1}{M}\sum_{i=1}^{M} X_i - \mu \leq -\epsilon, M \geq n\right) \leq e^{-n(\tau\epsilon - \Lambda_D(\tau)D^2)}. \qquad (2.32)$$

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

$$P\left(\left|\frac{1}{M}\sum_{i=1}^{M} X_i - \mu\right| \geq \epsilon, M \geq n\right) \leq 2e^{-n(\tau\epsilon - \Lambda_D(\tau)D^2)}. \qquad (2.33)$$

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 non-recursive PLA sampling algorithm obtained by replacing* (2.26) *by*

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

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

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

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*,

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

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))

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

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

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

Define the events

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

By the law of total probability,

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

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

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

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

$$\begin{aligned} \lambda(\epsilon,\delta) &= \left\lceil \frac{2M_{\epsilon,\delta}}{\ln l} \ln\left[\frac{l M_{\epsilon,\delta}}{\ln l}\left(\frac{2M_{\epsilon,\delta}}{\delta}\right)^{1/M_{\epsilon,\delta}}\right]\right\rceil, \\ M_{\epsilon,\delta} &= \max\left\{6, \left\lceil \frac{R_{\max} H \ln(4/\delta)}{(R_{\max}H + \epsilon)\ln((R_{\max}H + \epsilon)/R_{\max}H) - \epsilon} \right\rceil\right\}, \end{aligned}$$

$$\mu_i^*(\epsilon,\delta) = 1 - 2^{-\frac{1}{N_i}} < 1 - 2^{-\frac{1}{\lambda(\epsilon,\delta)}}.$$

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 $i = 0, 1, \ldots, H-1$,

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

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

Define $\theta := \inf_{x \in X, i=0,\ldots,H-1} \theta_i(x)$. Given $\delta_i \in (0,1)$, $i = 0, \ldots, H-1$, 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 Eq. (2.35)) and* $0 < \mu_i < \mu_i^* = 1 - 2^{-\frac{1}{N_i}}$ *for a given* $\delta_i \in (0,1)$, $i = 0, \ldots, H-1$. *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 Eq. (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 $x \in X_s^i$,

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

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

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

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,

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

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

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

for $\epsilon > 0$, then

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

which further implies

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

and therefore

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

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 $\delta_i \in (0,1)$,

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

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 the result 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,

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

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 [146]). 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

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

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 $N_{H-1} > \lambda(\theta/2^{H+1}, \delta_{H-1}) \ge \lambda(\theta/2^H, \delta_{H-1})$ and $\mu_{H-1} \in (0, 1-2^{-1/N_{H-1}})$,

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

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 $\mu_{H-3} \in (0, 1-2^{-1/N_{H-3}})$,

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

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

$$P\left(\left|\hat{V}_1^{N_1}(x) - V_1^*(x)\right| > \frac{\theta}{2^2}\right)$$
$$< 1 - (1-\delta_1)(1-\delta_2)^{N_2}(1-\delta_3)^{N_2 N_3} \times \cdots \times (1-\delta_{H-1})^{N_2 \cdots N_{H-1}}$$

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* $\delta_i \in (0,1)$, $i = 0, \ldots, H-1$, *select* $N_i > \lambda(\theta/2^{i+2}, \delta_i)$ *and* $0 < \mu_i < \mu_i^* = 1 - 2^{-1/N_i}$, $i = 1, \ldots, H-1$. *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 Eq.* (2.38), *for all* $\epsilon \in (0,1)$,

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

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

Proof Define the event

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

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

$$P\left(E'(\kappa + K)\right) \geq P\left(E'(\kappa + K) \middle| E(K)\right) P\left(E(K)\right), \quad \kappa = 1, 2, \ldots,$$

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

By selecting $N_0 > K = \lambda(\frac{\theta}{4}, \delta_0)$ and $N_i > \lambda(\frac{\theta}{2^{i+2}}, \delta_i)$, $i = 1, \ldots, H-1$, 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* $\delta_i \in (0,1), i = 0, \ldots, H-1$ *and* $\epsilon \in (0, \theta]$, *select* $N_i > \lambda(\frac{\epsilon}{2^{i+2}}, \delta_i), 0 < \mu_i < \mu_i^* = 1 - 2^{-\frac{1}{N_i}}, i = 0, \ldots, H-1$. *Then under the PLA sampling algorithm with* ρ *in Eq.* (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 Theorems 2.8 and 2.9 are obtained with coarse estimation of various parameters/terms. For example, we used the worst-case values of $\theta_i(x), x \in X, i = 0, \ldots, H-1$ and $(R_{\max} H)^2$ for bounding $\sup_{x \in X} V_i^*(x), i = 0, \ldots, H-1$, 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.

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 $y \in X$,

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

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

$$M_i\big(x,a(k)\big) \leftarrow M_i\big(x,a(k)\big) + \sum_{z \in X} r\big(x,a(k),z\big) I_{i+1}^k(z) + \hat{V}_{i+1}^{N_{i+1}}\big(I_{i+1}^k\big),$$

$$N_{a(k)}^i(x) \leftarrow N_{a(k)}^i(x) + 1,$$

$$\hat{Q}_i^{N_i}\big(x,a(k)\big) \leftarrow \frac{M_i^{N_i}(x,a(k))}{N_{a(k)}^i(x)}.$$

- 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

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-

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)

state by

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

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 $h : \Psi \times [0, 1] \to X_I$, 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 $R_I : \Psi \times [0, 1] \to \Re^+$ 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. Fig. 2.10) applied to the unreduced POMDP model.

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

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)

in [104], using the inventory control problem of Sect. 2.1.5. The numerical results for UCB sampling are based on the alternative estimator 2 in Sect. 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 the 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 Sect. 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.

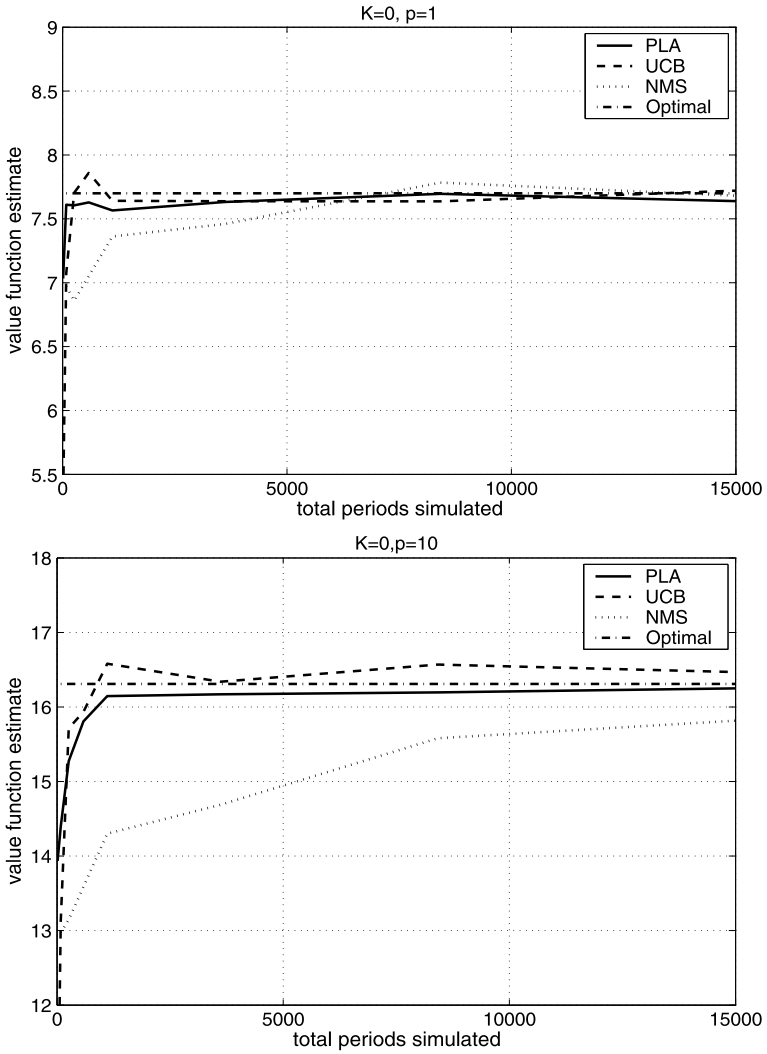


Fig. 2.13 Value function estimates (mean of 30 simulation replications) of the PLA, UCB, and NMS algorithms for the inventory control example case (i) as a function of the number of samples at each state: $H = 3$, $M = 20$, $x_0 = 5$, $D_t \sim DU(0, 9)$, $h = 1$, $K = 0$

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 the UCB estimates also show a significant reduction in the standard error over the NMS estimate.

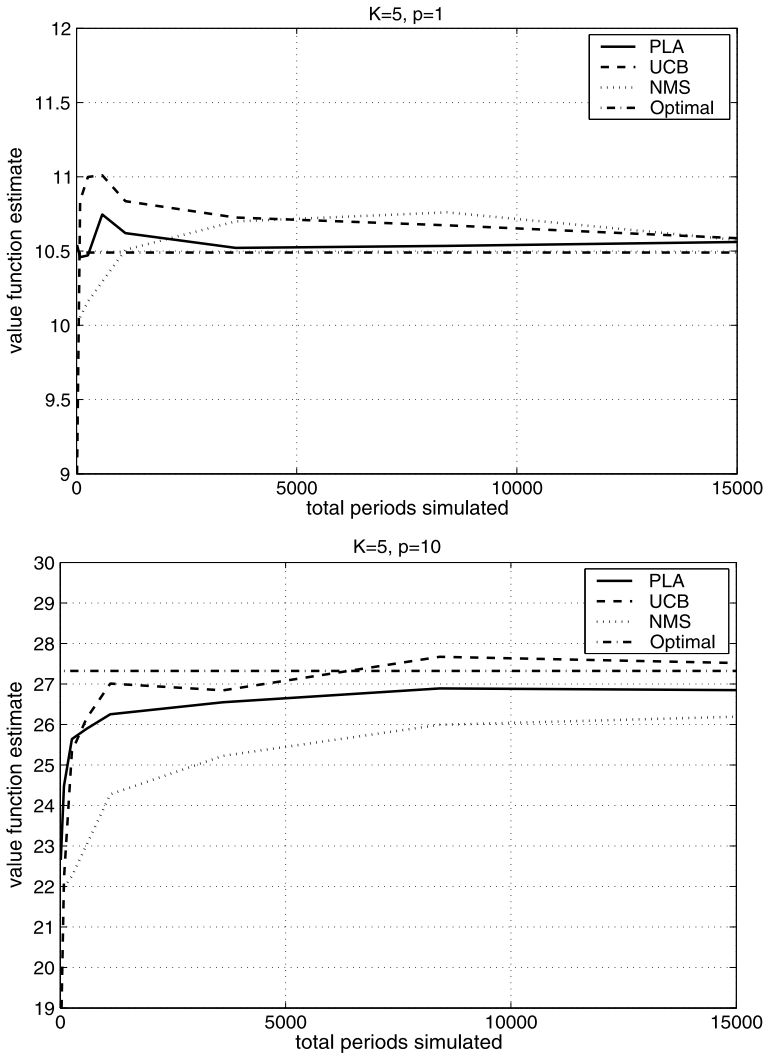


Fig. 2.14 Value function estimates (mean of 30 simulation replications) of the PLA, UCB, and NMS algorithms for the inventory control example case (i) as a function of the number of samples at each state: $H = 3$, $M = 20$, $x_0 = 5$, $D_t \sim DU(0, 9)$, $h = 1$, $K = 5$

2.3 Notes

The expected regret analysis for multi-armed bandit models motivating the UCB sampling algorithm goes back to [119] (see, also [27]). 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.

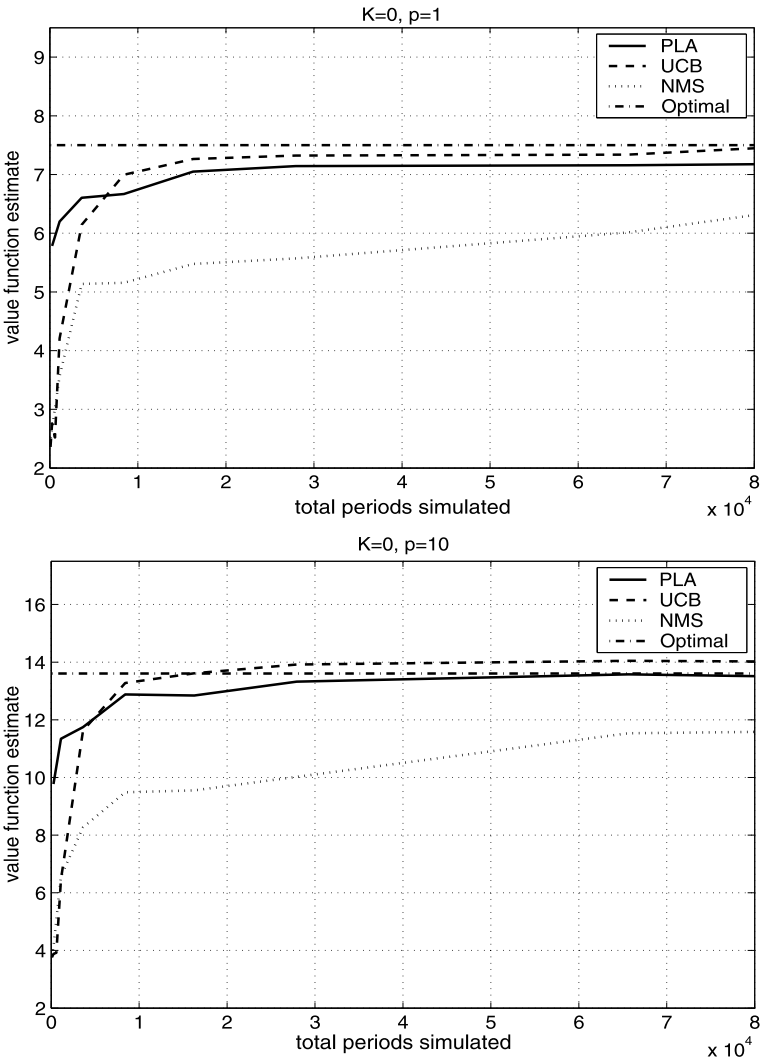


Fig. 2.15 Value function estimates (mean of 30 simulation replications) of the PLA, UCB, and NMS algorithms for the inventory control example case (ii) as a function of the number of samples at each state: $H = 3, M = 20, x_0 = 5, D_t \sim DU(0, 9), h = 1, K = 0$

The pursuit algorithm designed with learning automata that motivated the PLA sampling algorithm presented here for MDPs was introduced in [173] (see also [137] and [155]). The finite-time analysis of the pursuit algorithm is based on [146], 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 [133] and in the overview survey article [174], whereas application of learning automata for solving controlled (ergodic)

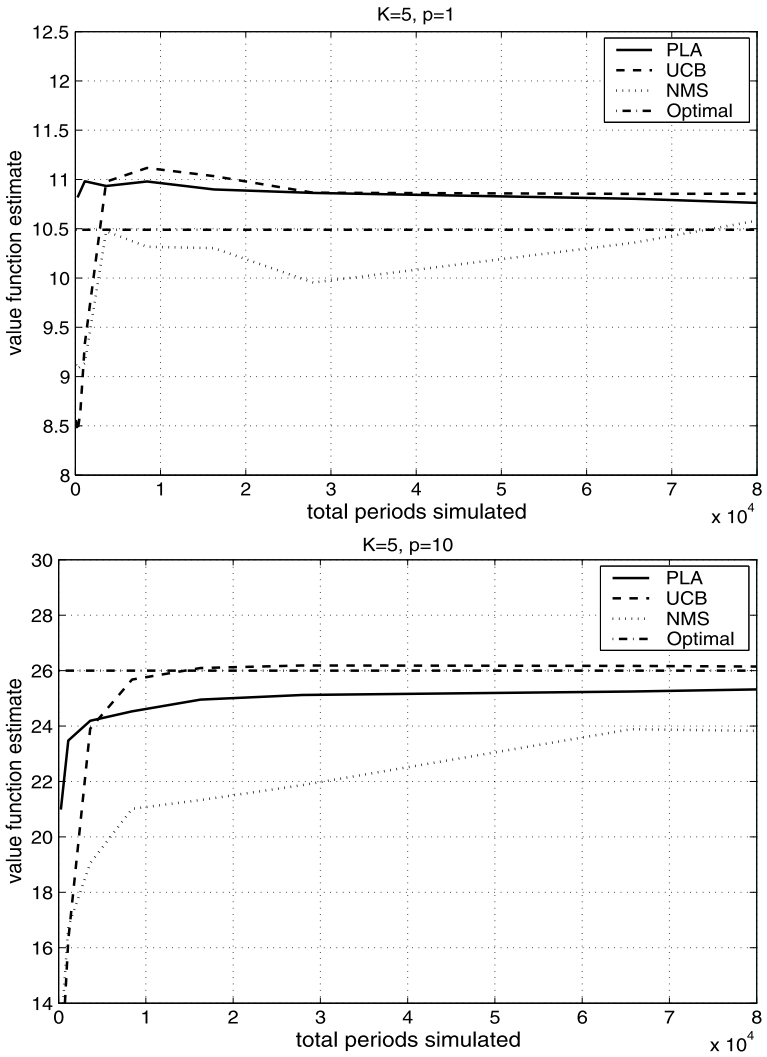


Fig. 2.16 Value function estimates (mean of 30 simulation replications) of the PLA, UCB, and NMS algorithms for the inventory control example case (ii) as a function of the number of samples at each state: $H = 3$, $M = 20$, $x_0 = 5$, $D_t \sim DU(0, 9)$, $h = 1$, $K = 5$

Markov chains in a model-free reinforcement learning (RL) framework for a loss function defined on the chains can be found in the books [143, 144]. Controlling ergodic Markov chains for the infinite-horizon average reward within a similar RL framework is considered in [183]. A *uniform* bound on the empirical performance of policies within a simulation model of (partially observable) MDPs is provided in [99]. 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 [42]; 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 [45]. Again, since both algorithms in this chapter are recursive, we chose the more descriptive "pursuit learning automata" (PLA) label.

The idea of multi-stage adaptive sampling has been adopted by the artificial intelligence (AI) game-playing community in the form of Monte Carlo tree search (MCTS), where it has become perhaps the dominant approach, "due to its spectacular success in the difficult problem of computer Go" (abstract, [26]). The MCTS approach estimates the value of a potential move by building a sampled game tree using simulation, analogous to the multi-stage UCB sampling algorithm of Sect. 2.2.1 (see Fig. 2.3), the major difference being that because the state space is finite, nodes may be revisited multiple times during the sampling process and hence values stored to increase the computational efficiency substantially. In fact, what is declared "the most popular algorithm in the MCTS family, the *Upper Confidence Bound for Trees (UCT) algorithm*" (p. 7, Sect. 3.3, [26]) is acknowledged in [108] to be closely related to the multi-stage adaptive sampling UCB algorithm introduced in [42].

Chapter 3
Population-Based Evolutionary Approaches

In this chapter, we present population-based evolutionary algorithms for finding optimal (stationary) policies for *infinite*-horizon MDPs. Specifically, the goal is to find a stationary policy $\pi^* \in \Pi_s$ that maximizes the expected total discounted reward given by Eq. (1.11). Note that the Bellman optimality principle, which can be used to find $\pi^*(x)$ via (1.13) by serving as the basis of the policy improvement step in policy iteration (PI) via Eq. (1.18), must optimize over the *entire* action space $A(x)$ for each state x. This can pose a computational challenge for problems with large (possibly uncountable) action spaces. The algorithms in this chapter are targeted for such large action space problems where the policy improvement step in PI becomes computationally prohibitive, and value iteration (VI) is also impractical. The computational complexity of each iteration of our algorithms is polynomial in the size of the state space, but unlike PI and VI, it is insensitive to the size of the action space, making the algorithms most suitable for problems with relatively small state spaces compared to the size of the action spaces. In the case of infinite action spaces, our approach avoids the need for any discretization or truncation that could lead to computational difficulties, either resulting in an action space that is too large or in a solution that is not accurate enough.

The approach taken by the algorithms in this chapter directly searches the *policy space* to avoid carrying out an optimization over the entire action space at each PI step, and updates a *population* of policies as in genetic algorithms (GAs), using appropriate analogous operations for the MDP setting. A key feature of the algorithms is an *elite policy* that has a value function at least as good as the best value function in the previous population. The other key feature is an *action selection distribution* that generates mutations of policies to explore the policy space. The monotonicity property of the elite policy and the exploration property of the action selection distribution ensure that the algorithms converge with probability one to a population in which the elite policy is an optimal policy. A description of a general framework for the population-based evolutionary approach is provided in Fig. 3.1, where $\Lambda_k \subset \Pi_s$ denotes the kth generation population of policies and $n = |\Lambda_k| > 1$ is the constant population size.

H.S. Chang et al., *Simulation-Based Algorithms for Markov Decision Processes*, Communications and Control Engineering,
DOI 10.1007/978-1-4471-5022-0_3, © Springer-Verlag London 2013

General Population-Based Evolutionary Framework

Input: MDP $(X, A, A(\cdot), P, R)$, population size $n > 1$, initial population Λ_0, action selection distribution $\mathcal{P}_x \ \forall x \in X$, other policy mutation parameters.

Initialization: Set iteration count $k = 0$.

Loop until Stopping Rule is satisfied:
- Generate an **Elite Policy** $\hat{\pi}^k$ based on Λ_k.
- **Exploration:** Generate $(n-1)$ policies $\{\tilde{\pi}^1, \ldots, \tilde{\pi}^{n-1}\}$ via mutation operators and action selection distribution.
- **Next Population Generation:** $\Lambda_{k+1} = \{\hat{\pi}^k, \tilde{\pi}^1, \ldots, \tilde{\pi}^{n-1}\},\ i = 1, \ldots, n-1$.
- $k \leftarrow k + 1$.

Output: $\hat{\pi}^k$ an estimated optimal policy.

Fig. 3.1 Population-based evolutionary framework

We now formally define the key concepts described above. An *elite policy* $\hat{\pi} \in \Pi_s$ for $\Lambda \subset \Pi_s$ is a policy such that $\forall \pi \in \Lambda$,

$$V^{\hat{\pi}}(x) \geq V^{\pi}(x) \quad \forall x \in X.$$

If $\hat{\pi}^k$ denotes an elite policy for generation k, then the monotonicity property is as follows:

$$V^{\hat{\pi}^{k+1}}(x) \geq V^{\hat{\pi}^k}(x) \quad \forall x \in X.$$

An *action selection distribution* $\mathcal{P}_x$ for state $x \in X$ is a probability distribution over the action space $A(x)$. If the action space is discrete, then we will let $\mathcal{P}_x(a)$, $a \in A(x)$ denote the probability of action a, where $\sum_{a \in A(x)} \mathcal{P}_x(a) = 1$ and $\mathcal{P}_x(a) \geq 0$ for all $a \in A(x)$. If the action space is continuous, then $\mathcal{P}_x(\mathcal{A})$ will denote the probability for the (open) set $\mathcal{A} \subset A(x)$.

We begin with a basic algorithm called evolutionary policy iteration (EPI) that contains the main features of the population-based evolutionary approach and establish theoretical convergence. EPI uses an operation called policy switching to generate an elite policy, and has primitive mutation operations to ensure exploration of the policy space. Evolutionary random policy search (ERPS) builds upon the ideas of EPI and enhances the approach considerably by using more sophisticated mechanisms to generate an elite policy and to explore the policy space. The resulting algorithm is more computationally efficient, as well as being convergent in the broader setting of uncountable action spaces. Both algorithms are tested on some numerical examples, and are compared with PI to illustrate the substantial computational efficiency gains attainable by the approach.

3.1 Evolutionary Policy Iteration

Evolutionary policy iteration (EPI) is an iterative algorithm that works with a set (population) of policies and which eliminates the operation of maximization over the entire action space in the policy improvement step of PI. Central to EPI is an operation called *policy switching*, which generates an improved policy from a set of given policies, with a computation time on the order of the size of the state space. Each iteration of EPI consists of two main steps: generation of an elite policy by policy switching, and exploration of the policy space by generating additional policies via mutation and policy switching. A high-level description of EPI is shown in Fig. 3.2, where some steps (e.g., mutation) are described at a conceptual level, with details provided in the following subsections. Convergence of the EPI algorithm is independent of the initial population Λ_0, due to the **Policy Mutation** step, which is described in more detail in Sect. 3.1.2.

3.1.1 Policy Switching

Given a non-empty finite subset $\Lambda \subset \Pi_s$, a policy π_{ps} generated by *policy switching* with respect to Λ is given by

$$\pi_{\text{ps}}(x) \in \left\{ \arg\max_{\pi \in \Lambda} \left(V^{\pi}(x) \right)(x) \right\}, \quad x \in X. \tag{3.1}$$

An important property of policy switching is that the generated policy improves any policy in Λ (Theorem 3 in [41]):

Theorem 3.1 *Consider a non-empty finite subset $\Lambda \subset \Pi_s$ and the policy π_{ps} generated by policy switching with respect to Λ given by (3.1). Then, for all $x \in X$, $V^{\pi_{\text{ps}}}(x) \geq \max_{\pi \in \Lambda} V^{\pi}(x)$.*

In each step, EPI generates a policy $\hat{\pi}^k$, called the elite policy with respect to the current population Λ_k, which improves any policy in Λ_k via policy switching. Thus, the new population Λ_{k+1} contains a policy that is superior to any policy in the previous population, i.e., the desired monotonicity property holds.

Corollary 3.2 *Under EPI, for all $k \geq 0$, $V^{\hat{\pi}^{k+1}}(x) \geq V^{\hat{\pi}^k}(x)$ $\forall x \in X$.*

Proof The proof is by induction. The base step ($k = 0$) is obvious from the definition of $\hat{\pi}^0$ and $\hat{\pi}^1$ by Theorem 3.1. Assume that $V^{\hat{\pi}^i}(x) \geq V^{\hat{\pi}^{i-1}}(x), x \in X, \ \forall i \leq k$. Because EPI includes $\hat{\pi}^k$ in Λ_{k+1}, the elite policy at $k+1$ is generated over a population that contains $\hat{\pi}^k$, implying that $V^{\hat{\pi}^{k+1}}(x) \geq V^{\hat{\pi}^k}(x), x \in X$. □

Because policy switching directly manipulates policies, eliminating the operation of maximization over the entire action space, its computational-time complexity

Evolutionary Policy Iteration (EPI)

Input: MDP $(X, A, A(\cdot), P, R)$, population size $n > 1$, initial population Λ_0,
action selection distribution $\mathcal{P}_x \ \forall x \in X$,
policy mutation parameters $q_0 \in [0, 1]$, $P_l \in (0, 1)$, $P_g \in (0, 1]$, $P_l < P_g$.

Initialization: Set iteration count $k = 0$.

Loop until Stopping Rule is satisfied:

- **Elite Policy** via policy switching:

  1. Obtain the value function V^π for each $\pi \in \Lambda_k$.
  2. Generate an elite policy of Λ_k defined by

$$\hat{\pi}^k(x) \in \left\{ \arg\max_{\pi \in \Lambda_k} \big(V^\pi(x)\big)(x) \right\}, \quad x \in X. \tag{3.2}$$

- **Exploration** via policy mutation and policy switching:

  1. Generate $n-1$ random subsets $S_i, i = 1, \ldots, n-1$ of Λ_k
     by selecting $m \in \{2, \ldots, n-1\}$ with equal probability
     and selecting m policies in Λ_k with equal probability.
  2. Generate $n-1$ policies $\pi^i,\ i = 1, \ldots, n-1$ defined by

$$\pi^i(x) \in \left\{ \arg\max_{\pi \in S_i} \big(V^\pi(x)\big)(x) \right\}, \quad x \in X. \tag{3.3}$$

  3. For each $\pi^i, i = 1, \ldots, n-1$, generate mutated policy $\tilde{\pi}^i$ from π^i by
     With probability q_0 (local mutation),
     changing $\pi^i(x)$ according to $\mathcal{P}_x$ w.p. P_l for each $x \in X$,
     else (with probability $1 - q_0$) (global mutation)
     changing $\pi^i(x)$ according to $\mathcal{P}_x$ w.p. P_g for each $x \in X$.

- **Next Population Generation:** $\Lambda_{k+1} = \{\hat{\pi}^k, \tilde{\pi}^1, \ldots, \tilde{\pi}^{n-1}\}$.
- $k \leftarrow k + 1$.

Output: $\hat{\pi}^k$ an estimated optimal policy.

Fig. 3.2 Evolutionary policy iteration (EPI) description

is $O(m|X|)$, where $m = |S_i|$, independent of the action space size, leading to $O(nm|X|)$ complexity in the **Exploration** step, and hence $O(nm|X|^3)$ overall when including the $O(|X|^2)$ complexity for the policy evaluation needed to compute V^π. On the other hand, applying a single-policy improvement step of PI directly to each policy in Λ_k, instead of generating $\pi(S_i),\ i = 1, \ldots, n-1$, is of complexity $O(n|X|^2|A|)$.

After policy switching is used to generate an elite policy, $n-1$ other policies are generated by the two steps described in the second half of the **Policy Switching** step. These policies are then mutated in the **Policy Mutation** step and combined with the elite policy to form the next generation.

3.1.2 Policy Mutation and Population Generation

Policy mutation takes a given policy, and for each state, alters the specified action probabilistically. The main reason to generate mutated policies is to ensure exploration of the entire policy space, making a probabilistic convergence guarantee possible. EPI uses two types of mutation—"local" and "global"—which are differentiated by how much of the policy is likely to be changed (mutated). Local mutation is intended to mimic local search of "nearby" policies, whereas global mutation allows EPI to escape from local optima. A high mutation probability indicates that many components of the policy vector are likely to be mutated, representing a more global change, whereas a low mutation probability implies that very little mutation is likely to occur, meaning a more localized perturbation. For this reason, we assume that $P_l \ll P_g$, with P_l being very close to zero and P_g being very close to one. The **Policy Mutation** step first determines whether the mutation will be local or global with probability q_0. If the policy π is globally (locally) mutated, for each state x, $\pi(x)$ is changed with probability $P_g(P_l)$. If a mutation does occur, it is carried out according to the action selection distribution $\mathcal{P}_x$, i.e., if the mutated policy is denoted by π', then the new policy is generated according to $P(\pi'(x) = a) = \mathcal{P}_x(a)$, for all mutated states x (the actions for all other states remain unchanged). For example, one simple $\mathcal{P}_x$ is the uniform action selection distribution, in which case the new (mutated) policy would randomly select a new action for each mutated state x (independently) with equal probability over the set of admissible actions $A(x)$. By guaranteeing that every feasible action has a positive probability of being selected, this mutation mechanism allows exploration of the entire policy space, guaranteeing theoretical convergence of the algorithm.

3.1.3 Stopping Rule

Unlike in PI, even if the value functions for the two consecutive elite policies are identical, this does not necessarily mean that the elite policy is an optimal policy, so implementation of EPI requires specification of a stopping rule. The simplest one is to stop the algorithm when a predefined maximum number of iterations is reached. In the numerical experiments reported in Sect. 3.3, we use one of the most common stopping rules in standard GAs, whereby the algorithm is stopped when no further improvement in the value function is obtained for several, say K, consecutive iterations. Specifically, we stop the algorithm at iteration k when

$$V^{\hat{\pi}^{k+m}}(x) = V^{\hat{\pi}^{k}}(x) \quad \forall x \in X;\ m = 1, 2, \ldots, K.$$

For increasing K, the probability of being in a neighborhood of the optimum increases, and the elite policy at termination is optimal with more confidence. In particular, as $K \to \infty$, $k \to \infty$, since the algorithm will terminate when $N = K$, and if $N \neq K$, the value of k increases by one.

3.1.4 Convergence Analysis

As mentioned earlier, the critical requirement for theoretical convergence is ensuring that the optimal policy has a non-zero probability of being sampled from the action sampling distribution.

Theorem 3.3 *For finite X and $q_0 \in (0, 1)$, if $P_l > 0$, $P_g > 0$, and $\exists$ an optimal policy π^* and $\epsilon > 0$ s.t. $P_x(\pi^*(x)) \geq \epsilon \forall x$, then*

$$V^{\hat{\pi}^k}(x) \to V^{\pi^*}(x), \quad x \in X \text{ w.p.1},$$

as $k \to \infty$, uniformly over X, regardless of Λ_0.

Proof Let γ be the probability of generating an optimal policy by local mutation, and let β the probability of generating an optimal policy by global mutation. Then, letting π^* denote an optimal policy in Π_s, we have

$$\gamma \geq \prod_{x \in X} P_l \mathcal{P}_x\big(\pi^*(x)\big) \geq (P_l \epsilon)^{|X|} > 0,$$

$$\beta \geq \prod_{x \in X} P_g \mathcal{P}_x\big(\pi^*(x)\big) \geq (P_g \epsilon)^{|X|} > 0,$$

since $|X| < \infty$, $P_l > 0$, $P_g > 0$, and $\mathcal{P}_x(\pi^*(x)) > 0 \ \forall x$. Thus, the probability of generating an optimal policy by the **Policy Mutation** step is positive, independent of Λ_0.

Therefore, the probability that Λ_k does not contain an optimal policy (starting from an arbitrary Λ_0) is at most

$$\big((1-\gamma)(1-q_0)\big)^{(n-1)k}\big((1-\beta)(q_0)\big)^{(n-1)k},$$

which goes to zero as $k \to \infty$. By Lemma 3.2, once Λ_k contains an optimal policy, Λ_{k+m} contains an optimal policy for any $m \geq 1$, because the value of an optimal policy at any state is the maximum among all policies in Π_s. □

Theoretical convergence actually does not depend on the value of q_0, since the distinction between global and local mutation is arbitrary. If q_0 does take one of its extreme values (i.e., 0 or 1), then a corresponding mutation probability can be set to zero, since it will never be selected, and the proof will go through similarly. For example, if $q_0 = 0$, then $P_g > 0$ ensures $\beta > 0$, and the probability that Λ_k does not contain an optimal policy is at most $(1-\beta)^{(n-1)k}$.

Of course, the condition on π^* is difficult to verify in practice, since an optimal policy is unknown a priori, but if the feasible action set is finite, the condition can be easily satisfied.

Corollary 3.4 *For finite X and $q_0 \in (0,1)$, if $P_l > 0$, $P_g > 0$, and $A(x)$ is finite (and non-empty) for each $x \in X$, and $\mathcal{P}_x(a) > 0\ \forall a \in A(x)\ \forall x \in X$, then*

$$V^{\hat{\pi}^k}(x) \to V^{\pi^*}(x), \quad x \in X \text{ w.p.1},$$

as $k \to \infty$, uniformly over X, regardless of Λ_0.

Proof Since X is finite and $A(x)$ is finite for each $x \in X$, the condition in Theorem 3.3 is satisfied by simply taking $\epsilon = \min_{x \in X} \min_{a \in A(x)} \mathcal{P}_x(a)$. □

One possible simple choice for the action selection distribution P_x is the (discrete) uniform distribution on $A(x)$, e.g., if the actions in $A(x)$ are numbered $1, 2, \ldots, |A(x)|$, then $P_x \sim DU(1, |A(x)|)$, where $\epsilon = 1/(\max_{x \in X} |A(x)|)$ in the proof.

3.1.5 Parallelization

The EPI algorithm can be naturally parallelized to improve the convergence by partitioning the policy space Π_s into subsets of $\{\Pi^i\}$ such that $\bigcup_i \Pi^i = \Pi_s$ and $\Pi^i \cap \Pi^j = \emptyset\ \forall i \neq j$, and applying EPI to each Π^i in parallel. Once each parallel application of EPI terminates, the best policy $\pi^{*,i}$ from each is taken, and policy switching is applied to the set of best policies $\{\pi^{*,i}\}$. If the number of subsets in the partition is N, the overall convergence of the algorithm $\mathcal{A}$ is faster by a factor of N. We state a general result regarding such parallelization.

Theorem 3.5 *For a partition $\{\Pi^i\}$ of Π_s, let $\pi^{*,i}$ denote an optimal policy in Π^i such that for all $x \in X$, $V^{\pi^{*,i}}(x) \geq \max_{\pi \in \Pi^i} V^{\pi}(x)$. Then, an optimal policy in Π_s is given by*

$$\pi^*(x) \in \left\{ \arg\max_{\{\pi^{*,i}\}} \left(V^{\pi^{*,i}}(x) \right)(x) \right\}, \quad x \in X.$$

Proof Via policy switching, π^* improves the performance of each $\pi^{*,i}$, i.e., $V^{\pi^*}(x) \geq \max_{\{\pi^{*,i}\}} V^{\pi^{*,i}}(x),\ x \in X$, implying that π^* is an optimal policy for Π_s, since the partition covers the entire policy space. □

3.2 Evolutionary Random Policy Search

Evolutionary random policy search (ERPS) is an enhancement of EPI that improves upon both the elite policy determination and the mutation step by solving a sequence of sub-MDP problems defined on smaller policy spaces. As in EPI, each iteration of ERPS has two main steps:

1. An elite policy is generated by solving the sub-MDP problem constructed in the previous iteration using a variant of the policy improvement technique called policy improvement with reward swapping (PIRS).
2. Based on the elite policy, a group of policies is then obtained by using a "nearest neighbor" heuristic and random sampling of the entire action space, from which a new sub-MDP is created by restricting the original MDP problem (e.g., reward structure, transition probabilities) to the current available subsets of actions.

In contrast to EPI, which treats policies as the most essential elements in the action optimization step, and each elite policy is directly generated from a group of policies, in ERPS policies are regarded as intermediate constructions from which sub-MDP problems are then constructed and solved. Furthermore, the "nearest neighbor" heuristic provides a local search mechanism that leads to rapid convergence once a policy is found in a small neighborhood of an optimal policy. Under appropriate assumptions, the sequence of elite policies converges with probability one to an optimal policy, even for uncountable action spaces.

A high-level description of the ERPS algorithm is presented in Fig. 3.3. Detailed discussion of each of the steps follows. The input parameters are similar to EPI, with the search range parameters $\{r_i\}$ used to construct sub-MDPs replacing the mutation parameters P_g/P_l.

3.2.1 Policy Improvement with Reward Swapping

As mentioned earlier, the idea behind ERPS is to randomly split a large MDP problem into a sequence of smaller, manageable MDPs, and to extract a convergent sequence of policies via solving these smaller problems. For a given policy population Λ, if we restrict the original MDP (e.g., rewards, transition probabilities) to the subsets of actions $\Gamma(x) := \{\pi(x) : \pi \in \Lambda\}\ \forall x \in X$, then a sub-MDP problem is induced from Λ as $\mathcal{G}_\Lambda := (X, \Gamma, \Gamma(\cdot), P, R)$, where $\Gamma := \bigcup_x \Gamma(x) \subset A$. Note that in general $\Gamma(x)$ is a multi-set, which means that the set may contain repeated elements; however, we can always discard the redundant members and view $\Gamma(x)$ as the set of admissible actions at state x. Since ERPS is an iterative random search algorithm, rather than attempting to solve $\mathcal{G}_\Lambda$ exactly, it is more efficient to use approximation schemes and obtain an improved policy and/or good candidate policies with worst-case performance guarantee.

Given a non-empty finite subset $\Lambda \subset \Pi_s$, a policy π_{pirs} generated by *policy improvement with reward swapping* (PIRS) with respect to the sub-MDP $\mathcal{G}_\Lambda$ is given by

$$\pi_{\text{pirs}}(x) \in \arg\max_{u \in \Gamma(x)} \left\{ R(x,u) + \gamma \sum_{y \in X} P(x,u)(y) \bar{V}^{\Lambda}(y) \right\}, \tag{3.4}$$

where $\bar{V}^{\Lambda}(x) = \max_{\pi \in \Lambda} V^{\pi}(x)\ \forall x \in X$. PIRS is a variation of the policy improvement step in PI (cf. (1.24)) performed on the "swapped reward" $\bar{V}^{\Lambda}(x) =$

$\max_{\pi\in\Lambda} V^{\pi}(x)$, hence the name policy improvement with reward swapping. Note that the "swapped reward" $\bar{V}^{\Lambda}(x)$ may not be the value function corresponding to any single policy in Λ. However, we now show that the elite policy generated by PIRS improves any policy in Λ.

Theorem 3.6 *Consider a non-empty finite subset $\Lambda \subset \Pi_s$ and the policy π_{pirs} generated by PIRS with respect to $\mathcal{G}_{\Lambda}$ given by (3.4). Then, for all $x \in X$, $V^{\pi_{\text{pirs}}}(x) \geq \bar{V}^{\Lambda}(x)$. Furthermore, if π_{pirs} is not optimal for the sub-MDP $\mathcal{G}_{\Lambda}$, then $V^{\pi_{\text{pirs}}}(x) > \bar{V}^{\Lambda}(x)$ for at least one $x \in X$.*

Proof Let $V_0(x) = R(x, \pi_{\text{pirs}}(x)) + \gamma \sum_{y\in X} P(x, \pi_{\text{pirs}}(x))(y)\bar{V}^{\Lambda}(y)$, and consider the sequence $V_1(x), V_2(x), \ldots$ generated by the recursion $V_{i+1}(x) = R(x, \pi_{\text{pirs}}(x)) + \gamma \sum_{y\in X} P(x, \pi_{\text{pirs}}(x))(y)V_i(y),\ \forall i = 0, 1, 2, \ldots$. At state x, by definition of $\bar{V}^{\Lambda}(x)$, there exists $\pi \in \Lambda$ such that $\bar{V}^{\Lambda}(x) = V^{\pi}(x)$. It follows that

$$\begin{aligned} V_0(x) &\geq R\big(x, \pi(x)\big) + \gamma \sum_{y\in X} P\big(x, \pi(x)\big)(y)\bar{V}^{\Lambda}(y) \\ &\geq R\big(x, \pi(x)\big) + \gamma \sum_{y\in X} P\big(x, \pi(x)\big)(y)V^{\pi}(y) \\ &= V^{\pi}(x) \\ &= \bar{V}^{\Lambda}(x), \end{aligned}$$

and since x is arbitrary, we have

$$\begin{aligned} V_1(x) &= R\big(x, \pi_{\text{pirs}}(x)\big) + \gamma \sum_{y\in X} P\big(x, \pi_{\text{pirs}}(x)\big)(y)V_0(y) \\ &\geq R\big(x, \pi_{\text{pirs}}(x)\big) + \gamma \sum_{y\in X} P\big(x, \pi_{\text{pirs}}(x)\big)(y)\bar{V}^{\Lambda}(y) \\ &= V_0(x). \end{aligned}$$

By induction $V_{i+1}(x) \geq V_i(x),\ \forall x \in X$ and $\forall i = 0, 1, 2, \ldots$. It is well known [12] that the sequence $V_0(x), V_1(x), V_2(x), \ldots$ generated above converges to $V^{\pi_{\text{pirs}}}(x)$, $\forall x \in X$. Therefore $V^{\pi_{\text{pirs}}}(x) \geq \bar{V}^{\Lambda}(x),\ \forall x \in X$. If $V^{\pi_{\text{pirs}}}(x) = \bar{V}^{\Lambda}(x),\ \forall x \in X$, then PIRS reduces to the standard policy improvement on policy π_{pirs}, and it follows that π_{pirs} satisfies Bellman's optimality equation and is thus optimal for $\mathcal{G}_{\Lambda}$. Hence we must have $V^{\pi_{\text{pirs}}}(x) > \bar{V}^{\Lambda}(x)$ for some $x \in X$ whenever π_{pirs} is not optimal. □

According to Theorem 3.6, in each step of ERPS, the elite policy $\hat{\pi}^k$ generated by PIRS with respect to the current sub-MDP $\mathcal{G}_{\Lambda_k}$, as given by (3.5), improves any policy in Λ_k. Thus, the new population Λ_{k+1} contains a policy that is superior to any policy in the previous population. Since $\hat{\pi}^k$ is directly used to generate the $(k+1)$th sub-MDP (see Fig. 3.3 and Sect. 3.2.2), the desired monotonicity property follows by the same induction argument used to establish Lemma 3.2.

Evolutionary Random Policy Search (ERPS)

Input: MDP $(X, A, A(\cdot), P, R)$, population size $n > 1$, initial population Λ_0, action selection distribution $\mathcal{P}_x \ \forall x \in X$, policy mutation parameters: exploitation probability $q_0 \in [0, 1]$, search range $r_i, \forall x_i \in X,\ i = 1, \ldots, |X|$.

Initialization: Set iteration count $k = 0$.
Construct initial sub-MDP as $\mathcal{G}_{\Lambda_0} := (X, \Gamma_0, \Gamma_0(\cdot), P, R),\ \Gamma_0 = \bigcup_x \Gamma_0(x)$.

Loop until Stopping Rule is satisfied:

- **Elite Policy** via policy improvement with reward swapping (PIRS):
  1. Obtain the value function V^π for each $\pi \in \Lambda_k$.
  2. Generate an elite policy of Λ_k using sub-MDP $\mathcal{G}_{\Lambda_k}$:

$$\hat{\pi}^k(x) \in \arg\max_{u \in \Gamma_k(x)} \left\{ R(x, u) + \gamma \sum_{y \in X} P(x, u)(y) \left[\max_{\pi \in \Lambda_k} V^\pi(y) \right] \right\}, \quad x \in X. \tag{3.5}$$

- **Exploration** via mutation and local nearest neighbor search:
  Generate $n - 1$ policies $\tilde{\pi}^i,\ i = 1, \ldots, n - 1$, by

  With probability q_0 (exploitation)
  choose the action $\tilde{\pi}^i(x)$ in the neighborhood of $\hat{\pi}^k(x),\ x \in X$,
  by using the "nearest neighbor" heuristic.
  else (with probability $1 - q_0$) (exploration)
  choose the action $\tilde{\pi}^i(x) \in A(x)$ according to $\mathcal{P}_x,\ x \in X$.

- **Next Population Generation:** $\Lambda_{k+1} = \{\hat{\pi}^k, \tilde{\pi}^1, \ldots, \tilde{\pi}^{n-1}\}$.
- Next sub-MDP: $\mathcal{G}_{\Lambda_{k+1}} := (X, \Gamma_{k+1}, \Gamma_{k+1}(\cdot), P, R)$, $\Gamma_{k+1} = \bigcup_x \Gamma_{k+1}(x)$.
- $k \leftarrow k + 1$.

Output: $\hat{\pi}^k$ an estimated optimal policy.

Fig. 3.3 Evolutionary random policy search (ERPS) description

Corollary 3.7 *Under ERPS, for all* $k \geq 0$,

$$V^{\hat{\pi}^{k+1}}(x) \geq V^{\hat{\pi}^k}(x) \quad \forall\, x \in X. \tag{3.6}$$

We now provide an intuitive comparison between PIRS and policy switching, which is used in EPI and directly operates upon individual policies in the population via (3.1), with a computational complexity is $O(n|X|)$. For a given group of policies Λ, let Ω be the policy space for the sub-MDP $\mathcal{G}_\Lambda$; it is clear that the size of Ω is on the order of $n^{|X|}$. Policy switching only takes into account each individual policy in Λ, while PIRS tends to search the entire space Ω, which is much larger than Λ. Although it is not clear in general that the elite policy generated by PIRS improves the elite policy generated by policy switching, since the policy improvement step is quite fast and it focuses on the best policy updating directions, this should be the case in many situations. For example, consider the case where one particular policy, say $\bar{\pi}$, dominates all other policies in Λ. It is obvious that policy

switching will choose $\bar{\pi}$ as the elite policy; thus, no further improvement can be achieved. In contrast, PIRS considers the sub-MDP $\mathcal{G}_\Lambda$; as long as $\bar{\pi}$ is not optimal for $\mathcal{G}_\Lambda$, a better policy can always be obtained.

The computational complexity of each iteration of PIRS is approximately the same as that of policy switching, because the policy evaluation step of PIRS, which is also used by policy switching, requires solution of n systems of linear equations, and the number of operations required by using a direct method (e.g., Gaussian elimination) is $O(n|X|^3)$, and this dominates the computational complexity of the policy improvement step, which is at most $O(n|X|^2)$.

3.2.2 Exploration

According to Corollary 3.7, the performance of the elite policy at the current iteration is no worse than the performances of the elite policies generated at previous iterations; thus the PIRS step alone can be viewed as a local search procedure with memory. As in EPI, to guarantee theoretical convergence, a mechanism to globally explore the policy space is required. One possibility is to use unbiased random sampling and choose at each iteration a sub-MDP problem by making use of the action selection distribution $\mathcal{P}_x$. The sub-MDPs at successive iterations are then independent of one another, and it is intuitively clear that we may obtain improved elite policies after a sufficient number of iterations. Such an unbiased sampling scheme is very effective in escaping local optima and is often useful in finding a good candidate solution. However, in practice, persistent improvements will be more and more difficult to achieve as the number of iterations (sampling instances) increases, since the probability of finding better elite policies becomes smaller and smaller. Thus, it appears that a biased sampling scheme could be more helpful, which can be accomplished by using a "nearest neighbor" heuristic.

To achieve a biased sampling configuration, ERPS combines exploitation ("nearest neighbor" heuristic) with exploration (unbiased sampling). The key to balancing these two types of searches is the use of the exploitation probability q_0. For a given elite policy $\hat{\pi}$, we construct a new policy, say $\tilde{\pi}$, in the next population generation as follows: At each state $x \in X$, with probability q_0, $\tilde{\pi}(x)$ is selected from a small neighborhood of $\hat{\pi}(x)$; and with probability $1 - q_0$, $\tilde{\pi}(x)$ is chosen by using the unbiased random sampling. The preceding procedure is performed repeatedly until we have obtained $n - 1$ new policies, and the next population generation is simply formed by the elite policy $\hat{\pi}$ and the $n - 1$ newly generated policies. Intuitively, on the one hand, the use of exploitation will introduce more robustness into the algorithm and helps to locate the exact optimal policy, while on the other hand, the exploration step will help the algorithm to escape local optima and to find attractive policies quickly. In effect, we see that this idea is equivalent to altering the underlying *action selection distribution*, in that $\mathcal{P}_x$ is artificially made more peaked around the action $\hat{\pi}(x)$.

Assuming that A is a non-empty metric space with a defined metric $d(\cdot, \cdot)$, then the "nearest neighbor" heuristic in ERPS (cf. Fig. 3.3) is implemented as follows:

- *Discrete Action Space*: Let r_i, a positive integer, be the search range for state $x_i,\ i = 1, 2, \ldots, |X|$, with $r_i < |A(x_i)|\ \forall i$. Generate a random variable $l \sim DU(1, r_i)$, where $DU(1, r_i)$ represents the discrete uniform distribution between 1 and r_i. Set $\tilde{\pi}^j(x_i) = a \in A(x_i)$ such that a is the lth closest action to $\hat{\pi}^k(x_i)$ (measured by $d(\cdot, \cdot)$).
- *Continuous Action Space*: Let $r_i > 0$ denote the search range for state $x_i,\ i = 1, 2, \ldots, |X|$. Choose an action uniformly from the set of neighbors $\{a : d(a, \hat{\pi}^k(x_i)) \leq r_i,\ a \in A(x_i)\}$, i.e., an action $\pi(x_i) \in A$ such that $d(\pi(x_i), \pi^k_*(x_i)) \leq r_i$.

The most favorable situation is an action space that is "naturally ordered," e.g., in inventory control problems where actions are the number of items or amount to be ordered ($A = \{0, 1, 2, \ldots\}$ or $A = \Re^+$, respectively), in which case the indexing and ordering becomes trivial.

Note the difference in the search range r_i between the discrete action space case and the continuous action space case. In the former case, r_i is a positive integer indicating the number of candidate actions that are the closest to the current elite action $\hat{\pi}^k(x_i)$, whereas in the latter case, r_i is the distance from the current elite action, which may take any positive value.

For the continuous case, if we further assume that A is a non-empty open connected subset of $\Re^N$ with some metric (e.g., the infinity-norm), then a detailed implementation of the above exploitation (local search) step is as follows.

- Generate a random vector $\lambda^i = (\lambda^i_1, \ldots, \lambda^i_N)^T$ with each $\lambda^i_h \sim U[-1, 1]$ independent for all $h = 1, 2, \ldots, N$, and choose the action $\tilde{\pi}^j(x_i) = \hat{\pi}^k(x_i) + \lambda^i r_i$.
- *If* $\tilde{\pi}^j(x_i) \notin A(x_i)$, then repeat the above step.

In this specific implementation, the same search range r_i is used along all directions of the action space. However, in practice, it may often be useful to generalize r_i to an N-dimensional vector, where each component controls the search range in a particular direction of the action space.

Note that the action space does not need to have any structure other than being a metric space. The metric $d(\cdot, \cdot)$ used in the "nearest neighbor" heuristic implicitly imposes a structure on the action space. It follows that the efficiency of the algorithm depends on how the metric is actually defined. Like most of the random search methods for global optimization, our approach is designed to explore the structure that good policies tend to be clustered together. Thus, in our context, a good metric should have a good potential in representing this structure. For example, the discrete metric (i.e., $d(a, a) = 0\ \forall a \in A$ and $d(a, b) = 1\ \forall a, b \in A,\ a \neq b$) should never be considered as a good choice, since it does not provide us with any useful information about the action space. For a given action space, a good metric always exists but may not be known a priori. In the special case where the action space is a subset of $\Re^N$, we take the Euclidean metric as the default metric, this is in accord with most of the optimization techniques employed in $\Re^N$.

3.2.3 Convergence Analysis

We define the distance between two policies π and π' by

$$d_\infty(\pi, \pi') := \max_{1 \le i \le |X|} d\big(\pi(x_i), \pi'(x_i)\big),$$

and for a given policy $\pi \in \Pi_s$ and any $\sigma > 0$, we define the σ-neighborhood of π by

$$\mathcal{N}(\pi, \sigma) := \big\{\pi' \mid d_\infty(\pi, \pi') \le \sigma,\ \forall \pi' \in \Pi_s\big\}.$$

For each policy $\pi \in \Pi_s$, we also define P_π as the transition matrix whose (x, y)th entry is $P(x, \pi(x))(y)$ and R_π as the one-stage reward vector whose (x)th entry is $R(x, \pi(x))$.

Since ERPS is a randomized algorithm, different runs of the algorithm will give different sequences of elite policies (i.e., sample paths); thus, ERPS induces a probability distribution over the set of all sequences of elite policies. We denote by $\hat{\mathcal{P}}(\cdot)$ and $\hat{E}(\cdot)$ the probability and expectation taken with respect to this distribution.

Let $\|\cdot\|_\infty$ denote the infinity-norm, given by $\|V\|_\infty := \max_{x \in X} |V(x)|$. If the argument is a matrix, it will denote the usual induced or spectral norm. We have the following convergence result for the ERPS algorithm.

Theorem 3.8 *Let π^* be an optimal policy with corresponding value function V^{π^*}, and let the sequence of elite policies generated by ERPS together with their corresponding value functions be denoted by $\{\hat{\pi}^k,\ k = 1, 2, \ldots\}$ and $\{V^{\hat{\pi}^k},\ k = 1, 2, \ldots\}$, respectively. Assume that*:

1. $q_0 < 1$.
2. *For any given $\ell > 0$, $\mathcal{P}_x(\{a|\ d(a, \pi^*(x)) \le \ell,\ a \in A(x)\}) > 0,\ \forall x \in X$.*
3. *There exist constants $\sigma > 0,\ \phi > 0,\ L_1 < \infty$, and $L_2 < \infty$, such that for all $\pi \in \mathcal{N}(\pi^*, \sigma)$ we have $\|P_\pi - P_{\pi^*}\|_\infty \le \max\{L_1 d_\infty(\pi, \pi^*), \frac{1-\gamma}{\gamma} - \phi\}$ $(0 < \gamma < 1)$, and $\|R_\pi - R_{\pi^*}\|_\infty \le L_2 d_\infty(\pi, \pi^*)$.*

Then for any given $\varepsilon > 0$, there exists a random variable $\mathcal{M}_\varepsilon > 0$ with $\hat{E}(\mathcal{M}_\varepsilon) < \infty$ such that $\|V^{\hat{\pi}^k} - V^{\pi^*}\|_\infty \le \varepsilon\ \forall k \ge \mathcal{M}_\varepsilon$.

Assumption 1 restricts the exploitation probability from pure local search. Assumption 2 simply requires that any "ball" that contains the optimal policy will have a strictly positive probability measure. It is trivially satisfied if the set $\{a|d(a, \pi^*(x)) \le \ell,\ a \in A(x)\}$ has a positive (Borel) measure $\forall\, x \in X$ and the action selection distribution $\mathcal{P}_x$ has infinite tails (e.g., Gaussian distribution). Assumption 3 imposes some Lipschitz type of conditions on P_π and R_π; as we will see, it formalizes the notion that near optimal policies are clustered together. The assumption can be verified if P_π and R_π are explicit functions of π (which is the case in our numerical examples; see Sect. 3.3). For a given $\varepsilon > 0$, such a policy π satisfying $\|V^\pi - V^{\pi^*}\|_\infty \le \varepsilon$ is referred to as an ε-optimal policy.

The result in Theorem 3.8 implies the a.s. convergence of the sequence $\{V^{\hat{\pi}^k}, k = 0, 1, \ldots\}$ to the optimal value function V^{π^*}. To see this, note that Theorem 3.8 implies that $\hat{\mathcal{P}}(\|V^{\hat{\pi}^k} - V^{\pi^*}\|_\infty > \varepsilon) \to 0$ as $k \to \infty$ for every given ε, which means that the sequence converges in probability. Furthermore, since $\|V^{\hat{\pi}^k} - V^{\pi^*}\|_\infty \leq \varepsilon \ \forall k \geq \mathcal{M}_\varepsilon$ is equivalent to $\sup_{\bar{k}\geq k} \|V^{\hat{\pi}^{\bar{k}}} - V^{\pi^*}\|_\infty \leq \varepsilon \ \forall k \geq \mathcal{M}_\varepsilon$, we will also have $\hat{\mathcal{P}}(\sup_{\bar{k}\geq k} \|V^{\hat{\pi}^{\bar{k}}} - V^{\pi^*}\|_\infty > \varepsilon) \to 0$ as $k \to \infty$, and the a.s. convergence thus follows.

Proof The first step in the proof is to derive an upper bound for $\|V^\pi - V^{\pi^*}\|_\infty$ in terms of the distance $d_\infty(\pi, \pi^*)$. For policy π^* and policy π, we have

$$V^{\pi^*} = R_{\pi^*} + \gamma P_{\pi^*} V^{\pi^*}, \tag{3.7}$$

$$V^{\pi} = R_{\pi} + \gamma P_{\pi} V^{\pi}. \tag{3.8}$$

Now subtract the above two equations and define $\Delta V^{\pi^*} = V^\pi - V^{\pi^*}$, $\Delta P_{\pi^*} = P_\pi - P_{\pi^*}$ and $\Delta R_{\pi^*} = R_\pi - R_{\pi^*}$. We have

$$\Delta V^{\pi^*} = \left[I - (I - \gamma P_{\pi^*})^{-1} \gamma \Delta P_{\pi^*}\right]^{-1} (I - \gamma P_{\pi^*})^{-1} \left(\gamma \Delta P_{\pi^*} V^{\pi^*} + \Delta R_{\pi^*}\right). \tag{3.9}$$

Taking the norm of both sides of Eq. (3.9) and using the consistency property of the operator norm (i.e., $\|BC\| \leq \|B\| \cdot \|C\|$), it follows that

$$\left\|\Delta V^{\pi^*}\right\|_\infty \leq \left\|\left[I - (I - \gamma P_{\pi^*})^{-1} \gamma \Delta P_{\pi^*}\right]^{-1}\right\|_\infty \left\|(I - \gamma P_{\pi^*})^{-1}\right\|_\infty \tag{3.10}$$

$$\times \left(\gamma \|\Delta P_{\pi^*}\|_\infty \|V^{\pi^*}\|_\infty + \left\|\Delta R_{\pi^*}\right\|_\infty\right). \tag{3.11}$$

By Assumption 3, we have $\|\Delta P_{\pi^*}\|_\infty < \frac{1-\gamma}{\gamma}$, thus

$$\begin{aligned} \left\|\left(I - \gamma P_{\pi^*}\right)^{-1} \gamma \Delta P_{\pi^*}\right\|_\infty &\leq \left\|(I - \gamma P_{\pi^*})^{-1}\right\|_\infty \gamma \|\Delta P_{\pi^*}\|_\infty \\ &< \left\|(I - \gamma P_{\pi^*})^{-1}\right\|_\infty (1 - \gamma) \\ &< 1. \end{aligned}$$

We now try to divide both sides of (3.10) by $\|V^{\pi^*}\|_\infty$. Before we proceed, we need to distinguish between two cases, $\|V^{\pi^*}\|_\infty = 0$ and $\|V^{\pi^*}\|_\infty \neq 0$.

Case 1. If $R_{\pi^*} = 0$ (i.e., $R(x, \pi^*(x)) = 0$ for all $x \in X$), then we have $V^{\pi^*} = 0$. Thus $\Delta V^{\pi^*} = V^\pi$ and $\Delta R_{\pi^*} = R_\pi$. By noting $\|P_\pi\|_\infty = 1$, it follows from Eq. (3.8) that

$$\left\|\Delta V^{\pi^*}\right\|_\infty = \left\|V^\pi\right\|_\infty \leq \frac{1}{1 - \gamma \|P_\pi\|_\infty} \|R_\pi\|_\infty = \frac{1}{1-\gamma} \|\Delta R_{\pi^*}\|_\infty. \tag{3.12}$$

Then by Assumption 3,

$$\left\|\Delta V^{\pi^*}\right\|_\infty \leq \frac{L_2}{1-\gamma} d_\infty(\pi, \pi^*). \tag{3.13}$$

Case 2. If $R_{\pi^*} > 0$ (i.e., $R(x, \pi^*(x)) > 0$ for some $x \in X$), then from Eq. (3.7), $V^{\pi^*} > 0$. Divide both sides of (3.10) by $\|V^{\pi^*}\|_\infty$, use the relation that $\|(I - B)^{-1}\| \leq \frac{1}{1-\|B\|}$ whenever $\|B\| < 1$ and the consistency property; it immediately follows that

$$\begin{aligned}
\frac{\|\Delta V^{\pi^*}\|_\infty}{\|V^{\pi^*}\|_\infty} &\leq \frac{\|(I - \gamma P_{\pi^*})^{-1}\|_\infty}{1 - \|(I - \gamma P_{\pi^*})^{-1}\|_\infty \gamma \|\Delta P_{\pi^*}\|_\infty} \left\{ \gamma \|\Delta P_{\pi^*}\|_\infty + \frac{\|\Delta R_{\pi^*}\|_\infty}{\|V^{\pi^*}\|_\infty} \right\} \\
&= \frac{\|(I - \gamma P_{\pi^*})^{-1}\|_\infty \|I - \gamma P_{\pi^*}\|_\infty}{1 - \|(I - \gamma P_{\pi^*})^{-1}\|_\infty \gamma \|\Delta P_{\pi^*}\|_\infty} \\
&\quad \times \left\{ \frac{\gamma \|\Delta P_{\pi^*}\|_\infty}{\|I - \gamma P_{\pi^*}\|_\infty} + \frac{\|\Delta R_{\pi^*}\|_\infty}{\|I - \gamma P_{\pi^*}\|_\infty \|V^{\pi^*}\|_\infty} \right\} \\
&\leq \frac{\mathcal{K}}{1 - \mathcal{K} \frac{\gamma \|\Delta P_{\pi^*}\|_\infty}{\|I - \gamma P_{\pi^*}\|_\infty}} \left\{ \frac{\gamma \|\Delta P_{\pi^*}\|_\infty}{\|I - \gamma P_{\pi^*}\|_\infty} + \frac{\|\Delta R_{\pi^*}\|_\infty}{\|R_{\pi^*}\|_\infty} \right\} \\
&\leq \frac{\mathcal{K}}{1 - \mathcal{K} \frac{\gamma \|\Delta P_{\pi^*}\|_\infty}{\|I - \gamma P_{\pi^*}\|_\infty}} \left\{ \frac{\gamma L_1}{\|I - \gamma P_{\pi^*}\|_\infty} + \frac{L_2}{\|R_{\pi^*}\|_\infty} \right\} d_\infty(\pi, \pi^*), \quad (3.14)
\end{aligned}$$

where $\mathcal{K} = \|(I - \gamma P_{\pi^*})^{-1}\|_\infty \|I - \gamma P_{\pi^*}\|_\infty$.

In view of (3.13) and (3.14), we conclude that for any given $\varepsilon > 0$, there exists $\theta > 0$ such that for any $\pi \in \mathcal{N}(\pi^*, \sigma)$ where

$$d_\infty(\pi, \pi^*) := \max_{1 \leq i \leq |X|} d(\pi(x_i), \pi^*(x_i)) \leq \theta, \quad (3.15)$$

we have $\|V^\pi - V^{\pi^*}\|_\infty = \|\Delta V^{\pi^*}\|_\infty \leq \varepsilon$. Also, $\max_{1 \leq i \leq |X|} d(\pi(x_i), \pi^*(x_i)) \leq \theta$ is equivalent to

$$d(\pi(x_i), \pi^*(x_i)) \leq \theta, \quad \forall i = 1, 2, \ldots, |X|. \quad (3.16)$$

By Assumption 2, the set of actions that satisfies (3.16) will have a strictly positive probability measure, and since $q_0 < 1$, it follows that the probability a population generation does not contain a policy in the neighborhood $\mathcal{N}(\pi^*, \min\{\theta, \sigma\})$ of the optimal policy is strictly less than 1. Let ψ be the probability that a randomly constructed policy is in $\mathcal{N}(\pi^*, \min\{\theta, \sigma\})$. Then at each iteration, the probability that at least one policy is obtained in $\mathcal{N}(\pi^*, \min\{\theta, \sigma\})$ is $1 - (1 - \psi)^{n-1}$, where n is the population size. Assume that, at iteration k, we obtain a policy $\tilde{\pi}^j$ in $\mathcal{N}(\pi^*, \min\{\theta, \sigma\})$. Then, it is guaranteed that $\|V^{\tilde{\pi}^j} - V^{\pi^*}\|_\infty \leq \varepsilon$ (by the initial part of the proof). The elite policy obtained at the next iteration improves all the available policies in Λ_{k+1} (by Theorem 3.6). Therefore, if $\hat{\pi}^{k+1}$ is the elite policy obtained in the next iteration, then we have $\|V^{\hat{\pi}^{k+1}} - V^{\pi^*}\|_\infty \leq \|V^{\tilde{\pi}^j} - V^{\pi^*}\|_\infty \leq \varepsilon$. Since we now have an elite policy $\hat{\pi}^{k+1}$ that satisfies $\|V^{\hat{\pi}^{k+1}} - V^{\pi^*}\|_\infty \leq \varepsilon$, then in subsequent iterations of the algorithm we will always have an elite policy in Λ_m such that $\|V^{\hat{\pi}^m} - V^{\pi^*}\|_\infty \leq \varepsilon$, for $m = k + 1, k + 2, \ldots$ (see Corollary 3.7). Let $\mathcal{M}_\varepsilon$ denote the number of iterations required to generate such an elite policy for the first time.

We clearly have $\|V^{\hat{\pi}^k} - V^{\pi^*}\|_\infty \leq \varepsilon \ \forall k \geq \mathcal{M}_\varepsilon$. Now consider a random variable $\bar{\mathcal{M}}$ that is geometrically distributed with a success probability of $1-(1-\psi)^{n-1}$. It is not difficult to see that $\bar{\mathcal{M}}$ dominates $\mathcal{M}_\varepsilon$ stochastically (i.e., $\bar{\mathcal{M}} \geq_{st} \mathcal{M}_\varepsilon$), and because $\psi > 0$, it follows that $\hat{E}(\mathcal{M}_\varepsilon) \leq \hat{E}(\bar{\mathcal{M}}) = \frac{1}{1-(1-\psi)^{n-1}} < \infty$. □

Note that for a discrete finite action space, Assumption 3 in Theorem 3.8 is automatically satisfied, and Assumption 2 also holds trivially if we take $\mathcal{P}_x(a) > 0$ for all actions $a \in A(x)$. Furthermore, when the action space is finite, there always exists an $\varepsilon > 0$ such that the only ε-optimal policy is the optimal policy itself. We have the following stronger convergence result for ERPS when the action space is finite.

Corollary 3.9 (Discrete Finite Action Space) *If the action space is finite, $q_0 < 1$, and the action selection distribution $\mathcal{P}_x(a) > 0 \ \forall a \in A(x), \ \forall x \in X$, then there exists a random variable $\mathcal{M} > 0$ with $\hat{E}(\mathcal{M}) < \infty$ such that $V^{\hat{\pi}^k} = V^{\pi^*} \ \forall k \geq \mathcal{M}$.*

3.3 Numerical Examples

In this section, we consider two discrete-time controlled queueing problems and compare the performance of ERPS, EPI, and standard PI. For the size of state space considered, VI would not be competitive. For ERPS, the same search range parameter is prescribed for all states, denoted by a single variable r, and since in both problems, all actions $a \in A$ are admissible for all states $x \in X$, the action selection distribution is chosen to be the uniform distribution over the action space A for all states $x \in X$. We therefore drop the explicit display of x in $\mathcal{P}_x$, and simply use $\mathcal{P}$ instead of $\mathcal{P}_x$. All computations were performed on an IBM PC with a 2.4 GHz Pentium 4 processor and 512 MB memory, and the computational time units are in seconds.

3.3.1 A One-Dimensional Queueing Example

We consider a finite-capacity single-server queue with controlled service completion probabilities. Assume that a server can serve only one customer in a period, and the service of a customer begins/ends only at the beginning/end of any period. Customers arrive at a queue independently with probability $p = 0.2$, and there is at most one arrival per period (so no arrival with probability 0.8). The maximum queue length is $\mathcal{L}$, and an arrival that finds $\mathcal{L}$ customers in the queue is lost. We let x_t, the state variable, be the number of customers in the system at the beginning of period t. The action to be chosen at each state is the service completion probability a, which takes value in a set A. In period t, a possible service completion is generated with probability $a(x_t)$, which results in a transition to state x_{t+1} a cost of $R(x_t, a(x_t))$

being incurred. The goal is to choose the optimal service completion probability for each state such that the total discounted cost $E[\sum_{t=0}^{\infty} \gamma^t R(x_t, a(x_t))]$ is minimized (since R is a cost function here, the definitions in the previous sections need to be replaced with minimizing operators).

3.3.1.1 Discrete Action Space

Two different choices of one-stage cost functions are considered: (i) a simple cost function that is convex in both state and action; (ii) a complicated non-convex cost function. The MDP problem resulting from case (i) may possess some nice properties (e.g., free of multiple local optimal solutions), so finding an optimal solution should be a relatively easy task, whereas the cost function in case (ii) introduces some further computational difficulties (e.g., multiple local minima), intended to more fully test the effectiveness of a global algorithm like ERPS. For both cases, unless otherwise specified, the following parameter settings are used: maximum queue length $\mathcal{L} = 49$; state space $X = \{0, 1, 2, \ldots, 49\}$; discount factor $\gamma = 0.98$; action set $A = \{10^{-4}k : \ k = 0, 1, \ldots, 10^4\}$; and in ERPS, population size $n = 10$, search range $r = 10$, and the standard Euclidean distance is used to define the neighborhood. All results for ERPS are based on 30 independent replications.

For case (i), the one-stage cost at any period for being in state x and taking action a is given by

$$R(x, a) = x + 50a^2. \tag{3.17}$$

We test the convergence of ERPS by varying the values of the exploitation probability. Table 3.1 compares the performance of the algorithm in terms of both CPU time and the following performance measure:

$$\text{relative error} := \max_{x \in X} \frac{|V(x) - V^*(x)|}{|V^*(x)|}, \tag{3.18}$$

which signifies the maximum relative deviation of the value function V from the optimal value function V^*. The computational time required for PI to find V^* was 15 seconds, and the value of $\|V^*\|_\infty$ is approximately 2.32e+03. Test results indicate superior performance of ERPS over PI; in particular, for the cases ($q_0 = 0.25, \ K = 32$), ($q_0 = 0.5, \ K = 16$), and ($q_0 = 0.75, \ K = 16$), ERPS attains the optimal solutions in all 30 independent trials within 2 seconds. Moreover, we see that the algorithm performs quite well even when $q_0 = 0$, which corresponds to pure random search from the action selection point of view. We believe that this is because that ERPS (under $q_0 = 0$) will differ from a pure random search algorithm in the space of policies, in that ERPS is a population-based approach and it contains a PIRS step which tends to search the policy space induced by the population of policies, whereas a pure random search algorithm merely compares the performances of all sampled policies and then simply takes the best one.

To explore the computational complexity of ERPS, tests were performed on MDPs with increasing numbers of actions; for each problem, the foregoing setting

Table 3.1 Convergence results for ERPS ($n = 10,\ r = 10$) based on 30 independent replications (standard errors in parentheses), where K is the stopping rule parameter

q_0	K	CPU time	Relative error
0.0	2	0.84 (0.03)	3.98e–05 (8.20e–06)
	4	1.42 (0.05)	1.34e–05 (2.43e–06)
	8	2.63 (0.10)	4.14e–06 (8.58e–07)
	16	5.20 (0.16)	7.64e–07 (1.07e–07)
	32	8.96 (0.38)	2.72e–07 (3.17e–08)
0.25	2	0.94 (0.02)	1.19e–08 (4.46e–09)
	4	1.09 (0.02)	4.09e–09 (2.04e–09)
	8	1.24 (0.02)	7.94e–10 (2.88e–10)
	16	1.54 (0.03)	4.87e–11 (3.91e–11)
	32	1.85 (0.04)	0.00e–00 (0.00e–00)
0.50	2	0.92 (0.02)	2.10e–08 (1.51e–08)
	4	1.02 (0.02)	1.50e–09 (8.52e–10)
	8	1.13 (0.02)	5.95e–10 (5.03e–10)
	16	1.27 (0.03)	0.00e–00 (0.00e–00)
0.75	2	1.14 (0.02)	2.79e–09 (2.53e–09)
	4	1.20 (0.02)	5.59e–11 (3.97e–11)
	8	1.27 (0.02)	3.38e–11 (3.38e–11)
	16	1.43 (0.03)	0.00e–00 (0.00e–00)
1.0	2	12.13 (0.02)	1.92e–10 (5.97e–11)
	4	12.17 (0.02)	5.60e–11 (4.00e–11)
	8	12.27 (0.01)	0.00e–00 (0.00e–00)

is used except that the action space now takes the form $A = \{hk :\ k = 0, 1, \ldots, \frac{1}{h}\}$, where h is the mesh size, selected sequentially (one for each problem) from the set $\{\frac{1}{100}, \frac{1}{250}, \frac{1}{500}, \frac{1}{1000}, \frac{1}{2500}, \frac{1}{5000}, \frac{1}{10000}, \frac{1}{25000}, \frac{1}{50000}, \frac{1}{100000}, \frac{1}{200000}\}$.

In Fig. 3.4, we plot the running time required for PI and ERPS to find the optimal solutions as a function of the number of actions of each MDP considered, where the results for ERPS are the averaged time over 30 independent replications. Empirical results indicate that the computational time for PI increases linearly in the number of actions (note the log scale used in Fig. 3.4), while the running time required for ERPS does so in an asymptotic sense. We see that ERPS delivers very competitive performances even when the action space is small; when the action space is relatively large (number of actions greater than 10^4), ERPS reduces the computational efforts of PI by well over an order of magnitude. In the experiments, we used a search range $r = 10$ in ERPS, regardless of the size of the action space; we believe the performance of the algorithm could be enhanced by using a search range that is proportional to the size of the action space. Moreover, the computational effort of

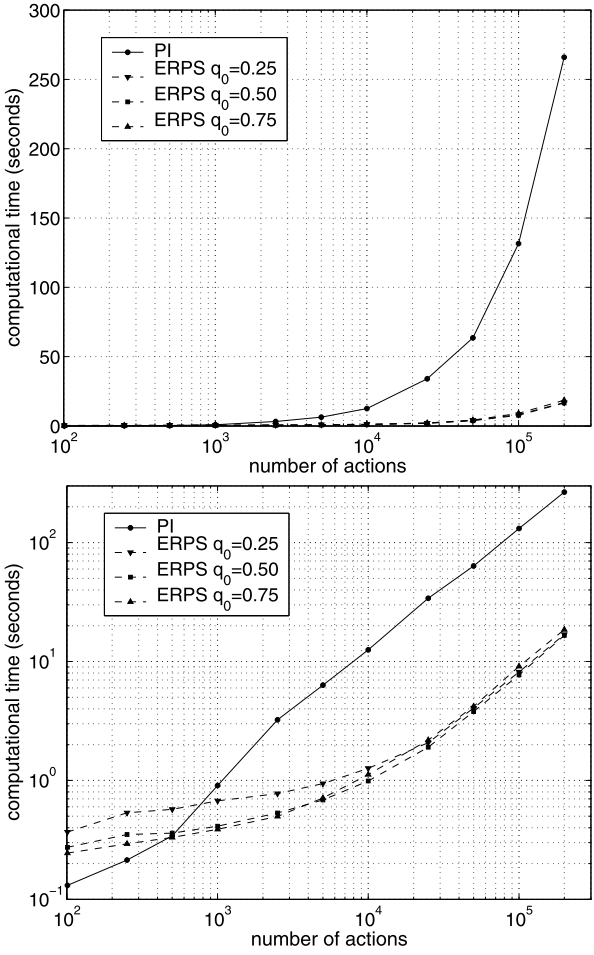


Fig. 3.4 Running time required for PI and ERPS ($n = 10,\ r = 10$, based on 30 independent replications) to find the optimal solutions to MDPs with different numbers of actions, (**a**) using log scale for horizontal axis; (**b**) using log–log plot

ERPS can be reduced considerably if we are merely seeking solutions within some required accuracy rather than insisting on the optimal solution.

For case (ii), we used the following one-stage cost function:

$$R(x, a) = x + 5\left[\frac{|X|}{2}\sin(2\pi a) - x\right]^2, \tag{3.19}$$

which induces a tradeoff in choosing between large values of a to reduce the state x and appropriate values of a to make the squared term small. Moreover, since the sine function is not monotone, the resultant MDP problem has a very high number of local minima; some typical locally optimal policies are shown in Fig. 3.5.

Table 3.2 shows the convergence properties of EPI and ERPS, where both algorithms start with the same initial population. The computational time required for PI to find the optimal value function V^* was 14 seconds. For EPI, we have tested

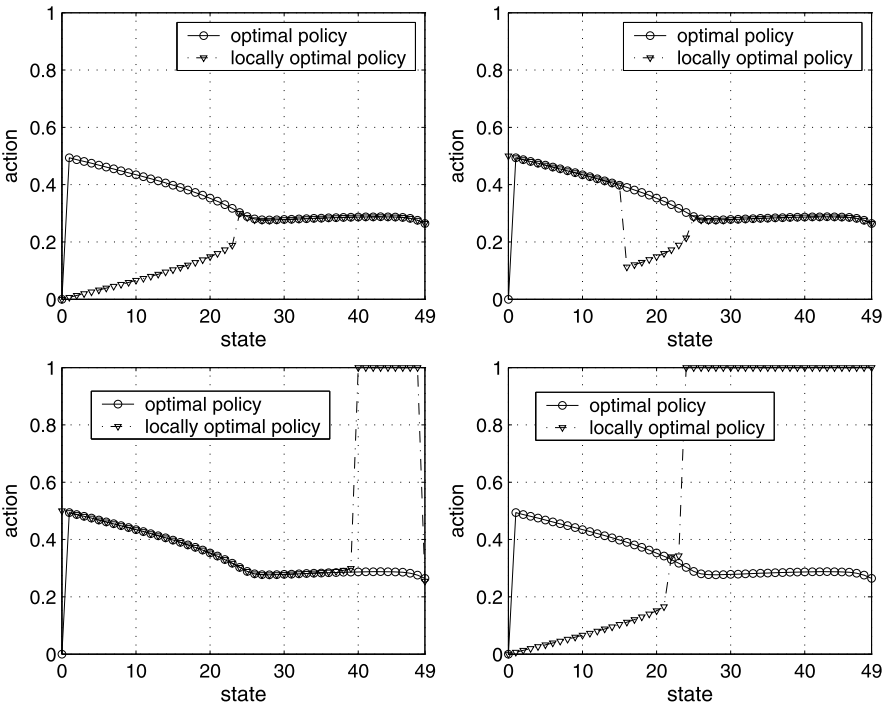


Fig. 3.5 Four typical locally optimal solutions to the test problem

different sets of values for the overall mutation probability q_0 and the global and local mutation probabilities P_g and P_l, respectively; the results reported in Table 3.2 are the best results obtained. Also note that because of the slow convergence of EPI, the values for the stopping control parameter K are chosen much larger than those for ERPS. The typical performances of ERPS and EPI are given in Fig. 3.6, where we have plotted the corresponding value functions of the generated elite policies for some particular iterations.

In order to demonstrate the role of the exploitation probability q_0 in the ERPS algorithm, we fix the stopping control parameter $K = 10$ and vary q_0. The numerical results are recorded in Table 3.3, where N_{opt} indicates the number of times an optimal solution was found out of 30 trials. The $q_0 = 1.0$ case corresponds to pure local search. Obviously in this case, the algorithm gets trapped into a local minimum, which has a mean maximum relative deviation of 1.35e+01. However, note that the standard error is very small, which means that the local minimum is estimated with very high precision. This shows that the "nearest neighbor" heuristic is indeed useful in fine-tuning the solutions. In contrast, the pure random search ($q_0 = 0$) case is helpful in avoiding the local minima, yielding a lower mean relative deviation of 2.42e–2, but it is not very good in locating the exact optimal solutions, as none was found out of 30 trials. Roughly, increasing q_0 between 0 and 0.5 leads to a more accurate estimation of the optimal solution; however, increasing q_0 on the range 0.6

Table 3.2 Convergence results for EPI ($n = 10$) and ERPS ($n = 10,\ r = 10$) based on 30 independent replications (standard errors in parentheses), where K is the stopping rule parameter

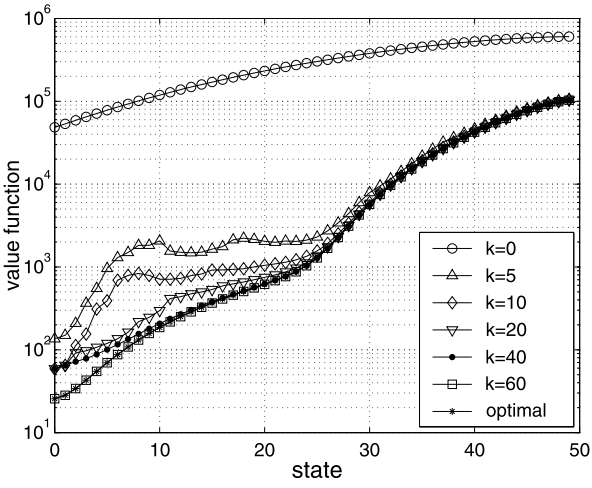
Algorithm	K	CPU time	Relative error
EPI $q_0 = 0.9$ $P_g = 0.9$ $P_l = 0.1$	20	2.13 (0.11)	3.48e–00 (3.16e–01)
	40	3.82 (0.17)	1.55e–00 (1.73e–01)
	80	6.83 (0.35)	8.34e–01 (8.57e–02)
	160	17.03 (0.61)	1.65e–01 (1.83e–02)
ERPS $q_0 = 0.5$ $r = 10$	2	1.03 (0.02)	1.42e–01 (7.95e–02)
	4	1.12 (0.03)	8.64e–02 (6.01e–02)
	8	1.29 (0.03)	4.32e–02 (4.32e–02)
	16	1.49 (0.03)	2.25e–07 (1.36e–07)
	32	1.86 (0.04)	0.00e–00 (0.00e–00)

Table 3.3 Performance of ERPS with different exploitation probabilities ($n = 10,\ K = 10,\ r = 10$) based on 30 independent replications (standard errors in parentheses)

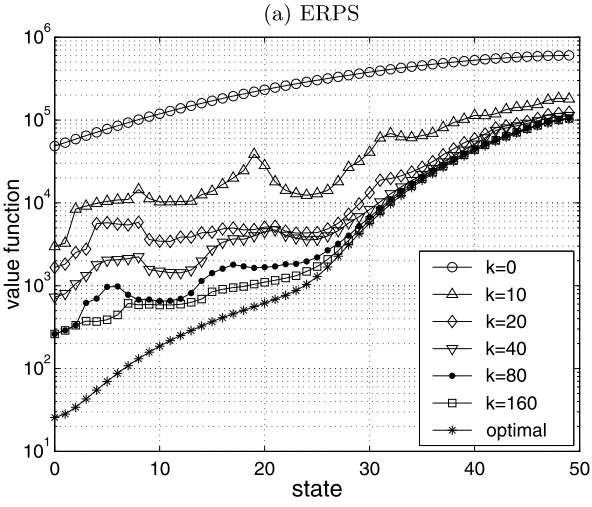
q_0	CPU time	N_{opt}	Relative error
0.0	3.47 (0.14)	0	2.42e–02 (6.04e–03)
0.1	2.04 (0.04)	6	5.82e–05 (4.42e–05)
0.2	1.48 (0.03)	14	8.75e–06 (4.76e–06)
0.3	1.36 (0.02)	23	1.78e–07 (1.08e–07)
0.4	1.28 (0.03)	22	3.25e–06 (2.56e–06)
0.5	1.32 (0.03)	26	2.44e–06 (2.32e–06)
0.6	1.43 (0.04)	26	1.67e–01 (9.77e–02)
0.7	1.47 (0.04)	24	2.08e–01 (8.97e–02)
0.8	1.80 (0.04)	20	5.49e–01 (1.51e–01)
0.9	2.28 (0.08)	8	1.19e–00 (1.89e–01)
1.0	8.90 (0.02)	0	1.35e+01 (3.30e–16)

to 1.0 decreases the quality of the solution, because the local search part begins to gradually dominate, so that the algorithm is more easily trapped in local minima. This also explains why we have larger variances when $q_0 = 0.6, 0.7, 0.8, 0.9$ in Table 3.3. Notice that the algorithm is very slow in the pure local search case; setting $q_0 < 1$ speeds up the algorithm substantially.

To provide a numerical comparison between the "nearest neighbor" heuristic (biased sampling) and the policy mutation procedure (unbiased sampling), we construct a new algorithm that uses the PIRS step to generate the elite policy from the current population of policies but the policy mutation procedure (as in EPI) to generate the remaining policies in the next population. Denote this new algorithm by PIRS+PM. In both ERPS and PIRS+PM, we fix the population size $n = 10$, and stop the algorithms only when a desired accuracy is reached. In Table 3.4, we record the length of time required for different algorithms to reach a relative deviation of at least 1.0e–03. Indeed, we see that ERPS uses far less time to reach a required accuracy than PIRS+PM.



(a) ERPS



(b) EPI

Fig. 3.6 Convergence of the value function for case (ii), where k is the iteration counter: (**a**) ERPS ($n = 10$, $r = 10$, $K = 16$, $q_0 = 0.5$); (**b**) EPI ($n = 10$, $K = 160$, $q_0 = 0.9$, $P_g = 0.9$, $P_l = 0.1$)

3.3.1.2 Continuous Action Space

We test the algorithm when the action space A is continuous, where the service completion probability can be any value between 0 and 1. Again, two cost functions are considered, corresponding to cases (i) and (ii) in Sect. 3.3.1.1. In both cases, the maximum queue length $\mathcal{L}$, state space X, and the discount factor γ are all taken to be the same as before.

In the numerical experiments, we approximated the optimal costs V_1^* and V_2^* for each of the respective cases (i) and (ii) by two value functions $\hat{V}_1^*$ and $\hat{V}_2^*$, which were computed by using a discretization-based policy iteration (PI) algorithm, where we first uniformly discretize the action space into evenly spaced points by using a mesh size h, and then apply the standard PI algorithm on the discretized

Table 3.4 Average time required to reach a precision of at least 1.0e–6 for different algorithms, based on 30 independent replications (standard errors in parentheses)

Algorithm	Parameters	CPU time	Relative error
ERPS $r = 10$	$q_0 = 0.0$	14.34 (1.68)	5.01e–04 (4.59e–05)
	$q_0 = 0.1$	1.05 (0.02)	4.28e–04 (5.29e–05)
	$q_0 = 0.3$	0.91 (0.04)	4.04e–04 (5.77e–05)
	$q_0 = 0.5$	0.94 (0.04)	4.36e–04 (6.01e–05)
	$q_0 = 0.7$	1.63 (0.18)	3.06e–04 (5.59e–05)
	$q_0 = 0.9$	4.10 (0.64)	2.12e–04 (4.27e–05)
PIRS+PM	$q_0 = 0.9,\ P_g = 0.9,\ P_l = 0.1$	66.6 (9.8)	5.19e–04 (5.30e–05)
	$q_0 = 0.7,\ P_g = 0.9,\ P_l = 0.1$	39.1 (6.6)	5.60e–04 (5.19e–05)
	$q_0 = 0.5,\ P_g = 0.9,\ P_l = 0.1$	21.7 (1.8)	6.14e–04 (4.42e–05)
	$q_0 = 0.3,\ P_g = 0.9,\ P_l = 0.1$	23.4 (3.1)	4.85e–04 (5.77e–05)
	$q_0 = 0.1,\ P_g = 0.9,\ P_l = 0.1$	21.1 (2.9)	5.81e–04 (5.78e–05)
	$q_0 = 0.0,\ P_g = 1.0,\ P_l = 0.0$	23.7 (2.7)	4.49e–04 (5.71e–05)

problem. In both cases, we take $h = 1e{-}8$. Note that a brute-force calculation of $\hat{V}_1^*$ or $\hat{V}_2^*$ requires more than 40 hours of CPU time.

We set the population size $n = 10$ and termination control parameter $K = 10$, and test the ERPS algorithm by using different values of the search range r. The performance of the algorithm is also compared with that of the discretization-based PI algorithm. Tables 3.5 and 3.6 give the performances of both algorithms for cases (i) and (ii), respectively. Note that the relative deviations are actually computed by replacing the optimal value functions V_1^* and V_2^* with their corresponding approximations $\hat{V}_1^*$ and $\hat{V}_2^*$ in Eq. (3.18).

Test results indicate that ERPS outperforms the discretization-based PI algorithm in both cases, not only in computational time but also in solution quality. We observe that the computational time for PI increases by a factor of 2 for each halving of the mesh size, while the time for ERPS increases at a much slower rate.

3.3.2 A Two-Dimensional Queueing Example

The second example, shown in Fig. 3.7, is a slight modification of the first one, with the difference being that now we have a single queue that feeds two independent servers with different service completion probabilities a_1 and a_2. We consider only the continuous action space case. The action to be chosen at each state x is $(a_1, a_2)^T$, which takes value from the set $A = [0, 1] \times [0, 1]$. We assume that an arrival that finds the system empty will always be served by the server with service completion probability a_1. The state space of this problem is $X = \{0, 1_{S_1}, 1_{S_2}, 2, \ldots, 48\}$, where we have assumed that the maximum queue length (not including those in service) is

Table 3.5 Comparison of the ERPS algorithm ($n = 10,\ K = 10$) with the deterministic PI algorithm for case (i), where the results of ERPS are based on 30 independent replications (standard errors in parentheses)

Algorithm	Parameters	CPU time	Relative error
ERPS ($r = \frac{1}{4000}$)	$q_0 = 0.25$	2.66 (0.10)	1.12e–11 (3.72e–12)
	$q_0 = 0.50$	2.27 (0.09)	2.86e–12 (4.20e–13)
	$q_0 = 0.75$	2.94 (0.08)	1.11e–12 (2.51e–13)
ERPS ($r = \frac{1}{8000}$)	$q_0 = 0.25$	2.63 (0.10)	2.87e–12 (5.62e–13)
	$q_0 = 0.50$	2.93 (0.10)	6.12e–13 (1.49e–13)
	$q_0 = 0.75$	3.10 (0.11)	3.94e–13 (7.02e–14)
ERPS ($r = \frac{1}{16000}$)	$q_0 = 0.25$	2.85 (0.09)	8.80e–13 (2.45e–13)
	$q_0 = 0.50$	3.27 (0.10)	1.87e–13 (3.85e–14)
	$q_0 = 0.75$	3.72 (0.10)	9.91e–14 (2.34e–14)
PI	$h = \frac{1}{4000}$	6 (N/A)	2.55e–08 (N/A)
	$h = \frac{1}{8000}$	12 (N/A)	1.35e–08 (N/A)
	$h = \frac{1}{16000}$	23 (N/A)	5.04e–09 (N/A)
	$h = \frac{1}{32000}$	46 (N/A)	5.84e–10 (N/A)
	$h = \frac{1}{128000}$	188 (N/A)	3.90e–11 (N/A)
	$h = \frac{1}{512000}$	793 (N/A)	3.83e–12 (N/A)

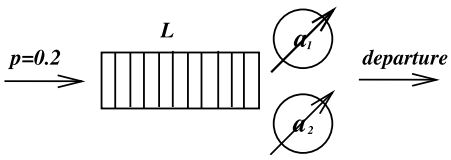


Fig. 3.7 A two-dimensional queueing example

46, and $1_{S_1}, 1_{S_2}$ are used to distinguish the situations whether server 1 or server 2 is busy when there is only one customer in the system. As before, the discount factor $\alpha = 0.98$.

The one-stage cost is taken to be

$$R(y, a_1, a_2) = y + \left[\frac{|X|}{2}\cos(\pi a_1) - y\right]^2 I_{\{S_1\}} + \left[\frac{|X|}{2}\sin(\pi a_2) - y\right]^2 I_{\{S_2\}}, \quad (3.20)$$

where

$$I_{\{S_i\}} = \begin{cases} 1 & \text{if server } i \text{ is busy,} \\ 0 & \text{otherwise} \end{cases} \quad (i = 1, 2), \quad \text{and} \quad y = \begin{cases} 1 & \text{if } x \in \{1_{S_1}, 1_{S_2}\}, \\ x & \text{otherwise.} \end{cases}$$

The performances of the ERPS and the discretization-based PI are reported in Table 3.7. In ERPS, both the population size n and the stopping control parame-

Table 3.6 Comparison of the ERPS algorithm ($n = 10,\ K = 10$) with the deterministic PI algorithm for case (ii), where the results of ERPS are based on 30 independent replications (standard errors in parentheses)

Algorithm	Parameters	CPU time	Relative error
ERPS ($r = \frac{1}{4000}$)	$q_0 = 0.25$	2.74 (0.10)	1.09e–07 (3.24e–08)
	$q_0 = 0.50$	2.86 (0.08)	2.19e–08 (6.15e–09)
	$q_0 = 0.75$	3.13 (0.09)	7.69e–09 (1.36e–09)
ERPS ($r = \frac{1}{8000}$)	$q_0 = 0.25$	3.06 (0.12)	1.47e–08 (3.61e–09)
	$q_0 = 0.50$	2.98 (0.13)	4.55e–09 (9.77e–10)
	$q_0 = 0.75$	3.57 (0.08)	1.76e–09 (4.21e–10)
ERPS ($r = \frac{1}{16000}$)	$q_0 = 0.25$	3.17 (0.09)	9.50e–09 (3.55e–09)
	$q_0 = 0.50$	3.26 (0.11)	1.42e–09 (2.44e–10)
	$q_0 = 0.75$	4.17 (0.12)	3.49e–10 (7.70e–11)
PI	$h = \frac{1}{4000}$	5 (N/A)	8.35e–04 (N/A)
	$h = \frac{1}{8000}$	11 (N/A)	4.51e–05 (N/A)
	$h = \frac{1}{16000}$	21 (N/A)	4.50e–05 (N/A)
	$h = \frac{1}{32000}$	42 (N/A)	9.66e–06 (N/A)
	$h = \frac{1}{128000}$	175 (N/A)	8.96e–07 (N/A)
	$h = \frac{1}{512000}$	734 (N/A)	2.34e–08 (N/A)

Table 3.7 A two-dimensional test example, where the ERPS results ($n = 10,\ K = 10$) are based on 30 independent replications (standard errors in parentheses)

Algorithm	Parameters	CPU time	Relative error
ERPS ($r = \frac{1}{100}$)	$q_0 = 0.25$	3.18 (0.15)	3.86e–04 (3.18e–05)
	$q_0 = 0.50$	3.16 (0.16)	7.48e–03 (7.25e–03)
	$q_0 = 0.75$	3.54 (0.14)	5.83e–02 (1.78e–02)
ERPS ($r = \frac{1}{200}$)	$q_0 = 0.25$	3.31 (0.12)	9.44e–05 (8.25e–06)
	$q_0 = 0.50$	3.26 (0.12)	7.31e–03 (7.27e–03)
	$q_0 = 0.75$	3.88 (0.17)	5.48e–02 (1.83e–02)
ERPS ($r = \frac{1}{400}$)	$q_0 = 0.25$	3.53 (0.12)	2.06e–05 (1.97e–06)
	$q_0 = 0.50$	3.74 (0.12)	7.27e–03 (7.26e–03)
	$q_0 = 0.75$	4.36 (0.14)	3.55e–02 (1.48e–02)
PI	$h = \frac{1}{100}$	14 (N/A)	6.23e–02 (N/A)
	$h = \frac{1}{200}$	55 (N/A)	2.98e–02 (N/A)
	$h = \frac{1}{400}$	226 (N/A)	1.24e–03 (N/A)

ter K are set to 10. In PI, we adopt a uniform discretization, where the same mesh size h is used in both coordinates of the action space. Again, in computing the

relative deviation, we approximated V^* by $\hat{V}^*$, which was computed by using the discretization-based PI algorithm with a mesh size $h = \frac{1}{15000}$. Notice that the computational time for PI increases by a factor of 4 for each halving of the mesh size, whereas the time required by ERPS increases much more slowly.

These preliminary numerical experiments indicate some robustness with respect to the choice of the exploitation probability q_0, which balances exploitation and exploration in action selections, in that values between 0.25 and 0.75 all seem to work well. Another approach is to follow a similar strategy as in simulated annealing whereby the value of q_0 is gradually increased from 0 to 1, which corresponds to the transitioning of the search mechanism from pure random sampling to pure local search. The numerical results also demonstrate the potential for orders of magnitude computational efficiency gains over traditional policy iteration on a select set of test cases.

An important implementation issue is the dependence of ERPS on the underlying distance metric, as determining a good metric could be challenging for those problems that do not have a natural metric already available. One possible way to circumvent this is to adaptively update/change the action selection distribution $\mathcal{P}$ at each iteration of the algorithm based on the sampling information obtained during the previous iterations in such a way that more promising policies will have a larger chance of being selected. Using the model reference adaptive search (MRAS) framework of Chap. 4, this approach will be developed in Sect. 4.4.

3.4 Extension to Simulation-Based Setting

In this chapter, the key steps of policy switching in EPI, given by (3.2) and (3.3), and PIRS in ERPS, given by (3.5), are based on the computation of the exact values of the infinite-horizon value function $V^\pi(x), x \in X, \pi \in \Pi_s$, which essentially requires solving a system of linear equations. For large state spaces, this may be impractical. Furthermore, in the predominant setting of this book, the transition probabilities needed to solve the system of equations may not be explicitly available; instead, we have a simulation model, i.e., $(X, A, A(\cdot), f, R')$ instead of $(X, A, A(\cdot), P, R)$.

In this simulation-based setting, we first have to approximate the infinite-horizon value function by a finite-horizon value function:

$$V^\pi(x) = E\left[\sum_{t=0}^{H-1} \gamma^t R'\big(x_t, \pi(x_t), w_t\big) \middle| x_0 = x\right],$$

selecting an appropriate horizon length H. Simulation is then applied to estimate this value function, which can then be used in (3.2) and (3.3) for EPI, and in (3.5) for ERPS, respectively. Unfortunately, the resulting computational complexity still depends on the size of the state space, since the operations defined by (3.2), (3.3), and (3.5) all require computation of the policy over the *entire* state space.

In Chap. 5, we consider "on-line" versions of policy switching and PIRS, i.e., policies in which the system (either the actual system itself or a simulation model of the system) evolves to a particular state that is observed, and the action to be taken in that particular state is then computed on-line at the decision time via the methods of policy switching and PIRS.

3.5 Notes

EPI and ERPS were introduced in [43] and [95], respectively, where in the latter case, the PIRS step was called "policy improvement with cost swapping" (PICS). The literature applying evolutionary algorithms such as GAs for solving MDPs is relatively sparse. Reference [121] uses a GA approach to construct the minimal set of affine functions that describes the value function in partially observable MDPs, yielding a variant of value iteration (VI). Reference [46] proposes an approach that maps heuristically "simple" GA [169] into the framework of PI. Unfortunately, the convergence to an optimal policy is not always guaranteed. Some other work includes [182], where GAs are used to find good finite-horizon policies for partially observable MDPs, and [8], where a genetic search in policy space similar to [46] for solving infinite-horizon discounted MDPs is proposed with no convergence guarantee. Chang [33] applies "marriage in honey-bees optimization" with a similar framework to EPI.

A concept of elitism related to our work was introduced by De Jong [53]. However, in his work, the elitist is a best policy in the *current* population, whereas in EPI and ERPS, the elite policy may not be the best policy in the current population, but is guaranteed to be better than all policies in the *previous* population, although it may not be a member of the previous population. Policy switching was introduced in [41], although the operation of improving upon two given policies via (3.1) can also be found in [148, p. 152].

The issue of large action spaces was addressed in early work by MacQueen [124], who used some inequality forms of Bellman's equation together with bounds on the optimal value function to identify and eliminate non-optimal actions in order to reduce the size of the action sets to be searched at each iteration of the algorithm. Since then, the procedure has been applied to several standard methods like policy iteration (PI), value iteration (VI), and modified policy iteration (see Chap. 6 in [145] for a review). All of these algorithms generally require a finite action space. Perhaps the most straightforward and the most commonly used numerical approach in dealing with MDPs with uncountable action spaces is via the use of discretization (see [154]). In practice, this could lead to computational difficulties, either resulting in an action space that is too large or in a solution that is not accurate enough.

The queueing examples used in the numerical experiments were adapted from [52].

Chapter 4
Model Reference Adaptive Search

In this chapter, we present a general global optimization method called model reference adaptive search (MRAS), and explore some of its applications for solving MDPs. We start by introducing the MRAS method in a deterministic optimization context, where the performance function can be evaluated exactly. Then we generalize the method to stochastic settings where the performance function can only be estimated with some noise, e.g., via simulation or real-time observation. MRAS can be applied either directly or indirectly for solving MDPs. In the former case, we use the method as a particular policy learning approach to find the best policy within a class of parameterized policies for both finite- and infinite-horizon MDPs, whereas in the latter case, we combine the method with the ERPS algorithm introduced in Chap. 3 to provide another population-based MDP solution technique with balanced exploration and exploitation. We also discuss an approach in which MRAS can be used as another sampling mechanism in the adaptive multi-stage sampling approach of Chap. 2. Finally, at the end of the chapter, we present a recently developed stochastic approximation framework for studying a class of simulation- and sampling-based optimization algorithms. We illustrate the framework through an exemplary algorithm instantiation called model-based annealing random search (MARS) and discuss its application to finite-horizon MDPs. The MARS algorithm can also be used as an on-line simulation-based approach for solving infinite-horizon MDPs, which will be discussed in Chap. 5.

The MRAS approach falls into the class of *model-based* methods for global optimization, whereby new solutions are generated via an intermediate *probability model* that is updated or induced from the previous solutions (see Fig. 4.1). Each iteration of these algorithms usually involves the following two phases:

1. Generate/sample candidate solutions (random samples, trajectories) according to a specified probability distribution model.
2. Update the probabilistic model, on the basis of the solutions generated in the first phase, in order to bias the future search toward "better" solutions.

The idea is to concentrate the probability mass of the distribution model on the set of promising solutions, so that good solutions will be sampled with high probabil-

H.S. Chang et al., *Simulation-Based Algorithms for Markov Decision Processes*,
Communications and Control Engineering,
DOI 10.1007/978-1-4471-5022-0_4, © Springer-Verlag London 2013

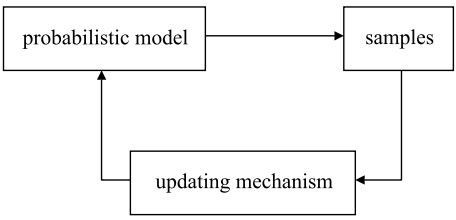


Fig. 4.1 Depiction of two phases iterated in model-based methods

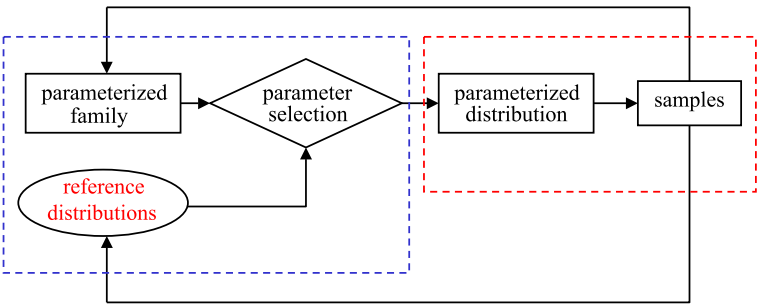


Fig. 4.2 Depiction of iterative procedure in the MRAS method

ity. Throughout, unless otherwise indicated, a "distribution" (function) will mean a probability density/mass function (p.d.f./p.m.f.), covering both the continuous and the discrete cases, with integrals understood to be replaced by summations in the latter case.

The theoretical and practical efficiencies of model-based methods are primarily determined by the two key questions of how to update these probabilistic models and how to efficiently generate samples from them. In MRAS, these difficulties are circumvented by sampling from a family of *parameterized* distributions and using a *sequence* of intermediate *reference* models to facilitate and guide the updating of the parameters associated with the family of parameterized distributions. A schematic description of MRAS is given in Fig. 4.2. One hopes that the parameterized family is specified with some structure so that, once the parameter is determined, sampling from each of these distributions should be a relatively easy task. For example, for optimization problems in $\Re^n$, one possible choice of the parameterized distribution is the multivariate normal (Gaussian) distribution, which can be efficiently sampled from and represented relatively compactly by its mean vector and covariance matrix. An additional advantage of using the parameterized family is that the task of updating the entire sampling distribution now simplifies to the task of updating its associated parameters, which is carried out by minimizing a certain distance between the parameterized family and the *reference* distributions. The sequence of reference distributions in MRAS is primarily used to guide the parameter updating process and to express the desired properties of the method. Thus, these distributions are often selected so that they can be shown to converge to a degenerate distribution with all probability mass concentrated on the set of optimal solutions. Intuitively, the

sampling distribution can be viewed as a compact approximation of the reference distribution (i.e., the projection of the reference distribution on the parameterized family). Thus, as the sequence of reference distributions converges, the sequence of samples generated from their compact approximations (i.e., sampling distributions) should also converge to the optimum.

This chapter is organized as follows. The general MRAS approach for global optimization—both deterministic and stochastic—is presented in Sect. 4.1, followed by convergence analysis in Sect. 4.2. The two main applications of MRAS in the MDP setting are presented in Sects. 4.3 and 4.4, which include numerical results for the algorithms, whereas Sect. 4.5 contains a brief discussion of a proposed algorithm for applying MRAS to the adaptive sampling framework of Chap. 2. In Sect. 4.6, we present a systematic framework based on stochastic approximation theory that allows us to study a class of randomized optimization algorithms in a uniform manner. A particular algorithm instantiation of the framework called model-based annealing random search is presented in Sect. 4.6.1 and its application to the MDP setting is discussed in Sect. 4.6.2. With the exception of the requisite Sect. 4.1, each of the sections in this chapter can be read independently. In particular, for those readers interested primarily in the practical implementation of the MRAS algorithms to the various MDP settings rather than in the mathematics guaranteeing theoretical convergence, the technical proofs of Sect. 4.2 can be skipped without loss of continuity.

4.1 The Model Reference Adaptive Search Method

We consider the following optimization problem:

$$\mathbf{x}^* \in \arg\max_{\mathbf{x}\in\mathcal{X}} J(\mathbf{x}), \quad \mathbf{x} \in \mathcal{X} \subseteq \Re^n, \tag{4.1}$$

where $\mathcal{X}$ is a non-empty set in $\Re^n$, and $J : \mathcal{X} \to \Re$ is a deterministic function that is bounded from below, i.e., $\exists \mathcal{M} > -\infty$ such that $J(\mathbf{x}) \geq \mathcal{M}\ \forall \mathbf{x} \in \mathcal{X}$. Throughout this chapter, we assume that (4.1) has a unique global optimal solution, i.e., $\exists \mathbf{x}^* \in \mathcal{X}$ such that $J(\mathbf{x}) < J(\mathbf{x}^*)\ \forall \mathbf{x} \neq \mathbf{x}^*,\ \mathbf{x} \in \mathcal{X}$.

MRAS works with a family of parameterized distributions $\{f(\cdot, \theta),\ \theta \in \Theta\}$ on the solution space, where Θ is the parameter space. Assume that at the kth iteration of the method, we have a sampling distribution $f(\cdot, \theta_k)$. We generate candidate solutions from this sampling distribution. The performances of these randomly generated solutions are then evaluated and used to calculate a new parameter vector $\theta_{k+1} \in \Theta$ according to a specified parameter updating rule. The above steps are performed repeatedly until a termination criterion is satisfied. The idea is that if the parameter updating rule is chosen appropriately, the future sampling process will be more and more concentrated on regions containing high-quality solutions.

In MRAS, the parameter updating is determined by a sequence of *reference* distributions $\{g_k\}$. At each iteration k, we consider the *projection* of g_k on the family

of distributions $\{f(\cdot,\theta),\ \theta \in \Theta\}$ and compute the new parameter vector θ_{k+1} that minimizes the Kullback–Leibler (KL) divergence

$$\mathcal{D}\big(g_k, f(\cdot,\theta)\big) := E_{g_k}\left[\ln \frac{g_k(\mathbf{X})}{f(\mathbf{X},\theta)}\right] = \int_{\mathcal{X}} \ln \frac{g_k(\mathbf{x})}{f(\mathbf{x},\theta)} g_k(\mathbf{x})\nu(d\mathbf{x}), \tag{4.2}$$

where ν is the Lebesgue/counting measure defined on $\mathcal{X}$, $\mathbf{X} \in \Re^n$ is a random vector taking values in $\mathcal{X}$, and E_{g_k} denotes the expectation taken with respect to g_k. Intuitively, $f(\cdot,\theta_{k+1})$ can be viewed as a compact representation (approximation) of the reference distribution g_k; consequently, the feasibility and effectiveness of the method will, to some large extent, depend on the choices of reference distributions.

There are many different ways to construct the sequence of reference distributions $\{g_k\}$. In this chapter, we use the following simple iterative scheme. Let $g_1(\mathbf{x}) > 0\ \forall \mathbf{x} \in \mathcal{X}$ be an initial distribution on the solution space $\mathcal{X}$. At each iteration $k \geq 1$, a new distribution is computed by tilting the previous distribution $g_{k-1}(\mathbf{x})$ with the performance function $J(\mathbf{x})$ (for simplicity, here we assume $J(\mathbf{x}) > 0\ \forall \mathbf{x} \in \mathcal{X}$), i.e.,

$$g_k(\mathbf{x}) = \frac{J(\mathbf{x}) g_{k-1}(\mathbf{x})}{\int_{\mathcal{X}} J(\mathbf{x}) g_{k-1}(\mathbf{x})\nu(d\mathbf{x})}, \quad \forall \mathbf{x} \in \mathcal{X}. \tag{4.3}$$

By assigning greater weight to solutions having larger values for J, each iteration of Eq. (4.3) improves the expected performance in the sense that

$$E_{g_k}\big[J(\mathbf{X})\big] = \frac{E_{g_{k-1}}[(J(\mathbf{X}))^2]}{E_{g_{k-1}}[J(\mathbf{X})]} \geq E_{g_{k-1}}\big[J(\mathbf{X})\big].$$

Furthermore, it is possible to show that the sequence $\{g_k, k = 1, 2, \ldots\}$ will converge to a distribution that concentrates only on the optimal solution for arbitrary g_1, with $\lim_{k\to\infty} E_{g_k}[J(\mathbf{X})] = J(\mathbf{x}^*)$.

The rest of this section introduces three algorithms:

- MRAS_0: the idealized version of the algorithm, where the objective function J is deterministic and we assume that expectations can be evaluated exactly.
- MRAS_1: applicable version of MRAS_0, where J is deterministic but expectations are estimated by their corresponding sample averages.
- MRAS_2: extension of MRAS_1 to stochastic optimization, where the objective function J cannot be evaluated exactly, but can be estimated with some noise, along with expectations approximated by sample averages.

Only the last two algorithms are of any practical interest, but the first serves as a basic foundation.

4.1.1 The MRAS_0 *Algorithm (Idealized Version)*

Figure 4.3 presents the MRAS_0 algorithm, which is a particular instantiation of MRAS that uses the sequence of reference distributions generated by Eq. (4.3).

Algorithm MRAS$_0$

Input: $\rho \in (0, 1]$, $\varepsilon \geq 0$, strictly increasing function $\mathcal{H} : \Re \to \Re^+$, family of distributions $\{f(\cdot, \theta)\}$, with θ_0 s.t. $f(\mathbf{x}, \theta_0) > 0 \; \forall \mathbf{x} \in \mathcal{X}$.

Initialization: Set iteration count $k = 0$.

Loop until Stopping Rule is satisfied:

- Calculate the $(1 - \rho)$-quantile:

$$\chi_k = \sup_l \{l : P_{\theta_k}(J(\mathbf{X}) \geq l) \geq \rho\}.$$

- Update elite threshold:

$$\bar{\chi}_k = \begin{cases} \chi_k & \text{if } k = 0 \text{ or } \chi_k \geq \bar{\chi}_{k-1} + \varepsilon, \\ \bar{\chi}_{k-1} & \text{otherwise.} \end{cases}$$

- Update parameter vector:

$$\theta_{k+1} \in \arg\max_{\theta \in \Theta} E_{\theta_k} \left[\frac{[\mathcal{H}(J(\mathbf{X}))]^k}{f(\mathbf{X}, \theta_k)} I\{J(\mathbf{X}) \geq \bar{\chi}_k\} \ln f(\mathbf{X}, \theta) \right]. \tag{4.4}$$

- $k \leftarrow k + 1$.

Output: θ_k.

Fig. 4.3 Description of MRAS$_0$ algorithm

Throughout the chapter, we use P_{θ_k} and E_{θ_k} to denote the respective probability and expectation taken with respect to the distribution $f(\cdot, \theta_k)$. Thus, under our notational convention,

$$P_{\theta_k}(J(\mathbf{X}) \geq \chi) = \int_{\mathcal{X}} I\{J(\mathbf{x}) \geq \chi\} f(\mathbf{x}, \theta_k) \nu(d\mathbf{x}),$$

$$E_{\theta_k}[J(\mathbf{X})] = \int_{\mathcal{X}} J(\mathbf{x}) f(\mathbf{x}, \theta_k) \nu(d\mathbf{x}).$$

In MRAS$_0$, only a portion of the samples—the set of elite samples—is used to update the probability model. This is achieved primarily through a quantile estimate of the performance function values of the current samples. In the MRAS$_0$ algorithm, the parameter ρ determines the approximate proportion of samples used to update the probabilistic model. At each iteration k of the algorithm, the $(1 - \rho)$-quantile of the performance function values with respect to the distribution $f(\cdot, \theta_k)$ is calculated. These quantile values $\{\chi_k\}$ are used to construct a sequence of non-decreasing thresholds $\{\bar{\chi}_k\}$, and only those candidate solutions having performances better than these thresholds will be used in the parameter update via (4.4). Intuitively, the primary reason for using the thresholds $\{\bar{\chi}_k\}$ is that such a bootstrapping approach for selecting the elite samples will quickly direct the search of the algorithm towards a sequence of "improving" regions, which could be more efficient than simply using

the sequence of quantile values $\{\chi_k\}$ or even a fixed threshold to determine the elite samples.

During the initialization step of MRAS$_0$, a small number ε and a strictly increasing function $\mathcal{H} : \Re \to \Re^+$ are also specified. The function $\mathcal{H}$ is used to account for the cases where the value of $J(\mathbf{x})$ is negative for some $\mathbf{x}$, and the parameter ε ensures that each strict increment in the sequence $\{\bar{\chi}_k\}$ is lower bounded, i.e.,

$$\inf_{\substack{\bar{\chi}_k \neq \bar{\chi}_{k-1} \\ k=1,2,\ldots}} (\bar{\chi}_k - \bar{\chi}_{k-1}) \geq \varepsilon.$$

We require ε to be strictly positive for continuous problems, and non-negative for discrete (finite) problems.

In continuous domains, the division by $f(\mathbf{x}, \theta_k)$ in the parameter update (4.4) is well defined if $f(\mathbf{x}, \theta_k)$ has infinite support (e.g., normal p.d.f.), whereas in discrete/combinatorial domains, the division is still valid as long as each point $\mathbf{x}$ in the solution space has a positive probability of being sampled. Additional regularity conditions on $f(\mathbf{x}, \theta_k)$ imposed in Sect. 4.2 for the convergence proofs ensure that the parameter update (4.4) can be used interchangeably with the following:

$$\theta_{k+1} \in \arg\max_{\theta \in \Theta} \int_{\mathcal{X}} \big[\mathcal{H}\big(J(\mathbf{x})\big)\big]^k I\big\{J(\mathbf{x}) \geq \bar{\chi}_k\big\} \ln f(\mathbf{x}, \theta) \nu(d\mathbf{x}).$$

The following lemma shows that there is a sequence of reference models $\{g_k,\ k = 1, 2, \ldots\}$ implicit in MRAS$_0$, and that the corresponding parameter updating in MRAS$_0$ indeed minimizes the KL-divergence $\mathcal{D}(g_{k+1}, f(\cdot, \theta))$.

Lemma 4.1 *The parameter θ_{k+1} computed at the kth iteration of the* MRAS$_0$ *algorithm via* (4.4) *minimizes the KL-divergence $\mathcal{D}(g_{k+1}, f(\cdot, \theta))$, where*

$$g_{k+1}(\mathbf{x}) := \frac{\mathcal{H}(J(\mathbf{x}))I\{J(\mathbf{x}) \geq \bar{\chi}_k\} g_k(\mathbf{x})}{E_{g_k}[\mathcal{H}(J(\mathbf{X}))I\{J(\mathbf{X}) \geq \bar{\chi}_k\}]} \quad \forall \mathbf{x} \in \mathcal{X},\ k = 1, 2, \ldots, \quad \textit{and}$$

$$g_1(\mathbf{x}) := \frac{I\{J(\mathbf{x}) \geq \bar{\chi}_0\}}{E_{\theta_0}[\frac{I\{J(\mathbf{X}) \geq \bar{\chi}_0\}}{f(\mathbf{X}, \theta_0)}]}.$$

Proof For notational brevity, define $\hat{\mathcal{H}}_k(J(\mathbf{x})) := \frac{[\mathcal{H}(J(\mathbf{x}))]^k}{f(\mathbf{x}, \theta_k)}$. We have

$$g_1(\mathbf{x}) = \frac{I\{J(\mathbf{x}) \geq \bar{\chi}_0\}}{E_{\theta_0}[\frac{I\{J(\mathbf{X}) \geq \bar{\chi}_0\}}{f(\mathbf{X}, \theta_0)}]} = \frac{I\{J(\mathbf{x}) \geq \bar{\chi}_0\}}{E_{\theta_0}[\hat{\mathcal{H}}_0(J(\mathbf{X}))I\{J(\mathbf{X}) \geq \bar{\chi}_0\}]}.$$

When $k \geq 1$, we have from the definition of g_k above,

$$\begin{aligned} g_2(\mathbf{x}) &= \frac{\mathcal{H}(J(\mathbf{x}))I\{J(\mathbf{x}) \geq \bar{\chi}_1\} g_1(\mathbf{x})}{E_{g_1}[\mathcal{H}(J(\mathbf{X}))I\{J(\mathbf{X}) \geq \bar{\chi}_1\}]} \\ &= \frac{\mathcal{H}(J(\mathbf{x}))I\{J(\mathbf{x}) \geq \bar{\chi}_1\} I\{J(\mathbf{x}) \geq \bar{\chi}_0\}}{E_{\theta_1}[\hat{\mathcal{H}}_1(J(\mathbf{X}))I\{J(\mathbf{X}) \geq \bar{\chi}_1\} I\{J(\mathbf{X}) \geq \bar{\chi}_0\}]} \end{aligned}$$

$$= \frac{\mathcal{H}(J(\mathbf{x}))I\{J(\mathbf{x}) \geq \bar{\chi}_1\}}{E_{\theta_1}[\hat{\mathcal{H}}_1(J(\mathbf{X}))I\{J(\mathbf{X}) \geq \bar{\chi}_1\}]},$$

where the last equality follows from the fact that the sequence $\{\bar{\chi}_k\}$ is non-decreasing. Proceeding iteratively, it is easy to see that

$$g_{k+1}(\mathbf{x}) = \frac{[\mathcal{H}(J(\mathbf{x}))]^k I\{J(\mathbf{x}) \geq \bar{\chi}_k\}}{E_{\theta_k}[\hat{\mathcal{H}}_k(J(\mathbf{X}))I\{J(\mathbf{X}) \geq \bar{\chi}_k\}]}, \quad \forall k = 0, 1, \ldots. \tag{4.5}$$

Thus, the KL-divergence between g_{k+1} and $f(\cdot, \theta)$ can be written as

$$\begin{aligned}\mathcal{D}\big(g_{k+1}, f(\cdot, \theta)\big) &= E_{g_{k+1}}\big[\ln g_{k+1}(\mathbf{X})\big] - E_{g_{k+1}}\big[\ln f(\mathbf{X}, \theta)\big] \\ &= E_{g_{k+1}}\big[\ln g_{k+1}(\mathbf{X})\big] - \frac{E_{\theta_k}[\hat{\mathcal{H}}_k(J(\mathbf{X}))I\{J(\mathbf{X}) \geq \bar{\chi}_k\} \ln f(\mathbf{X}, \theta)]}{E_{\theta_k}[\hat{\mathcal{H}}_k(J(\mathbf{X}))I\{J(\mathbf{X}) \geq \bar{\chi}_k\}]},\end{aligned}$$

$k = 0, 1, \ldots$. The result follows by observing that minimizing $\mathcal{D}(g_{k+1}, f(\cdot, \theta))$ is equivalent to maximizing $E_{\theta_k}[\hat{\mathcal{H}}_k(J(\mathbf{X}))I\{J(\mathbf{X}) \geq \bar{\chi}_k\} \ln f(\mathbf{X}, \theta)]$. □

4.1.1.1 Natural Exponential Family

Convergence of the MRAS$_0$ algorithm clearly depends on the choice of the family of parameterized distributions. For example, if the parameterized family is a singleton set, i.e., contains only one distribution, then there is in general no way to ensure the convergence of the algorithm. In addition, a practical consideration is selecting a family for which the parameter update given by (4.4) is relatively easy. The natural exponential family (NEF) results in a globally convergent algorithm for which the parameter update given by (4.4) can be obtained analytically.

Definition 4.2 A parameterized family of distributions $\{f(\cdot, \theta), \theta \in \Theta \subseteq \Re^m\}$ on $\mathcal{X}$ is said to belong to the natural exponential family (NEF) if there exist mappings $h : \Re^n \to \Re$, $\Upsilon : \Re^n \to \Re^m$, and $K : \Re^m \to \Re$ such that

$$f(\mathbf{x}, \theta) = \exp\big\{\theta^T \Upsilon(\mathbf{x}) - K(\theta)\big\}h(\mathbf{x}), \quad \forall \theta \in \Theta, \tag{4.6}$$

where $K(\theta) = \ln \int_{\mathbf{x} \in \mathcal{X}} \exp\{\theta^T \Upsilon(\mathbf{x})\}h(\mathbf{x})\nu(d\mathbf{x})$, Θ is the natural parameter space $\Theta = \{\theta \in \Re^m : |K(\theta)| < \infty\}$, and the superscript "$T$" denotes the vector transposition. For the case where $f(\cdot, \theta)$ is a p.d.f., we assume that Υ is a continuous mapping.

The function $K(\theta)$, called the log partition function, plays an important role in the theory of natural exponential family. Let $\text{int}(\Theta)$ be the interior of Θ. It is well-known that for any $\theta \in \text{int}(\Theta)$, $K(\theta)$ is strictly convex with $\nabla_\theta K(\theta) = E_\theta[\Upsilon(\mathbf{X})]$ and Hessian matrix $\text{Cov}_\theta[\Upsilon(\mathbf{X})]$. Therefore, the Jacobian of the mean vector function $m(\cdot) : \Re^m \to \Re^m$ defined by

$$m(\theta) := E_\theta\big[\Upsilon(\mathbf{X})\big]$$

is strictly positive definite and invertible. From the inverse function theorem, it follows that $m(\theta)$ is also invertible. Intuitively, $m(\theta)$ can be viewed as a transformed version of the sufficient statistic $\Upsilon(\mathbf{x})$, whose value contains all information necessary in estimating the parameter θ. Many common distributions belong to the NEF, e.g., Gaussian, Poisson, binomial, geometric, and certain multivariate forms of them.

For continuous optimization problems in $\Re^n$, if multivariate normal p.d.f.s are used in MRAS_0, i.e.,

$$f(\mathbf{x}, \theta_k) = \frac{1}{\sqrt{(2\pi)^n |\Sigma_k|}} \exp\left(-\frac{1}{2}(\mathbf{x} - \mu_k)^T \Sigma_k^{-1}(\mathbf{x} - \mu_k)\right), \tag{4.7}$$

where $\theta_k := (\mu_k;\ \Sigma_k)$, in which the parameters are updated in (4.4) as

$$\mu_{k+1} = \frac{E_{\theta_k}[\{[\mathcal{H}(J(\mathbf{X}))]^k / f(\mathbf{X}, \theta_k)\} I\{J(\mathbf{X}) \geq \bar{\gamma}_k\} \mathbf{X}]}{E_{\theta_k}[\{[\mathcal{H}(J(\mathbf{X}))]^k / f(\mathbf{X}, \theta_k)\} I\{J(\mathbf{X}) \geq \bar{\gamma}_k\}]}, \tag{4.8}$$

$$\Sigma_{k+1} = \frac{E_{\theta_k}[\{[\mathcal{H}(J(\mathbf{X}))]^k / f(\mathbf{X}, \theta_k)\} I\{J(\mathbf{X}) \geq \bar{\gamma}_k\} (\mathbf{X} - \mu_{k+1})(\mathbf{X} - \mu_{k+1})^T]}{E_{\theta_k}[\{[\mathcal{H}(J(\mathbf{X}))]^k / f(\mathbf{X}, \theta_k)\} I\{J(\mathbf{X}) \geq \bar{\gamma}_k\}]}, \tag{4.9}$$

the sequence of parameterized p.d.f.s will converge to a degenerate p.d.f. with all probability mass at the global optimum.

If the components of the random vector $\mathbf{X} = (\mathsf{X}_1, \ldots, \mathsf{X}_n)$ are independent, i.e., each has a univariate density/mass of the form

$$f(\mathsf{x}_i, \vartheta_i) = \exp\big(\mathsf{x}_i \vartheta_i - K(\vartheta_i)\big) h(\mathsf{x}_i), \quad \mathsf{x}_i \in \Re,\ \vartheta_i \in \Re,\ \forall i = 1, \ldots, n,$$

then the algorithm will lead to the following convergence:

$$\lim_{k \to \infty} m(\theta_k) := \lim_{k \to \infty} E_{\theta_k}[\mathbf{X}] = \mathbf{x}^*, \quad \text{where } \theta_k := \big(\vartheta_1^k, \ldots, \vartheta_n^k\big).$$

4.1.2 *The* MRAS_1 *Algorithm (Adaptive Monte Carlo Version)*

MRAS_0 is an idealized algorithm that assumes that quantile values and expectations with respect to $f(\cdot, \theta)$ can be evaluated exactly. In practice, i.i.d. samples are drawn from $f(\cdot, \theta)$ in order to estimate expected values and quantiles with their corresponding sample mean and sample quantiles. Figure 4.4 presents the MRAS_1 algorithm, which is an adaptive Monte Carlo version of MRAS_0 that uses samples from $f(\cdot, \theta)$ and adaptive updating of the quantile parameter and sample size. For example, the parameter update given by (4.4) of MRAS_0 is replaced with its stochastic counterpart in (4.10).

However, the theoretical convergence can no longer be guaranteed for a simple stochastic counterpart of MRAS_0. In particular, the set $\{\mathbf{x} : J(\mathbf{x}) \geq \bar{\gamma}_k, \mathbf{x} \in \{\mathbf{X}_k^1, \ldots, \mathbf{X}_k^{N_k}\}\}$ involved in (4.10) may be empty, since all the random samples

Algorithm MRAS_1**—Adaptive Monte Carlo Version**

Input: $\rho_0 \in (0,1]$, $N_0 > 1$, $\varepsilon \geq 0$, $\alpha > 1$, $\lambda \in (0,1]$, strictly increasing function $\mathcal{H}$: $\Re \to \Re^+$, family of distributions $\{f(\cdot,\theta)\}$, with θ_0 s.t. $f(\mathbf{x},\theta_0) > 0\ \forall\, \mathbf{x} \in \mathcal{X}$.

Initialization: Set iteration count $k = 0$; $\tilde{\theta}_0 = \theta_0$.

Loop until Stopping Rule is satisfied:

1. Generate N_k i.i.d. samples $\Lambda_k = \{\mathbf{X}_k^1, \ldots, \mathbf{X}_k^{N_k}\}$ according to $\tilde{f}(\cdot,\tilde{\theta}_k) = (1-\lambda) f(\cdot,\tilde{\theta}_k) + \lambda f(\cdot,\theta_0)$.
2. Compute the sample $(1-\rho_k)$-quantile:

$$\tilde{\chi}_k(\rho_k, N_k) = J_{(\lceil (1-\rho_k) N_k \rceil)},$$

   where $J_{(i)}$ is the ith order statistic of $\{J(\mathbf{X}_k^i),\ i = 1, \ldots, N_k\}$.
3. Update elite threshold:
   if $k = 0$ **or** $\tilde{\chi}_k(\rho_k, N_k) \geq \bar{\chi}_{k-1} + \frac{\varepsilon}{2}$, **then**
   3a. Set $\bar{\chi}_k = \tilde{\chi}_k(\rho_k, N_k),\ \rho_{k+1} = \rho_k,\ N_{k+1} = N_k$;
   else, find the largest $\bar{\rho} \in (0, \rho_k)$ such that $\tilde{\chi}_k(\bar{\rho}, N_k) \geq \bar{\chi}_{k-1} + \frac{\varepsilon}{2}$;
   3b. **if** such a $\bar{\rho}$ exists, **then** set $\bar{\chi}_k = \tilde{\chi}_k(\bar{\rho}, N_k),\ \rho_{k+1} = \bar{\rho},\ N_{k+1} = N_k$;
   3c. **else** (no such $\bar{\rho}$ exists), set $\bar{\chi}_k = \bar{\chi}_{k-1},\ \rho_{k+1} = \rho_k,\ N_{k+1} = \lceil \alpha N_k \rceil$.
   endif
4. Update parameter vector:

$$\tilde{\theta}_{k+1} \in \arg\max_{\theta \in \Theta} \frac{1}{N_k} \sum_{\mathbf{x} \in \Lambda_k} \frac{[\mathcal{H}(J(\mathbf{x}))]^k}{\tilde{f}(\mathbf{x},\tilde{\theta}_k)} I\{J(\mathbf{x}) \geq \bar{\chi}_k\} \ln f(\mathbf{x},\theta). \tag{4.10}$$

5. $k \leftarrow k+1$.

Output: $\tilde{\theta}_k$.

Fig. 4.4 Description of MRAS_1 algorithm

generated at the current iteration may be much worse than those generated at the previous iteration. Thus, we can only expect the algorithm to converge if the expected values in the MRAS_0 algorithm are closely approximated. The quality of the approximation will depend on the number of elite samples used at each iteration in the parameter update, and this quantity depends on the quantile parameter—ρ in MRAS_0—and the number of samples generated in each iteration. In MRAS_1, the sample size is adaptively increasing, and the quantile parameter is adaptively decreasing. The rate of increase in the sample size is controlled by an extra parameter $\alpha > 1$, specified during the initialization step. For example, if the initial sample size is N_0, then after k increments, the sample size will be approximately $\lceil \alpha^k N_0 \rceil$.

At each iteration k, N_k random samples $\Lambda_k = \{\mathbf{X}_k^1, \ldots, \mathbf{X}_k^{N_k}\}$ are drawn from the distribution $\tilde{f}(\cdot,\tilde{\theta}_k)$, which is a mixture of the initial distribution $f(\cdot,\theta_0)$ and the distribution calculated from the previous iteration $f(\cdot,\tilde{\theta}_k)$. In practice, the ini-

tial distribution $f(\cdot, \theta_0)$ can be chosen according to some prior knowledge of the problem structure; however, if nothing is known about where the good solutions are, it could be chosen such that each region in the solution space will have an (approximately) equal probability of being sampled. Intuitively, mixing in the initial distribution forces the algorithm to explore the entire solution space and to maintain a global perspective during the search process. Also note that if $\lambda = 1$, then random samples will always be drawn from the initial distribution, in which case MRAS_1 becomes a pure random sampling approach.

At Step 2, the sample $(1 - \rho_k)$-quantile $\tilde{\chi}_k$ is calculated by first ordering the sample performances $J(\mathbf{X}_k^i)$, $i = 1, \ldots, N_k$ from smallest to largest, $J_{(1)} \leq J_{(2)} \leq \cdots \leq J_{(N_k)}$, and then taking the $\lceil (1 - \rho_k) N_k \rceil$th order statistic, where $\lceil a \rceil$ is the smallest integer greater than or equal to a. Step 3 of MRAS_1 extracts a sequence of non-decreasing thresholds $\{\bar{\chi}_k\}$ from the sequence of sample quantiles $\{\tilde{\chi}_k\}$, and determines the appropriate values of ρ_{k+1} and N_{k+1} to be used in the next iteration. At each iteration k, Step 3 first checks whether the inequality $\tilde{\chi}_k(\rho_k, N_k) \geq \bar{\chi}_{k-1} + \varepsilon/2$ is satisfied, where $\bar{\chi}_{k-1}$ is the threshold value used in the previous iteration. If the inequality holds, this means that both the current ρ_k value and the current sample size N_k are satisfactory, and the parameter update (in Step 4) is carried out using the newly obtained sample quantile. Otherwise, either ρ_k is too large or the sample size N_k is too small. To determine which, it is checked to see if there exists a smaller $\bar{\rho} < \rho_k$ such that the above inequality can be satisfied with the new sample $(1 - \bar{\rho})$-quantile. If such a $\bar{\rho}$ does exist, then the current sample size N_k is still deemed acceptable, and only ρ_k is decreased. Accordingly, the parameter vector is updated (in Step 4) using the sample $(1 - \bar{\rho})$-quantile. On the other hand, if no such $\bar{\rho}$ exists, then the parameter vector is updated (in Step 4) by using the previous elite threshold, and the sample size N_k is increased by a factor α.

It is important to note that the set $\{\mathbf{x} \in \Lambda_k : J(\mathbf{x}) \geq \bar{\chi}_k\}$ could be empty if Step 3c is visited. If this happens, the right-hand side of (4.10) will be equal to zero, so any $\theta \in \Theta$ is a maximizer, in which case we take $\tilde{\theta}_{k+1} := \tilde{\theta}_k$.

4.1.3 *The* MRAS_2 *Algorithm (Stochastic Optimization)*

Now we extend the MRAS method to the stochastic optimization setting, in which only noisy estimates of the performance function are available. Specifically, we consider optimization problems of the following form:

$$\mathbf{x}^* \in \arg\max_{\mathbf{x} \in \mathcal{X}} E\big[\mathcal{J}(\mathbf{x}, \psi)\big], \tag{4.11}$$

where the solution space $\mathcal{X}$ is a non-empty set in $\Re^n$, $\mathcal{J} : \mathcal{X} \times \Psi \to \Re$ is a deterministic function, and ψ is a random variable (possibly depending on $\mathbf{x}$) taking values in Ψ, which represents the stochastic effects of the system and with respect to which the expectation is taken. We assume that $\mathcal{J}(\mathbf{x}, \psi)$ is measurable and integrable with respect to the distribution of ψ for all $\mathbf{x} \in \mathcal{X}$. Define $J(\mathbf{x}) = E[\mathcal{J}(\mathbf{x}, \psi)]$, and assume that $J(\mathbf{x})$ cannot be obtained easily, but that i.i.d. samples of $\mathcal{J}(\mathbf{x}, \psi)$ are

Algorithm MRAS$_2$**—Stochastic Optimization**

Input: $\rho_0 \in (0,1]$, $N_0 > 1$, $\varepsilon > 0$, $\alpha > 1$, $\lambda \in (0,1)$, strictly increasing function $\mathcal{H} : \Re \to \Re^+$, family of distributions $\{f(\cdot,\theta)\}$, with θ_0 s.t. $f(\mathbf{x},\theta_0) > 0\ \forall \mathbf{x} \in \mathcal{X}$, simulation allocation rule $\{M_k\}$.

Initialization: Set iteration count $k = 0$; $\tilde{\theta}_0 = \theta_0$.

Loop until Stopping Rule is satisfied:

1. Generate N_k i.i.d. samples $\Lambda_k = \{\mathbf{X}_k^1, \ldots, \mathbf{X}_k^{N_k}\}$ according to $\tilde{f}(\cdot,\tilde{\theta}_k) = (1-\lambda) f(\cdot,\tilde{\theta}_k) + \lambda f(\cdot,\theta_0)$.
   Take M_k observations for every $\mathbf{x} \in \Lambda_k$,
   and calculate sample performances $\bar{\mathcal{J}}_k(\mathbf{x}) = \frac{1}{M_k}\sum_{i=1}^{M_k} \mathcal{J}_i(\mathbf{x})\ \forall \mathbf{x} \in \Lambda_k$.
2. Compute the sample $(1-\rho_k)$-quantile:
$$\tilde{\chi}_k(\rho_k, N_k) = \bar{\mathcal{J}}_{(\lceil (1-\rho_k)N_k \rceil)},$$
   where $\bar{\mathcal{J}}_{(i)}$ is the ith order statistic of $\{\bar{\mathcal{J}}(\mathbf{X}_k^i),\ i = 1, \ldots, N_k\}$.
3. **if** $k = 0$ **or** $\tilde{\chi}_k(\rho_k, N_k) \geq \bar{\chi}_{k-1} + \varepsilon$, **then**
   3a. Set $\bar{\chi}_k = \tilde{\chi}_k(\rho_k, N_k)$, $\rho_{k+1} = \rho_k$, $N_{k+1} = N_k$,
   $\mathbf{X}_k^* = \mathbf{X}_{1-\rho_k}$, where $\mathbf{X}_{1-\rho_k} \in \{\mathbf{x} \in \Lambda_k : \bar{\mathcal{J}}_k(\mathbf{x}) = \tilde{\chi}_k(\rho_k, N_k)\}$;
   else, find the largest $\bar{\rho} \in (0, \rho_k)$ such that $\tilde{\chi}_k(\bar{\rho}, N_k) \geq \bar{\chi}_{k-1} + \varepsilon$;
   3b. **if** $\bar{\rho}$ exists, **then** set $\bar{\chi}_k = \tilde{\chi}_k(\bar{\rho}, N_k)$, $\rho_{k+1} = \bar{\rho}$, $N_{k+1} = N_k$,
   $\mathbf{X}_k^* = \mathbf{X}_{1-\bar{\rho}} \in \{\mathbf{x} \in \Lambda_k :\ \bar{\mathcal{J}}_k(\mathbf{x}) = \tilde{\chi}_k(\bar{\rho}, N_k)\}$;
   3c. **else**, set $\bar{\chi}_k = \bar{\mathcal{J}}_k(\mathbf{X}_{k-1}^*)$, $\rho_{k+1} = \rho_k$, $N_{k+1} = \lceil \alpha N_k \rceil$,
   $\mathbf{X}_k^* = \mathbf{X}_{k-1}^*$.
   endif
4. Update parameter vector:
$$\tilde{\theta}_{k+1} \in \arg\max_{\theta \in \Theta} \frac{1}{N_k} \sum_{\mathbf{x} \in \Lambda_k} \frac{[\mathcal{H}(\bar{\mathcal{J}}_k(\mathbf{x}))]^k}{\tilde{f}(\mathbf{x},\tilde{\theta}_k)} \tilde{I}\big(\bar{\mathcal{J}}_k(\mathbf{x}), \bar{\chi}_k\big) \ln f(\mathbf{x},\theta), \tag{4.12}$$
$$\text{where } \tilde{I}(z,\chi) := \begin{cases} 0 & \text{if } z \leq \chi - \varepsilon, \\ (z - \chi + \varepsilon)/\varepsilon & \text{if } \chi - \varepsilon < z < \chi, \\ 1 & \text{if } z \geq \chi. \end{cases}$$
5. $k \leftarrow k + 1$.

Output: $\tilde{\theta}_k$.

Fig. 4.5 Description of MRAS$_2$ algorithm

available, e.g., via simulation or real-time observation, and denote the ith sample by $\mathcal{J}_i(\mathbf{x})$. We assume that (4.11) has a unique global optimal solution, i.e., $\exists\, \mathbf{x}^* \in \mathcal{X}$ such that $J(\mathbf{x}) < J(\mathbf{x}^*)\ \forall \mathbf{x} \neq \mathbf{x}^*,\ \mathbf{x} \in \mathcal{X}$.

Figure 4.5 presents the MRAS$_2$ algorithm, which is a generalization of the MRAS$_1$ algorithm appropriately modified and extended for stochastic settings, the

main addition being the requirement of an additional sample mean to estimate the performance function. Thus, in addition to the sampling allocation rule $\{N_k\}$ used in MRAS$_1$, there is an observation allocation rule $\{M_k, k = 0, 1, \ldots\}$, specified during the initialization step of MRAS$_2$, where M_k indicates the number of simulation observations to be allocated to each of the candidate solutions sampled at the kth iteration. Assumption L3 in Sect. 4.2 provides conditions on $\{M_k\}$ that ensure convergence of the algorithm. At iteration k, the sample mean based on the M_k observations $\bar{\mathcal{J}}_k(\mathbf{x}) = \frac{1}{M_k}\sum_{i=1}^{M_k} \mathcal{J}_i(\mathbf{x})$ is used to estimate the true performance $J(\mathbf{x})$.

At Step 2 of MRAS$_2$, the sample $(1-\rho_k)$-quantile $\tilde{\chi}_k$ with respect to $\tilde{f}(\cdot, \tilde{\theta}_k)$ is calculated by first ordering the sample performances $\bar{\mathcal{J}}_k(\mathbf{X}_k^i)$, $i = 1, \ldots, N_k$ from smallest to largest, $\bar{\mathcal{J}}_{k,(1)} \leq \bar{\mathcal{J}}_{k,(2)} \leq \cdots \leq \bar{\mathcal{J}}_{k,(N_k)}$, and then taking the $\lceil (1-\rho_k)N_k \rceil$th order statistic.

MRAS$_2$ also extends the MRAS$_1$ construction of the sequence of thresholds $\{\bar{\chi}_k, k = 0, 1, \ldots\}$. In particular, at each iteration k, the algorithm uses an additional variable $\mathbf{X}_k^*$ to record the sample that achieves the current threshold value $\bar{\chi}_k$. Whenever Step 3c is visited, M_k i.i.d. observations are allocated to $\mathbf{X}_{k-1}^*$ (i.e., the sample that achieves the threshold value at iteration $k-1$), setting the current threshold as $\bar{\chi}_k = \bar{\mathcal{J}}_k(\mathbf{X}_{k-1}^*) = \frac{1}{M_k}\sum_{i=1}^{M_k} \mathcal{J}_i(\mathbf{X}_{k-1}^*)$. If more than one sample achieves a threshold value, ties are broken arbitrarily. It is easy to observe that in a deterministic setting, i.e., $J(\mathbf{x})$ can be evaluated exactly, Steps 1–3 of MRAS$_2$ coincide with those of MRAS$_1$.

Another modification from MRAS$_1$ occurs at Step 4, where a continuous filter function $\tilde{I}(\cdot, \cdot)$, as opposed to the original indicator function, is used in parameter updating (cf. (4.12)). The function eliminates from consideration those obviously inferior solutions among Λ_k that have performance worse than $\bar{\gamma}_k - \varepsilon$. However, since all performance evaluations will contain some noise, $\tilde{I}(\cdot, \bar{\gamma}_k)$ is chosen to be continuous to provide some robustness, in the sense that those solutions with true performance better than $\bar{\gamma}_k$ but whose current estimates are slightly worse than $\bar{\gamma}_k$ (between $\bar{\gamma}_k - \varepsilon$ and $\bar{\gamma}_k$) will still be included in parameter updating. Thus, in the long run, as more precise performance estimates are obtained, $\tilde{I}(\cdot, \bar{\gamma}_k)$ ensures (w.p.1) that all solutions with true performance better than $\bar{\gamma}_k$ will be used to calculate the new parameter $\tilde{\theta}_{k+1}$.

The following lemma shows that there is a sequence of reference models $\{\tilde{g}_k\}$ implicit in MRAS$_2$, and that the corresponding parameter update in MRAS$_2$ minimizes the KL-divergence $\mathcal{D}(\tilde{g}_{k+1}, f(\cdot, \theta))$. The proof is simple and is thus omitted.

Lemma 4.3 *The parameter $\tilde{\theta}_{k+1}$ computed at the kth iteration of* MRAS$_2$ *via* (4.12) *minimizes the KL-divergence $\mathcal{D}(\tilde{g}_{k+1}, f(\cdot, \theta))$, where*

$$\tilde{g}_{k+1}(\mathbf{x}) := \begin{cases} \frac{[[\mathcal{H}(\bar{\mathcal{J}}_k(\mathbf{x}))]^k / \tilde{f}(\mathbf{x}, \tilde{\theta}_k)]\tilde{I}(\bar{\mathcal{J}}_k(\mathbf{x}), \bar{\chi}_k)}{\sum_{\mathbf{x}\in\Lambda_k}[[\mathcal{H}(\bar{\mathcal{J}}_k(\mathbf{x}))]^k / \tilde{f}(\mathbf{x}, \tilde{\theta}_k)]\tilde{I}(\bar{\mathcal{J}}_k(\mathbf{x}), \bar{\chi}_k)} \\ \quad \textit{if } \{\mathbf{x} \in \Lambda_k : \bar{\mathcal{J}}_k(\mathbf{x}) > \bar{\chi}_k - \varepsilon\} \neq \emptyset, \\ \tilde{g}_k(\mathbf{x}) \quad \textit{otherwise}, \end{cases} \tag{4.13}$$

$k = 0, 1, \ldots,$ *where*

$$\bar{\chi}_k := \begin{cases} \tilde{\chi}_k(\rho_k, N_k) & \text{if Step 3a is visited,} \\ \tilde{\chi}_k(\bar{\rho}, N_k) & \text{if Step 3b is visited,} \\ \bar{\mathcal{J}}_k(\mathbf{X}^*_{k-1}) & \text{if Step 3c is visited.} \end{cases}$$

4.2 Convergence Analysis of MRAS

In this section, we provide theoretical global convergence results for each of the algorithms presented in the previous section when the parameterized distribution is NEF.

4.2.1 MRAS$_0$ *Convergence*

Proving global convergence of MRAS$_0$ requires some additional regularity conditions.

Assumption A1 For any given constant $\xi < J(\mathbf{x}^*)$, $\{\mathbf{x} : J(\mathbf{x}) \geq \xi\} \cap \mathcal{X}$ has a strictly positive Lebesgue or discrete measure.

Assumption A2 For any given constant $\delta > 0$, $\sup_{\mathbf{x} \in A_\delta} J(\mathbf{x}) < J(\mathbf{x}^*)$, where $A_\delta := \{\mathbf{x} : \|\mathbf{x} - \mathbf{x}^*\| \geq \delta\} \cap \mathcal{X}$, and we define the supremum over the empty set to be $-\infty$.

Assumption A3 $\exists$ compact set Ξ such that $\{\mathbf{x} : J(\mathbf{x}) \geq \bar{\chi}_0\} \cap \mathcal{X} \subseteq \Xi$, where $\bar{\chi}_0 = \sup_l \{l : P_{\theta_0}(J(\mathbf{X}) \geq l) \geq \rho\}$ is defined as in the MRAS$_0$ algorithm.

Assumption A4 The maximizer of (4.4) is an interior point of Θ $\forall k$.

Intuitively, Assumption A1 ensures that any neighborhood of the optimal solution $\mathbf{x}^*$ will have a positive probability of being sampled. For ease of exposition, Assumption A1 restricts the class of problems under consideration to either continuous or discrete problems, but the convergence results can be easily extended to problems with a mixture of both continuous and discrete variables. Since J has a unique global optimizer, Assumption A2 is satisfied by many functions encountered in practice. Note that both Assumptions A1 and A2 hold trivially when $\mathcal{X}$ is (discrete) finite and the counting measure is used. Assumption A3 restricts the search of the MRAS$_0$ algorithm to some compact set; it is satisfied if the function J has compact level sets or the solution space $\mathcal{X}$ is compact. In actual implementation of the algorithm, the parameter updating step of MRAS$_0$ given by (4.4) is often posed as an unconstrained optimization problem, i.e., $\Theta = \Re^m$, in which case Assumption A4 is automatically satisfied.

The convergence of MRAS$_0$ requires the following key observation.

Lemma 4.4 *If Assumptions A3–A4 hold, then we have*

$$m(\theta_{k+1}) := E_{\theta_{k+1}}\big[\Upsilon(\mathbf{X})\big] = E_{g_{k+1}}\big[\Upsilon(\mathbf{X})\big], \quad \forall k = 0, 1, \ldots,$$

where $E_{\theta_{k+1}}$ and $E_{g_{k+1}}$ denote the expectations taken with respect to $f(\cdot, \theta_{k+1})$ and g_{k+1}, respectively.

Proof Define $J_k(\theta, \bar{\gamma}_k) := \int_{\mathcal{X}} [\mathcal{H}(J(\mathbf{x}))]^k I\{J(\mathbf{x}) \geq \bar{\gamma}_k\} \ln f(\mathbf{x}, \theta) \nu(d\mathbf{x})$. Since $f(\cdot, \theta)$ belongs to the NEF, we can write

$$\begin{aligned} J_k(\theta, \bar{\gamma}_k) = & \int_{\mathcal{X}} \big[\mathcal{H}\big(J(\mathbf{x})\big)\big]^k I\big\{J(\mathbf{x}) \geq \bar{\gamma}_k\big\} \ln h(\mathbf{x}) \nu(d\mathbf{x}) \\ & + \int_{\mathcal{X}} \big[\mathcal{H}\big(J(\mathbf{x})\big)\big]^k I\big\{J(\mathbf{x}) \geq \bar{\gamma}_k\big\} \theta^T \Upsilon(\mathbf{x}) \nu(d\mathbf{x}) \\ & - \int_{\mathcal{X}} \big[\mathcal{H}\big(J(\mathbf{x})\big)\big]^k I\big\{J(\mathbf{x}) \geq \bar{\gamma}_k\big\} \ln \bigg[\int_{\mathcal{X}} \exp\big(\theta^T \Upsilon(\mathbf{y})\big) h(\mathbf{y}) \nu(d\mathbf{y}) \bigg] \nu(d\mathbf{x}). \end{aligned}$$

Thus the gradient of $J_k(\theta, \bar{\gamma}_k)$ with respect to θ can be expressed as

$$\begin{aligned} \nabla_\theta J_k(\theta, \bar{\gamma}_k) = & \int_{\mathcal{X}} \big[\mathcal{H}\big(J(\mathbf{x})\big)\big]^k I\big\{J(\mathbf{x}) \geq \bar{\gamma}_k\big\} \Upsilon(\mathbf{x}) \nu(d\mathbf{x}) \\ & - \frac{\int_{\mathcal{X}} e^{\theta^T \Upsilon(\mathbf{y})} \Upsilon(\mathbf{y}) h(\mathbf{y}) \nu(d\mathbf{y})}{\int_{\mathcal{X}} e^{\theta^T \Upsilon(\mathbf{y})} h(\mathbf{y}) \nu(d\mathbf{y})} \int_{\mathcal{X}} \big[\mathcal{H}\big(J(\mathbf{x})\big)\big]^k I\big\{J(\mathbf{x}) \geq \bar{\gamma}_k\big\} \nu(d\mathbf{x}), \end{aligned}$$

where the validity of the interchange of derivative and integral above is guaranteed by the dominated convergence theorem.

By Assumption A3 and the non-decreasing property of the sequence $\{\bar{\gamma}_k\}$, it turns out that the above gradient $\nabla_\theta J_k(\theta, \bar{\gamma}_k)$ is finite and thus well-defined. Moreover, since $\rho > 0$, it can be seen from the MRAS$_0$ algorithm that the set $\{\mathbf{x} : J(\mathbf{x}) \geq \bar{\gamma}_k\} \cap \mathcal{X}$ has a strictly positive Lebesgue/counting measure. It follows that we must have $\int_{\mathcal{X}} [\mathcal{H}(J(\mathbf{x}))]^k I\{J(\mathbf{x}) \geq \bar{\gamma}_k\} \nu(d\mathbf{x}) > 0$.

By setting $\nabla_\theta J_k(\theta, \bar{\gamma}_k) = 0$, it immediately follows that

$$\int_{\mathcal{X}} \frac{[\mathcal{H}(J(\mathbf{x}))]^k I\{J(\mathbf{x}) \geq \bar{\gamma}_k\} \Upsilon(\mathbf{x})}{\int_{\mathcal{X}} [\mathcal{H}(J(\mathbf{y}))]^k I\{J(\mathbf{y}) \geq \bar{\gamma}_k\} \nu(d\mathbf{y})} \nu(d\mathbf{x}) = \int_{\mathcal{X}} \frac{e^{\theta^T \Upsilon(\mathbf{x})} h(\mathbf{x}) \Upsilon(\mathbf{x})}{\int_{\mathcal{X}} e^{\theta^T \Upsilon(\mathbf{y})} h(\mathbf{y}) \nu(d\mathbf{y})} \nu(d\mathbf{x}),$$

and by the definitions of g_{k+1} (see the proof of Lemma 4.1) and $f(\cdot, \theta)$, we have

$$E_{g_{k+1}}\big[\Upsilon(\mathbf{X})\big] = E_\theta\big[\Upsilon(\mathbf{X})\big]. \tag{4.14}$$

By Assumption A4, since θ_{k+1} is an optimal solution of the problem

$$\arg\max_\theta J_k(\theta, \bar{\gamma}_k),$$

it must satisfy Eq. (4.14). Therefore we conclude that

$$E_{g_{k+1}}\big[\Upsilon(\mathbf{X})\big] = E_{\theta_{k+1}}\big[\Upsilon(\mathbf{X})\big], \quad \forall k = 0, 1, \ldots. \qquad \square$$

We have the following convergence result for the MRAS$_0$ algorithm.

Theorem 4.5 *Let* $\{\theta_k,\ k = 1, 2, \ldots\}$ *be the sequence of parameters generated by* MRAS$_0$. *If* $\varepsilon > 0$ *and Assumptions* A1–A4 *are satisfied, then*

$$\lim_{k\to\infty} m(\theta_k) := \lim_{k\to\infty} E_{\theta_k}\big[\Upsilon(\mathbf{X})\big] = \Upsilon\big(\mathbf{x}^*\big), \tag{4.15}$$

where the limit is component-wise.

Proof In Lemma 4.4, we have already established a relationship between the sequence of reference models $\{g_k\}$ and the sequence of sampling distributions $\{f(\cdot, \theta_k)\}$. Therefore, proving Theorem 4.5 amounts to showing that $\lim_{k\to\infty} E_{g_k}[\Upsilon(\mathbf{X})] = \Upsilon(\mathbf{x}^*)$.

Recall from Lemma 4.1 that g_{k+1} can be expressed recursively as

$$g_{k+1}(\mathbf{x}) := \frac{\mathcal{H}(J(\mathbf{x}))I\{J(\mathbf{x}) \geq \bar{\chi}_k\}g_k(\mathbf{x})}{E_{g_k}[\mathcal{H}(J(\mathbf{X}))I\{J(\mathbf{X}) \geq \bar{\chi}_k\}]} \quad \forall \mathbf{x} \in \mathcal{X},\ k = 1, 2, \ldots.$$

Thus

$$\begin{aligned} E_{g_{k+1}}\big[\mathcal{H}\big(J(\mathbf{X})\big)I\big\{J(\mathbf{X}) \geq \bar{\chi}_k\big\}\big] &= \frac{E_{g_k}[[\mathcal{H}(J(\mathbf{X}))]^2 I\{J(\mathbf{X}) \geq \bar{\chi}_k\}]}{E_{g_k}[\mathcal{H}(J(\mathbf{X}))I\{J(\mathbf{X}) \geq \bar{\chi}_k\}]} \\ &\geq E_{g_k}\big[\mathcal{H}\big(J(\mathbf{X})\big)I\big\{J(\mathbf{X}) \geq \bar{\chi}_k\big\}\big]. \end{aligned} \tag{4.16}$$

Since $\bar{\chi}_k \leq J(\mathbf{x}^*)\ \forall k$, and each strict increment in the sequence $\{\bar{\chi}_k\}$ is lower bounded by the quantity $\varepsilon > 0$, there exists a finite $\mathcal{N}$ such that $\bar{\chi}_{k+1} = \bar{\chi}_k,\ \forall k \geq \mathcal{N}$. Before we proceed any further, we need to distinguish between two cases, $\bar{\chi}_{\mathcal{N}} = J(\mathbf{x}^*)$ and $\bar{\chi}_{\mathcal{N}} < J(\mathbf{x}^*)$.

Case 1. If $\bar{\chi}_{\mathcal{N}} = J(\mathbf{x}^*)$ (note that since $\rho > 0$, this could only happen when the solution space is discrete), then from the definition of g_{k+1} (see Lemma 4.1), we obviously have

$$g_{k+1}(\mathbf{x}) = 0, \quad \forall \mathbf{x} \neq \mathbf{x}^*,$$

and

$$g_{k+1}\big(\mathbf{x}^*\big) = \frac{[\mathcal{H}(J(\mathbf{x}^*))]^k I\{J(\mathbf{x}^*) = J(\mathbf{x}^*)\}}{\int_{\mathcal{X}}[\mathcal{H}(J(\mathbf{x}))]^k I\{J(\mathbf{x}) = J(\mathbf{x}^*)\}\nu(d\mathbf{x})} = 1 \quad \forall k \geq \mathcal{N}.$$

Hence it follows immediately that

$$E_{g_{k+1}}\big[\Upsilon(\mathbf{X})\big] = \Upsilon\big(\mathbf{x}^*\big) \quad \forall k \geq \mathcal{N}.$$

Case 2. If $\bar{\chi}_{\mathcal{N}} < J(\mathbf{x}^*)$, then from Inequality (4.16), we have

$$\begin{aligned} &E_{g_{k+1}}\left[\mathcal{H}\big(J(\mathbf{X})\big)I\left\{J(\mathbf{X}) \geq \bar{\chi}_{k+1}\right\}\right] \\ &\quad \geq E_{g_k}\left[\mathcal{H}\big(J(\mathbf{X})\big)I\left\{J(\mathbf{X}) \geq \bar{\chi}_k\right\}\right], \quad \forall k \geq \mathcal{N}, \end{aligned} \tag{4.17}$$

i.e., the sequence $\{E_{g_k}[\mathcal{H}(J(\mathbf{X}))I\{J(\mathbf{X}) \geq \bar{\chi}_k\}], k = 1, 2, \ldots\}$ converges.

Now we show that the limit of the above sequence is $\mathcal{H}(J(\mathbf{x}^*))$. To do so, we proceed by contradiction and assume that

$$\mathcal{H}_* := \lim_{k\to\infty} E_{g_k}\left[\mathcal{H}\big(J(\mathbf{X})\big)I\left\{J(\mathbf{X}) \geq \bar{\chi}_k\right\}\right] < \mathcal{H}^* := \mathcal{H}\big(J\left(\mathbf{x}^*\right)\big). \tag{4.18}$$

Define the set $\mathcal{A}$ as

$$\mathcal{A} := \left\{\mathbf{x} : J(\mathbf{x}) \geq \bar{\chi}_{\mathcal{N}}\right\} \cap \left\{\mathbf{x} : \mathcal{H}\big(J(\mathbf{x})\big) \geq \frac{\mathcal{H}^* + \mathcal{H}_*}{2}\right\} \cap \mathcal{X}.$$

Since $\mathcal{H}$ is strictly increasing, its inverse $\mathcal{H}^{-1}$ exists. Thus $\mathcal{A}$ can be reformulated as

$$\mathcal{A} = \left\{\mathbf{x} : J(\mathbf{x}) \geq \max\left\{\bar{\chi}_{\mathcal{N}},\ \mathcal{H}^{-1}\left(\frac{\mathcal{H}^* + \mathcal{H}_*}{2}\right)\right\}\right\} \cap \mathcal{X}.$$

Since $\bar{\chi}_{\mathcal{N}} < J(\mathbf{x}^*)$, $\mathcal{A}$ has a strictly positive Lebesgue/discrete measure by Assumption A1.

Notice that g_k can be rewritten as

$$g_k(\mathbf{x}) = \prod_{i=1}^{k-1} \frac{\mathcal{H}(J(\mathbf{x}))I\{J(\mathbf{x}) \geq \bar{\chi}_i\}}{E_{g_i}[\mathcal{H}(J(\mathbf{X}))I\{J(\mathbf{X}) \geq \bar{\chi}_i\}]} \cdot g_1(\mathbf{x}).$$

Since $\lim_{k\to\infty} \frac{\mathcal{H}(J(\mathbf{x}))I\{J(\mathbf{x})\geq\bar{\chi}_k\}}{E_{g_k}[\mathcal{H}(J(\mathbf{X}))I\{J(\mathbf{X})\geq\bar{\chi}_k\}]} = \frac{\mathcal{H}(J(\mathbf{x}))I\{J(\mathbf{x})\geq\bar{\chi}_{\mathcal{N}}\}}{\mathcal{H}_*} > 1,\ \forall \mathbf{x} \in \mathcal{A}$, we conclude that

$$\lim_{k\to\infty} g_k(\mathbf{x}) = \infty, \quad \forall \mathbf{x} \in \mathcal{A}.$$

Thus, by Fatou's lemma, we have

$$\begin{aligned} 1 &= \liminf_{k\to\infty} \int_{\mathcal{X}} g_k(\mathbf{x})\nu(d\mathbf{x}) \geq \liminf_{k\to\infty} \int_{\mathcal{A}} g_k(\mathbf{x})\nu(d\mathbf{x}) \\ &\geq \int_{\mathcal{A}} \liminf_{k\to\infty} g_k(\mathbf{x})\nu(d\mathbf{x}) = \infty, \end{aligned}$$

which is a contradiction. Hence, it follows that

$$\lim_{k\to\infty} E_{g_k}\left[\mathcal{H}\big(J(\mathbf{X})\big)I\left\{J(\mathbf{X}) \geq \bar{\chi}_k\right\}\right] = \mathcal{H}^*. \tag{4.19}$$

To show $\lim_{k\to\infty} E_{g_k}[\Upsilon(\mathbf{X})] = \Upsilon(\mathbf{x}^*)$, we now bound the difference between $E_{g_k}[\Upsilon(\mathbf{X})]$ and $\Upsilon(\mathbf{x}^*)$. Note that $\forall k \geq \mathcal{N}$, we have

$$\begin{aligned}\left\| E_{g_k}\left[\Upsilon(\mathbf{X})\right] - \Upsilon\left(\mathbf{x}^*\right)\right\| &\leq \int_{\mathcal{X}} \left\|\Upsilon(\mathbf{x}) - \Upsilon\left(\mathbf{x}^*\right)\right\| g_k(\mathbf{x})\nu(d\mathbf{x}) \\ &= \int_{\mathcal{C}} \left\|\Upsilon(\mathbf{x}) - \Upsilon\left(\mathbf{x}^*\right)\right\| g_k(\mathbf{x})\nu(d\mathbf{x}), \end{aligned} \tag{4.20}$$

where $\mathcal{C} := \{\mathbf{x} : J(\mathbf{x}) \geq \bar{\chi}_{\mathcal{N}}\} \cap \mathcal{X}$ is the support of g_k, $\forall k > \mathcal{N}$.

By the assumption on Υ in Definition 4.2, for any given $\zeta > 0$, there exists a $\delta > 0$ such that $\|\mathbf{x} - \mathbf{x}^*\| < \delta$ implies $\|\Upsilon(\mathbf{x}) - \Upsilon(\mathbf{x}^*)\| < \zeta$. With A_δ defined from Assumption A2, we have from Inequality (4.20),

$$\begin{aligned}&\left\| E_{g_k}\left[\Upsilon(\mathbf{X})\right] - \Upsilon\left(\mathbf{x}^*\right)\right\| \\ &\leq \int_{A_\delta^c \cap \mathcal{C}} \left\|\Upsilon(\mathbf{x}) - \Upsilon\left(\mathbf{x}^*\right)\right\| g_k(\mathbf{x})\nu(d\mathbf{x}) \\ &\quad + \int_{A_\delta \cap \mathcal{C}} \left\|\Upsilon(\mathbf{x}) - \Upsilon\left(\mathbf{x}^*\right)\right\| g_k(\mathbf{x})\nu(d\mathbf{x}) \\ &\leq \zeta + \int_{A_\delta \cap \mathcal{C}} \left\|\Upsilon(\mathbf{x}) - \Upsilon\left(\mathbf{x}^*\right)\right\| g_k(\mathbf{x})\nu(d\mathbf{x}), \quad \forall k > \mathcal{N}. \end{aligned} \tag{4.21}$$

The rest of the proof amounts to showing that the second term in (4.21) is also bounded. Clearly the term $\|\Upsilon(\mathbf{x}) - \Upsilon(\mathbf{x}^*)\|$ is bounded on the set $A_\delta \cap \mathcal{C}$. We only need to find a bound for $g_k(\mathbf{x})$.

By Assumption A2, we have

$$\sup_{\mathbf{x}\in A_\delta \cap \mathcal{C}} J(\mathbf{x}) \leq \sup_{\mathbf{x}\in A_\delta} J(\mathbf{x}) < J\left(\mathbf{x}^*\right).$$

Define $\mathcal{H}_\delta := \mathcal{H}^* - \mathcal{H}(\sup_{\mathbf{x}\in A_\delta} J(\mathbf{x}))$. Since $\mathcal{H}$ is strictly increasing, we have $\mathcal{H}_\delta > 0$. Thus, it follows that

$$\mathcal{H}\left(J(\mathbf{x})\right) \leq \mathcal{H}^* - \mathcal{H}_\delta, \quad \forall \mathbf{x} \in A_\delta \cap \mathcal{C}. \tag{4.22}$$

On the other hand, from inequality (4.17) and Eq. (4.19), there exists $\bar{\mathcal{N}} > \mathcal{N}$ such that $\forall k \geq \bar{\mathcal{N}}$

$$E_{g_k}\left[\mathcal{H}\left(J(\mathbf{X})\right) I\left\{J(\mathbf{X}) \geq \bar{\chi}_k\right\}\right] \geq \mathcal{H}^* - \frac{1}{2}\mathcal{H}_\delta. \tag{4.23}$$

Observe that $g_k(\mathbf{x})$ can be alternatively expressed as

$$g_k(\mathbf{x}) = \prod_{i=\bar{\mathcal{N}}}^{k-1} \frac{\mathcal{H}(J(\mathbf{x}))I\{J(\mathbf{x}) \geq \bar{\chi}_i\}}{E_{g_i}[\mathcal{H}(J(\mathbf{X}))I\{J(\mathbf{X}) \geq \bar{\chi}_i\}]} \cdot g_{\bar{\mathcal{N}}}(\mathbf{x}), \quad \forall k \geq \bar{\mathcal{N}}.$$

Thus, it follows from inequalities (4.22) and (4.23) that

$$g_k(\mathbf{x}) \le \left(\frac{\mathcal{H}^* - \mathcal{H}_\delta}{\mathcal{H}^* - \mathcal{H}_\delta/2}\right)^{k-\bar{\mathcal{N}}} \cdot g_{\bar{\mathcal{N}}}(\mathbf{x}), \quad \forall \mathbf{x} \in A_\delta \cap \mathcal{C},\ \forall k \ge \bar{\mathcal{N}}.$$

Therefore,

$$\begin{aligned}\left\| E_{g_k}\left[\Upsilon(\mathbf{X})\right] - \Upsilon\left(\mathbf{x}^*\right)\right\| &\le \zeta + \sup_{\mathbf{x}\in A_\delta \cap \mathcal{C}} \left\|\Upsilon(\mathbf{x}) - \Upsilon\left(\mathbf{x}^*\right)\right\| \int_{A_\delta \cap \mathcal{C}} g_k(\mathbf{x})\nu(d\mathbf{x}) \\ &\le \zeta + \sup_{\mathbf{x}\in A_\delta \cap \mathcal{C}} \left\|\Upsilon(\mathbf{x}) - \Upsilon\left(\mathbf{x}^*\right)\right\| \left(\frac{\mathcal{H}^* - \mathcal{H}_\delta}{\mathcal{H}^* - \mathcal{H}_\delta/2}\right)^{k-\bar{\mathcal{N}}} \\ &\le \left(1 + \sup_{\mathbf{x}\in A_\delta \cap \mathcal{C}} \left\|\Upsilon(\mathbf{x}) - \Upsilon\left(\mathbf{x}^*\right)\right\|\right)\zeta \quad \forall k \ge \hat{\mathcal{N}},\end{aligned}$$

where $\hat{\mathcal{N}}$ is given by $\hat{\mathcal{N}} := \max\{\bar{\mathcal{N}},\ \lceil \bar{\mathcal{N}} + \ln\zeta / \ln(\frac{\mathcal{H}^* - \mathcal{H}_\delta}{\mathcal{H}^* - \mathcal{H}_\delta/2})\rceil\}$.

Since ζ is arbitrary, we have

$$\lim_{k\to\infty} E_{g_k}\left[\Upsilon(\mathbf{X})\right] = \Upsilon\left(\mathbf{x}^*\right).$$

Finally, the proof is completed by applying Lemma 4.4 to both Cases 1 and 2. □

Note that for many NEFs used in practice, Υ is a one-to-one mapping, in which case the convergence result (4.15) can be equivalently written as $\Upsilon^{-1}(\lim_{k\to\infty} m(\theta_k)) = \mathbf{x}^*$. Also note that for some particular distributions, the solution vector $\mathbf{x}$ itself will be a component of $\Upsilon(\mathbf{x})$ (e.g., multivariate normal distribution). Under these circumstances, we can interpret (4.15) as $\lim_{k\to\infty} m(\theta_k) = \mathbf{x}^*$. Another special case of particular interest is when the components of the random vector $\mathbf{X} = (\mathsf{X}_1, \ldots, \mathsf{X}_n)$ are independent, i.e., each has a univariate p.d.f./p.m.f. of the form

$$f(\mathsf{x}_i, \vartheta_i) = \exp\big(\mathsf{x}_i\vartheta_i - K(\vartheta_i)\big)h(\mathsf{x}_i), \quad \mathsf{x}_i \in \Re,\ \vartheta_i \in \Re,\ \forall i = 1, \ldots, n.$$

In this case, since the distribution of the random vector $\mathbf{X}$ is simply the product of the marginal distributions, we will clearly have $\Upsilon(\mathbf{x}) = \mathbf{x}$. Thus, (4.15) is again equivalent to $\lim_{k\to\infty} m(\theta_k) = \mathbf{x}^*$, where $\theta_k := (\vartheta_1^k, \ldots, \vartheta_n^k)$, and ϑ_i^k is the value of ϑ_i at the kth iteration.

As mentioned in Sect. 4.1, for problems with *finite* solution spaces, Assumptions A1 and A2 are automatically satisfied. Furthermore, if we take the input parameter $\varepsilon = 0$, then Step 2 of MRAS$_0$ is equivalent to setting $\bar{\chi}_k = \max_{0\le i\le k} \chi_i$. Thus, $\{\bar{\chi}_k\}$ is non-decreasing and each strict increment in the sequence is bounded from below by

$$\min_{\substack{J(\mathbf{x})\neq J(\mathbf{y}) \\ \mathbf{x},\mathbf{y}\in\mathcal{X}}} \left|J(\mathbf{x}) - J(\mathbf{y})\right|.$$

Therefore, the $\varepsilon > 0$ assumption in Theorem 4.5 can be relaxed to $\varepsilon \geq 0$.

As a result, we have the following results for the multivariate normal and independent univariate cases.

Corollary 4.6 (Multivariate Normal) *For continuous optimization problems in* $\Re^n$, *if multivariate normal p.d.f.s given by* (4.7) *are used in* MRAS$_0$, *where* $\theta_k := (\mu_k;\ \Sigma_k)$, $\varepsilon > 0$, *and Assumptions* A1–A4 *are satisfied, then*

$$\lim_{k\to\infty} \mu_k = \mathbf{x}^*, \quad \textit{and} \quad \lim_{k\to\infty} \Sigma_k = 0_{n\times n},$$

where $0_{n\times n}$ *represents an n-by-n zero matrix.*

Proof By Lemma 4.4, it is easy to show that

$$\mu_{k+1} = E_{g_{k+1}}(\mathbf{X}), \quad \forall k = 0, 1, \ldots,$$

and

$$\Sigma_{k+1} = E_{g_{k+1}}\left[(\mathbf{X} - \mu_{k+1})(\mathbf{X} - \mu_{k+1})^T\right], \quad \forall k = 0, 1, \ldots.$$

The rest of the proof amounts to showing that

$$\lim_{k\to\infty} E_{g_k}(\mathbf{X}) = \mathbf{x}^*, \quad \text{and} \quad \lim_{k\to\infty} E_{g_k}\left[(\mathbf{X} - \mu_k)(\mathbf{X} - \mu_k)^T\right] = 0_{n\times n},$$

which is the same as the proof of Theorem 4.5. □

Corollary 4.6 shows that in the multivariate normal case, the sequence of parameterized p.d.f.s will converge to a degenerate p.d.f. with all probability mass at the global optimum.

Corollary 4.7 (Independent Univariate) *If the components of the random vector* $\mathbf{X} = (\mathsf{X}_1, \ldots, \mathsf{X}_n)$ *used in* MRAS$_0$ *are independent, each has a univariate NEF distribution of the form*

$$f(\mathsf{x}_i, \vartheta_i) = \exp\big(\mathsf{x}_i\vartheta_i - K(\vartheta_i)\big)h(\mathsf{x}_i), \quad \mathsf{x}_i \in \Re,\ \vartheta_i \in \Re,\ \forall i = 1, \ldots, n,$$

$\varepsilon > 0$, *and Assumptions* A1–A4 *are satisfied, then*

$$\lim_{k\to\infty} m(\theta_k) := \lim_{k\to\infty} E_{\theta_k}[\mathbf{X}] = \mathbf{x}^*, \quad \textit{where } \theta_k := \left(\vartheta_1^k, \ldots, \vartheta_n^k\right).$$

4.2.2 MRAS$_1$ *Convergence*

To establish convergence properties of MRAS$_1$, we show that with high probability, the gaps between MRAS$_0$ and MRAS$_1$ (e.g., approximation errors incurred by replacing expected values with sample averages) can be made small enough such that

the convergence analysis of MRAS_1 can be ascribed to the convergence analysis of MRAS_0. As before, $P_{\tilde{\theta}_k}$ and $E_{\tilde{\theta}_k}$ denote the respective probability and expectation taken with respect to the distribution $f(\cdot, \tilde{\theta}_k)$, and we also let $\tilde{P}_{\tilde{\theta}_k}$ and $\tilde{E}_{\tilde{\theta}_k}$ denote the respective probability and expectation taken with respect to $\tilde{f}(\cdot, \tilde{\theta}_k)$. Note that since the sequence $\{\tilde{\theta}_k\}$ results from random samples generated at each iteration of MRAS_1, these quantities are also random.

To establish convergence, we assume the following conditions on the initial distribution $f(\cdot, \theta_0)$ and the parameter update.

Assumption A3′ There exists a compact set Ξ_ε such that $\{\mathbf{x} : J(\mathbf{x}) \geq J(\mathbf{x}^*) - \varepsilon\} \cap \mathcal{X} \subseteq \Xi_\varepsilon$. Moreover, the initial distribution $f(\mathbf{x}, \theta_0)$ is bounded away from zero on Ξ_ε, i.e., $f_* := \inf_{\mathbf{x} \in \Xi_\varepsilon} f(\mathbf{x}, \theta_0) > 0$.

Assumption A4′ The parameter vector $\tilde{\theta}_{k+1}$ computed via (4.10) at Step 4 of MRAS_1 is an interior point of Θ for all k.

Let $\tilde{g}_{k+1}$, $k = 0, 1, \ldots$, be defined by

$$\tilde{g}_{k+1}(\mathbf{x}) := \begin{cases} \frac{[[\mathcal{H}(J(\mathbf{x}))]^k / \tilde{f}(\mathbf{x}, \tilde{\theta}_k)] I\{J(\mathbf{x}) \geq \bar{\chi}_k\}}{\sum_{\mathbf{x} \in \Lambda_k} [[\mathcal{H}(J(\mathbf{x}))]^k / \tilde{f}(\mathbf{x}, \tilde{\theta}_k)] I\{J(\mathbf{x}) \geq \bar{\chi}_k\}} \\ \quad \text{if } \{\mathbf{x} \in \Lambda_k : J(\mathbf{x}) \geq \bar{\chi}_k\} \neq \emptyset, \\ \tilde{g}_k(\mathbf{x}) \quad \text{otherwise}, \end{cases} \tag{4.24}$$

where $\bar{\chi}_k$ is given by

$$\bar{\chi}_k := \begin{cases} \tilde{\chi}_k(\rho_k, N_k) & \text{if Step 3a is visited,} \\ \tilde{\chi}_k(\bar{\rho}, N_k) & \text{if Step 3b is visited,} \\ \bar{\chi}_{k-1} & \text{if Step 3c is visited.} \end{cases}$$

The following lemma shows the connection between $f(\cdot, \tilde{\theta}_{k+1})$ and $\tilde{g}_{k+1}$; the proof is similar to the proof of Lemma 4.4, and is thus omitted here.

Lemma 4.8 *If Assumption A4′ holds, then the parameter $\tilde{\theta}_{k+1}$ computed via (4.10) at Step 4 of MRAS_1 satisfies*

$$m(\tilde{\theta}_{k+1}) := E_{\tilde{\theta}_{k+1}}\big[\Upsilon(\mathbf{X})\big] = E_{\tilde{g}_{k+1}}\big[\Upsilon(\mathbf{X})\big], \quad k = 0, 1, \ldots.$$

Note that the region $\{\mathbf{x} : J(\mathbf{x}) \geq \bar{\chi}_k\}$ will become smaller and smaller as $\bar{\chi}_k$ increases. Lemma 4.8 shows that the sequence of sampling distributions $\{f(\cdot, \tilde{\theta}_{k+1})\}$ is adapted to this sequence of shrinking regions. For example, consider the case where $\{\mathbf{x} : J(\mathbf{x}) \geq \bar{\chi}_k\}$ is convex and $\Upsilon(\mathbf{x}) = \mathbf{x}$. Since $E_{\tilde{g}_{k+1}}[\mathbf{X}]$ is a convex combination of $\mathbf{X}_k^1, \ldots, \mathbf{X}_k^{N_k}$, the lemma implies that $E_{\tilde{\theta}_{k+1}}[\mathbf{X}] \in \{\mathbf{x} : J(\mathbf{x}) \geq \bar{\chi}_k\}$. Thus, it is natural to expect that the random samples generated at the next iteration will fall in the region $\{\mathbf{x} : J(\mathbf{x}) \geq \bar{\chi}_k\}$ with large probabilities (e.g., consider the normal

p.d.f. where its mode is equal to its mean). In contrast, if we use a fixed sampling distribution for all iterations as in pure random sampling (i.e., the $\lambda = 1$ case), then sampling from this sequence of shrinking regions could become a substantially difficult problem in practice.

Next, we present a useful intermediate result, which shows the convergence of the quantile estimates when random samples are generated from a sequence of different distributions.

Lemma 4.9 *For any given $\rho \in (0, 1)$, let χ_k be the set of true $(1-\rho)$-quantiles of $J(\mathbf{X})$ with respect to $\tilde{f}(\cdot, \tilde{\theta}_k)$, and let $\tilde{\chi}_k(\rho, N_k)$ be the corresponding sample quantile of $J(\mathbf{X}_k^1), \ldots, J(\mathbf{X}_k^{N_k})$, where $\tilde{f}(\cdot, \tilde{\theta}_k)$ and N_k are defined as in* MRAS$_1$, *and $\mathbf{X}_k^1, \ldots, \mathbf{X}_k^{N_k}$ are i.i.d. with common distribution $\tilde{f}(\cdot, \tilde{\theta}_k)$. Then the distance from $\tilde{\chi}_k(\rho, N_k)$ to χ_k tends to zero as $k \to \infty$ w.p.*1.

Proof Our proof is based on the proof of Lemma A1 in [151]. Notice that for given ρ and $\tilde{f}(\cdot, \tilde{\theta}_k)$, χ_k can be obtained as the optimal solution of the following problem (see [87]):

$$\min_{v \in \mathcal{V}} \ell_k(v), \tag{4.25}$$

where $\mathcal{V} = [0, J(\mathbf{x}^*)]$, $\ell_k(v) := \tilde{E}_{\tilde{\theta}_k} \phi(J(\mathbf{X}), v)$, and

$$\phi\big(J(\mathbf{x}), v\big) := \begin{cases} (1-\rho)(J(\mathbf{x}) - v) & \text{if } v \le J(\mathbf{x}), \\ \rho(v - J(\mathbf{x})) & \text{if } v \ge J(\mathbf{x}). \end{cases}$$

Similarly, the sample quantile $\tilde{\chi}_k(\rho, N_k)$ can be expressed as the solution to the sample average approximation of (4.25),

$$\min_{v \in \mathcal{V}} \bar{\ell}_k(v), \tag{4.26}$$

where $\bar{\ell}_k(v) := \frac{1}{N_k} \sum_{j=1}^{N_k} \phi(J(\mathbf{X}_k^j), v)$ and $\mathbf{X}_k^1, \ldots, \mathbf{X}_k^{N_k}$ are i.i.d. with density $\tilde{f}(\cdot, \tilde{\theta}_k)$.

Since the function $\phi(J(\mathbf{x}), v)$ is bounded and continuous on $\mathcal{V}$ for all $\mathbf{x} \in \mathcal{X}$, it is not difficult to show that $\ell_k(v)$ is continuous on $\mathcal{V}$.

Now consider a point $v \in \mathcal{V}$ and let $B_i \subseteq \mathcal{V}$ be a sequence of open balls containing v such that $B_{i+1} \subseteq B_i$ $\forall i$ and $\lim_{L \to \infty} \cap_{i=1}^{L} B_i = v$. Define the function

$$b_i\big(J(\mathbf{x})\big) := \sup\big\{\big|\phi\big(J(\mathbf{x}), u\big) - \phi\big(J(\mathbf{x}), v\big)\big| : u \in B_i\big\}.$$

From the dominated convergence theorem,

$$\lim_{i \to \infty} \tilde{E}_{\tilde{\theta}_k}\big[b_i\big(J(\mathbf{X})\big)\big] = \tilde{E}_{\tilde{\theta}_k}\Big[\lim_{i \to \infty} b_i\big(J(\mathbf{X})\big)\Big] = 0 \quad \forall k = 1, 2, \ldots, \tag{4.27}$$

where the last equality follows because $\phi(J(\mathbf{x}), v)$ is continuous on $\mathcal{V}$.

Since $|\bar{\ell}_k(u) - \bar{\ell}_k(v)| \le \frac{1}{N_k}\sum_{j=1}^{N_k} |\phi(J(\mathbf{X}_k^j), u) - \phi(J(\mathbf{X}_k^j), v)|$, it follows that

$$\sup_{u\in B_i} \left|\bar{\ell}_k(u) - \bar{\ell}_k(v)\right| \le \frac{1}{N_k}\sum_{j=1}^{N_k} b_i\big(J\big(\mathbf{X}_k^j\big)\big). \tag{4.28}$$

We now show that $\frac{1}{N_k}\sum_{j=1}^{N_k} b_i(J(\mathbf{X}_k^j)) \to \tilde{E}_{\tilde{\theta}_k}[b_i(J(\mathbf{X}))]$ as $k\to\infty$ w.p.1.

Let M be an upper bound for $b_i(J(\mathbf{x}))$, and let $\mathcal{T}_\varepsilon := \lceil (2[J(\mathbf{x}^*) - \mathcal{M}])/\varepsilon \rceil$, where $\mathcal{M}$ is a lower bound for the function $J(\mathbf{x})$, and ε is defined as in the MRAS$_1$ algorithm. Note that the total number of visits to Step 3a and 3b of MRAS$_1$ is bounded by $\mathcal{T}_\varepsilon$, thus for any $k > \mathcal{T}_\varepsilon$, the total number of visits to Step 3c is greater than $k - \mathcal{T}_\varepsilon$. Since conditional on $\tilde{\theta}_k$, $\frac{1}{N_k}\sum_{j=1}^{N_k} b_i(J(\mathbf{X}_k^j))$ is an unbiased estimate of $\tilde{E}_{\tilde{\theta}_k}[b_i(J(\mathbf{X}))]$, by the Hoeffding inequality [86], for any $\zeta > 0$,

$$P\left(\left|\frac{1}{N_k}\sum_{j=1}^{N_k} b_i\big(J\big(\mathbf{X}_k^j\big)\big) - \tilde{E}_{\tilde{\theta}_k}\big[b_i\big(J(\mathbf{X})\big)\big]\right| > \zeta \,\middle|\, \tilde{\theta}_k = \theta\right) \le 2\exp\left(\frac{-2N_k\zeta^2}{M^2}\right) \quad \forall k.$$

Therefore,

$$\begin{aligned}
&P\left(\left|\frac{1}{N_k}\sum_{j=1}^{N_k} b_i\big(J\big(\mathbf{X}_k^j\big)\big) - \tilde{E}_{\tilde{\theta}_k}\big[b_i\big(J(\mathbf{X})\big)\big]\right| > \zeta\right)\\
&\le 2\exp\left(\frac{-2N_k\zeta^2}{M^2}\right) \quad \forall k\\
&\le 2\exp\left(\frac{-2\alpha^{k-\mathcal{T}_\varepsilon}N_0\zeta^2}{M^2}\right) \quad \forall k > \mathcal{T}_\varepsilon\\
&\longrightarrow 0 \quad \text{as } k\to\infty, \text{ since } \alpha > 1.
\end{aligned}$$

Furthermore, it is easy to see that

$$\sum_{k=1}^{\infty} P\left(\left|\frac{1}{N_k}\sum_{j=1}^{N_k} b_i\big(J\big(\mathbf{X}_k^j\big)\big) - \tilde{E}_{\tilde{\theta}_k}\big[b_i\big(J(\mathbf{X})\big)\big]\right| > \zeta\right) < \infty.$$

Thus, by the Borel–Cantelli lemma

$$P\left(\left|\frac{1}{N_k}\sum_{j=1}^{N_k} b_i\big(J\big(\mathbf{X}_k^j\big)\big) - \tilde{E}_{\tilde{\theta}_k}\big[b_i\big(J(\mathbf{X})\big)\big]\right| > \zeta \text{ i.o.}\right) = 0,$$

which implies that $\frac{1}{N_k}\sum_{j=1}^{N_k} b_i(J(\mathbf{X}_k^j)) \to \tilde{E}_{\tilde{\theta}_k}[b_i(J(\mathbf{X}))]$ as $k\to\infty$ w.p.1. Note that by using a similar argument as above, we can also show that $\bar{\ell}_k(v) \to \ell_k(v)$ w.p.1 as $k\to\infty$.

The above result together with (4.27) and (4.28) implies that for any $\delta > 0$, there exists a small neighborhood B_v of v such that

$$\sup\left\{\left|\bar{\ell}_k(u) - \bar{\ell}_k(v)\right| : u \in B_v\right\} < \delta \quad \text{w.p.1}$$

for k sufficiently large. Since this holds for all $v \in \mathcal{V}$, we have $\mathcal{V} \subseteq \bigcup_{v\in\mathcal{V}} B_v$. Because $\mathcal{V}$ is compact, there exists a sequence of finite subcovers $B_{v_1}, \ldots, B_{v_m}$ such that

$$\sup\left\{\left|\bar{\ell}_k(u) - \bar{\ell}_k(v_j)\right| : u \in B_{v_j}\right\} < \delta \quad \text{w.p.1}$$

for k sufficiently large, $\mathcal{V} \subseteq \cup_{j=1}^m B_{v_j}$. Furthermore, by the continuity of $\ell_k(v)$, these open balls can be chosen in such a way that

$$\sup\left\{\left|\ell_k(u) - \ell_k(v_j)\right| : u \in B_{v_j}\right\} < \delta \quad \forall j = 1, \ldots, m.$$

Since $\bar{\ell}_k(v_j) \to \ell_k(v_j)$ w.p.1 as $k \to \infty$ for all $j = 1, \ldots, m$,

$$\left|\bar{\ell}_k(v_j) - \ell_k(v_j)\right| < \delta \quad \text{w.p.1}$$

for k sufficiently large, $\forall\, j = 1, \ldots, m$. For any $v \in \mathcal{V}$, without loss of generality assuming $v \in B_{v_j}$, we have w.p.1 for k sufficiently large,

$$\left|\bar{\ell}_k(v) - \ell_k(v)\right| \le \left|\bar{\ell}_k(v) - \bar{\ell}_k(v_j)\right| + \left|\ell_k(v) - \ell_k(v_j)\right| + \left|\bar{\ell}_k(v_j) - \ell_k(v_j)\right| < 3\delta,$$

which implies that $\bar{\ell}_k(v) \to \ell_k(v)$ uniformly w.p.1 on $\mathcal{V}$.

The rest of the proof follows from Theorem A1 in [151, p. 69], which basically states that if $\bar{\ell}_k(v) \to \ell_k(v)$ uniformly w.p.1, then the distance from $\tilde{\chi}_k(\rho, N_k)$ to χ_k tends to zero w.p.1 as $k \to \infty$. □

We now state the main theorem.

Theorem 4.10 *Let $\varepsilon > 0$, and define the ε-optimal set $\mathcal{O}_\varepsilon := \{\mathbf{x} : J(\mathbf{x}) \ge J(\mathbf{x}^*) - \varepsilon\} \cap \mathcal{X}$. If Assumptions A1, A3′, and A4′ are satisfied, then there exists a random variable $\mathcal{K}$ such that w.p.1, $\mathcal{K} < \infty$, and:*

(i) $\bar{\chi}_k > J(\mathbf{x}^*) - \varepsilon,\ \forall k \ge \mathcal{K}$.
(ii) $m(\tilde{\theta}_{k+1}) := E_{\tilde{\theta}_{k+1}}[\Upsilon(\mathbf{X})] \in \text{CONV}\{\Upsilon(\mathcal{O}_\varepsilon)\},\ \forall k \ge \mathcal{K}$, *where* $\text{CONV}\{\Upsilon(\mathcal{O}_\varepsilon)\}$ *indicates the convex hull of the set* $\Upsilon(\mathcal{O}_\varepsilon)$.

Furthermore, let β be a positive constant satisfying the condition that the set $\{\mathbf{x} : \mathcal{H}(J(\mathbf{x})) \ge \frac{1}{\beta}\}$ has a strictly positive Lebesgue/counting measure. If Assumptions A1, A2, A3′, and A4′ are all satisfied and $\alpha > (\beta\mathcal{H}^*)^2$, where $\mathcal{H}^* := \mathcal{H}(J(\mathbf{x}^*))$, then

(iii) $\lim_{k\to\infty} m(\tilde{\theta}_k) := \lim_{k\to\infty} E_{\tilde{\theta}_k}[\Upsilon(\mathbf{X})] = \Upsilon(\mathbf{x}^*)$ *w.p.1.*

Proof Part (i). The first part of the proof is an extension of the proofs given in [87]. First we claim that given ρ_k and $\bar{\chi}_{k-1}$, if $\bar{\chi}_{k-1} \leq J(\mathbf{x}^*) - \varepsilon$, then $\exists \bar{\mathcal{K}} < \infty$ w.p.1 and $\bar{\rho} \in (0, \rho_k)$ such that $\tilde{\chi}_{k'}(\bar{\rho}, N_{k'}) \geq \bar{\chi}_{k-1} + \frac{\varepsilon}{2}$ $\forall k' \geq \bar{\mathcal{K}}$. To show this, we proceed by contradiction.

Let $\rho_k^* := \tilde{P}_{\tilde{\theta}_k}(J(\mathbf{X}) \geq \bar{\chi}_{k-1} + \frac{2\varepsilon}{3})$. If $\bar{\chi}_{k-1} \leq J(\mathbf{x}^*) - \varepsilon$, then $\bar{\chi}_{k-1} + \frac{2\varepsilon}{3} \leq J(\mathbf{x}^*) - \frac{\varepsilon}{3}$. By Assumptions A1 and A3′, we have

$$\rho_k^* \geq \tilde{P}_{\tilde{\theta}_k}\left(J(\mathbf{X}) \geq J\left(\mathbf{x}^*\right) - \frac{\varepsilon}{3}\right) \geq \lambda \mathcal{C}(\varepsilon, \theta_0) > 0, \tag{4.29}$$

where $\mathcal{C}(\varepsilon, \theta_0) = \int_{\mathcal{X}} I\{J(\mathbf{x}) \geq J(\mathbf{x}^*) - \varepsilon/3\} f(\mathbf{x}, \theta_0) \nu(d\mathbf{x})$ is a constant.

Now assume that $\exists \rho \in (0, \rho_k^*)$ such that $\chi_k(\rho, \tilde{\theta}_k) < \bar{\chi}_{k-1} + \frac{2\varepsilon}{3}$, where $\chi_k(\rho, \tilde{\theta}_k)$ is the true $(1-\rho)$-quantile of $J(\mathbf{X})$ with respect to $\tilde{f}(\cdot, \tilde{\theta}_k)$. By the definition of quantiles, we have the following two inequalities:

$$\begin{aligned} &\tilde{P}_{\tilde{\theta}_k}\left(J(\mathbf{X}) \geq \chi_k(\rho, \tilde{\theta}_k)\right) \geq \rho, \\ &\tilde{P}_{\tilde{\theta}_k}\left(J(\mathbf{X}) \leq \chi_k(\rho, \tilde{\theta}_k)\right) \geq 1 - \rho > 1 - \rho_k^*. \end{aligned} \tag{4.30}$$

It follows that $\tilde{P}_{\tilde{\theta}_k}(J(\mathbf{X}) \leq \chi_k(\rho, \tilde{\theta}_k)) \leq \tilde{P}_{\tilde{\theta}_k}(J(\mathbf{X}) < \bar{\chi}_{k-1} + \frac{2\varepsilon}{3}) = 1 - \rho_k^*$ by the definition of ρ_k^*, which contradicts (4.30); thus, if $\bar{\chi}_{k-1} \leq J(\mathbf{x}^*) - \varepsilon$, then we must have

$$\chi_k(\rho, \tilde{\theta}_k) \geq \bar{\chi}_{k-1} + \frac{2\varepsilon}{3}, \quad \forall \rho \in \left(0, \rho_k^*\right).$$

Therefore by (4.29), there exists $\bar{\rho} \in (0, \min\{\rho_k, \lambda \mathcal{C}(\varepsilon, \theta_0)\}) \subseteq (0, \rho_k)$ such that $\chi_k(\bar{\rho}, \tilde{\theta}_k) \geq \bar{\chi}_{k-1} + \frac{2\varepsilon}{3}$ whenever $\bar{\chi}_{k-1} \leq J(\mathbf{x}^*) - \varepsilon$. By Lemma 4.9, the distance from the sample $(1-\bar{\rho})$-quantile $\tilde{\chi}_k(\bar{\rho}, N_k)$ to the set of $(1-\bar{\rho})$-quantiles $\chi_k(\bar{\rho}, \tilde{\theta}_k)$ goes to zero as $k \to \infty$ w.p.1, thus $\exists \bar{\mathcal{K}} < \infty$ w.p.1 such that $\tilde{\chi}_{k'}(\bar{\rho}, N_{k'}) \geq \bar{\chi}_{k-1} + \frac{\varepsilon}{2}$ $\forall k' \geq \bar{\mathcal{K}}$.

Notice that from the MRAS$_1$ algorithm, if neither Step 3a nor 3b is visited at the kth iteration, we will have $\rho_{k+1} = \rho_k$ and $\bar{\chi}_k = \bar{\chi}_{k-1}$. Thus, whenever $\bar{\chi}_{k-1} \leq J(\mathbf{x}^*) - \varepsilon$, Step 3a or 3b will be visited w.p.1 after a finite number of iterations. Furthermore, since the total number of visits to Steps 3a and 3b is finite (i.e., bounded by $(2[J(\mathbf{x}^*) - \mathcal{M}])/\varepsilon$, where recall that $\mathcal{M}$ is a lower bound for $J(\mathbf{x})$), we conclude that there exists $\mathcal{K} < \infty$ w.p.1 such that

$$\bar{\chi}_k > J\left(\mathbf{x}^*\right) - \varepsilon, \quad \forall k \geq \mathcal{K} \text{ w.p.1}.$$

Part (ii). From the MRAS$_1$ algorithm, it is easy to see that $\bar{\chi}_k \geq \bar{\chi}_{k-1}$, $\forall k = 1, 2, \ldots$. By Part (i), we have $\bar{\chi}_k \geq J(\mathbf{x}^*) - \varepsilon$, $\forall k \geq \mathcal{K}$ w.p.1. Thus, by the definition of $\tilde{g}_{k+1}(\mathbf{x})$ (see Eq. (4.24)), it follows immediately that if the set $\{\mathbf{x} \in \Lambda_k : J(\mathbf{x}) \geq \bar{\chi}_k\} \neq \emptyset$, then the support of $\tilde{g}_{k+1}(\mathbf{x})$ satisfies $\text{supp}\{\tilde{g}_{k+1}\} \subseteq \mathcal{O}_\varepsilon$ $\forall k \geq \mathcal{K}$ w.p.1; otherwise we will have $\text{supp}\{\tilde{g}_{k+1}\} = \emptyset$. We now discuss these two cases separately.

Case 1. If $\text{supp}\{\tilde{g}_{k+1}\} \subseteq \mathcal{O}_\varepsilon$, then we have $\{\Upsilon(\text{supp}\{\tilde{g}_{k+1}\})\} \subseteq \{\Upsilon(\mathcal{O}_\varepsilon)\}$. Since $E_{\tilde{g}_{k+1}}[\Upsilon(\mathbf{X})]$ is the convex combination of $\Upsilon(\mathbf{X}_k^1), \ldots, \Upsilon(\mathbf{X}_k^{N_k})$, it follows that

$$E_{\tilde{g}_{k+1}}\big[\Upsilon(\mathbf{X})\big] \in \text{CONV}\big\{\Upsilon\big(\text{supp}\{\tilde{g}_{k+1}\}\big)\big\} \subseteq \text{CONV}\big\{\Upsilon(\mathcal{O}_\varepsilon)\big\}.$$

Thus by Assumption A4′ and Lemma 4.8,

$$E_{\tilde{\theta}_{k+1}}\big[\Upsilon(\mathbf{X})\big] \in \text{CONV}\big\{\Upsilon(\mathcal{O}_\varepsilon)\big\}.$$

Case 2. If $\text{supp}\{\tilde{g}_{k+1}\} = \emptyset$ (note that this could only happen if Step 3c is visited), then from the algorithm, there exists some $\hat{k} < k+1$ such that $\bar{\chi}_k = \bar{\chi}_{\hat{k}}$ and $\text{supp}\{\tilde{g}_{\hat{k}}\} \neq \emptyset$. Without loss of generality, let $\hat{k}$ be the largest iteration counter such that the preceding properties hold. Since $\bar{\chi}_{\hat{k}} = \bar{\chi}_k > J(\mathbf{x}^*) - \varepsilon \ \forall k \geq \mathcal{K}$ w.p.1, we have $\text{supp}\{\tilde{g}_{\hat{k}}\} \subseteq \mathcal{O}_\varepsilon$ w.p.1. By following the discussion in Case 1, it is clear that

$$E_{\tilde{\theta}_{\hat{k}}}\big[\Upsilon(\mathbf{X})\big] \in \text{CONV}\big\{\Upsilon(\mathcal{O}_\varepsilon)\big\} \quad \text{w.p.1.}$$

Furthermore, since $\tilde{\theta}_{\hat{k}} = \tilde{\theta}_{\hat{k}+1} = \cdots = \tilde{\theta}_{k+1}$, we will again have

$$E_{\tilde{\theta}_{k+1}}\big[\Upsilon(\mathbf{X})\big] \in \text{CONV}\big\{\Upsilon(\mathcal{O}_\varepsilon)\big\}, \quad \forall k \geq \mathcal{K} \text{ w.p.1.}$$

Part (iii). Define $\hat{g}_{k+1}(\mathbf{x})$ as

$$\hat{g}_{k+1}(\mathbf{x}) := \frac{[\mathcal{H}(J(\mathbf{x}))]^k I\{J(\mathbf{x}) \geq \bar{\chi}_{k-1}\}}{\int_{\mathcal{X}} [\mathcal{H}(J(\mathbf{x}))]^k I\{J(\mathbf{x}) \geq \bar{\chi}_{k-1}\} \nu(d\mathbf{x})}, \quad \forall k = 1, 2, \ldots,$$

where $\{\bar{\chi}_k, k = 0, 1, \ldots\}$ is defined as in MRAS$_1$. Note that since $\bar{\chi}_k$ is a random variable, $\hat{g}_{k+1}(\mathbf{x})$ is also a random variable. It follows that

$$E_{\hat{g}_{k+1}}\big[\Upsilon(\mathbf{X})\big] = \frac{\int_{\mathcal{X}} [\beta\mathcal{H}(J(\mathbf{x}))]^k I\{J(\mathbf{x}) \geq \bar{\chi}_{k-1}\} \Upsilon(\mathbf{x}) \nu(d\mathbf{x})}{\int_{\mathcal{X}} [\beta\mathcal{H}(J(\mathbf{x}))]^k I\{J(\mathbf{x}) \geq \bar{\chi}_{k-1}\} \nu(d\mathbf{x})}.$$

Let $\omega = (\mathbf{X}_0^1, \ldots, \mathbf{X}_0^{N_0}, \mathbf{X}_1^1, \ldots, \mathbf{X}_1^{N_1}, \ldots)$ be a particular sample path generated by the algorithm. For each ω, the sequence $\{\bar{\chi}_k(\omega), \ k = 0, 1, \ldots\}$ is nondecreasing and each strict increase is lower bounded by $\varepsilon/2$. Thus, $\exists \tilde{\mathcal{N}}(\omega) > 0$ such that $\bar{\chi}_k(\omega) = \bar{\chi}_{k-1}(\omega) \ \forall k \geq \tilde{\mathcal{N}}(\omega)$. Now define $\Omega_1 := \{\omega : \lim_{k\to\infty} \bar{\chi}_k(\omega) = J(\mathbf{x}^*)\}$. By the definition of $\tilde{g}_{k+1}$ (see Eq. (4.24)), for each $\omega \in \Omega_1$ we clearly have $\lim_{k\to\infty} E_{\tilde{g}_k(\omega)}[\Upsilon(\mathbf{X})] = \Upsilon(\mathbf{x}^*)$; thus, it follows from Lemma 4.8 that $\lim_{k\to\infty} E_{\tilde{\theta}_k(\omega)}[\Upsilon(\mathbf{X})] = \Upsilon(\mathbf{x}^*), \ \forall \omega \in \Omega_1$. The rest of the proof amounts to showing that the result also holds almost surely (a.s.) on the set Ω_1^c.

Since $\lim_{k\to\infty} \bar{\chi}_k(\omega) = \bar{\chi}_{\tilde{\mathcal{N}}}(\omega) < J(\mathbf{x}^*) \ \forall \omega \in \Omega_1^c$, by Fatou's lemma,

$$\begin{aligned} &\liminf_{k\to\infty} \int_{\mathcal{X}} \big[\beta\mathcal{H}\big(J(\mathbf{x})\big)\big]^k I\big\{J(\mathbf{x}) \geq \bar{\chi}_{k-1}\big\} \nu(d\mathbf{x}) \\ &\quad \geq \int_{\mathcal{X}} \liminf_{k\to\infty} \big[\beta\mathcal{H}\big(J(\mathbf{x})\big)\big]^k I\big\{J(\mathbf{x}) \geq \bar{\chi}_{k-1}\big\} \nu(d\mathbf{x}) > 0 \quad \forall \omega \in \Omega_1^c, \quad (4.31) \end{aligned}$$

where the last inequality follows from the fact that $\beta\mathcal{H}(J(\mathbf{x})) \geq 1\ \forall\, \mathbf{x} \in \{\mathbf{x} : J(\mathbf{x}) \geq \max\{\mathcal{H}^{-1}(\frac{1}{\beta}), \bar{\chi}_{\tilde{\mathcal{N}}}\}\}$ and Assumption A1.

Since $f(\mathbf{x}, \theta_0) > 0\ \forall \mathbf{x} \in \mathcal{X}$, we have $\mathcal{X} \subseteq \text{supp}\{\tilde{f}(\cdot, \tilde{\theta}_k)\}\ \forall k$; thus

$$E_{\hat{g}_{k+1}}\big[\Upsilon(\mathbf{X})\big] = \frac{\tilde{E}_{\tilde{\theta}_k}[\beta^k \tilde{\mathcal{H}}_k(J(\mathbf{X}))I\{J(\mathbf{X}) \geq \bar{\chi}_{k-1}\}\Upsilon(\mathbf{X})]}{\tilde{E}_{\tilde{\theta}_k}[\beta^k \tilde{\mathcal{H}}_k(J(\mathbf{X}))I\{J(\mathbf{X}) \geq \bar{\chi}_{k-1}\}]} \quad \forall k = 1, 2, \ldots,$$

where $\tilde{\mathcal{H}}_k(J(\mathbf{x})) := [\mathcal{H}(J(\mathbf{x}))]^k / \tilde{f}(\mathbf{x}, \tilde{\theta}_k)$. We now show that $E_{\tilde{g}_{k+1}}[\Upsilon(\mathbf{X})] \to E_{\hat{g}_{k+1}}[\Upsilon(\mathbf{X})]$ a.s. on Ω_1^c as $k \to \infty$. Since we are only interested in the limiting behavior of $E_{\tilde{g}_{k+1}}[\Upsilon(\mathbf{X})]$, it is sufficient to show that

$$\frac{\frac{1}{N_k}\sum_{\mathbf{x}\in\Lambda_k} \beta^k \tilde{\mathcal{H}}_k(J(\mathbf{x}))I\{J(\mathbf{x}) \geq \bar{\chi}_k\}\Upsilon(\mathbf{x})}{\frac{1}{N_k}\sum_{\mathbf{x}\in\Lambda_k} \beta^k \tilde{\mathcal{H}}_k(J(\mathbf{x}))I\{J(\mathbf{x}) \geq \bar{\chi}_k\}} \longrightarrow E_{\hat{g}_{k+1}}\big[\Upsilon(\mathbf{X})\big] \quad \text{a.s. on } \Omega_1^c,$$

where and hereafter, whenever $\{\mathbf{x} \in \Lambda_k : J(\mathbf{x}) \geq \bar{\chi}_k\} = \emptyset$, we define $\frac{0}{0} = 0$.

For brevity, we introduce the following shorthand notation:

$$\begin{aligned}
\hat{Y}^k(\chi) &:= \tilde{E}_{\tilde{\theta}_k}\big[\beta^k \tilde{\mathcal{H}}_k\big(J(\mathbf{X})\big) I\big\{J(\mathbf{X}) \geq \chi\big\}\big], \\
\hat{Y}^k_\Upsilon(\chi) &:= \tilde{E}_{\tilde{\theta}_k}\big[\beta^k \tilde{\mathcal{H}}_k\big(J(\mathbf{X})\big) I\big\{J(\mathbf{X}) \geq \chi\big\}\Upsilon(\mathbf{X})\big], \\
\hat{Y}^k(\mathbf{x}, \chi) &:= \beta^k \tilde{\mathcal{H}}_k\big(J(\mathbf{x})\big) I\big\{J(\mathbf{x}) \geq \chi\big\}.
\end{aligned}$$

We also let $\mathcal{T}_\varepsilon := \lceil (2[J(\mathbf{x}^*) - \mathcal{M}])/\varepsilon \rceil$. Note that the total number of visits to Steps 3a and 3b of MRAS$_1$ is bounded by $\mathcal{T}_\varepsilon$, thus for any $k > \mathcal{T}_\varepsilon$, the total number of visits to Step 3c is greater than $k - \mathcal{T}_\varepsilon$.

We have

$$\begin{aligned}
&\frac{\frac{1}{N_k}\sum_{\mathbf{x}\in\Lambda_k} \beta^k \tilde{\mathcal{H}}_k(J(\mathbf{x}))I\{J(\mathbf{x}) \geq \bar{\chi}_k\}\Upsilon(\mathbf{x})}{\frac{1}{N_k}\sum_{\mathbf{x}\in\Lambda_k} \beta^k \tilde{\mathcal{H}}_k(J(\mathbf{x}))I\{J(\mathbf{x}) \geq \bar{\chi}_k\}} - E_{\hat{g}_{k+1}}\big[\Upsilon(\mathbf{X})\big] \\
&= \left(\frac{\frac{1}{N_k}\sum_{\mathbf{x}\in\Lambda_k} \hat{Y}^k(\mathbf{x}, \bar{\chi}_k)\Upsilon(\mathbf{x})}{\frac{1}{N_k}\sum_{\mathbf{x}\in\Lambda_k} \hat{Y}^k(\mathbf{x}, \bar{\chi}_k)} - \frac{\frac{1}{N_k}\sum_{\mathbf{x}\in\Lambda_k} \hat{Y}^k(\mathbf{x}, \bar{\chi}_{k-1})\Upsilon(\mathbf{x})}{\frac{1}{N_k}\sum_{\mathbf{x}\in\Lambda_k} \hat{Y}^k(\mathbf{x}, \bar{\chi}_{k-1})} \right) \\
&\quad + \left(\frac{\frac{1}{N_k}\sum_{\mathbf{x}\in\Lambda_k} \hat{Y}^k(\mathbf{x}, \bar{\chi}_{k-1})\Upsilon(\mathbf{x})}{\frac{1}{N_k}\sum_{\mathbf{x}\in\Lambda_k} \hat{Y}^k(\mathbf{x}, \bar{\chi}_{k-1})} - \frac{\hat{Y}^k_\Upsilon(\bar{\chi}_{k-1})}{\hat{Y}^k(\bar{\chi}_{k-1})} \right).
\end{aligned}$$

Since for each $\omega \in \Omega_1^c$, $\bar{\chi}_k(\omega) = \bar{\chi}_{k-1}(\omega)\ \forall k \geq \tilde{\mathcal{N}}(\omega)$, it follows that the first term

$$\frac{\frac{1}{N_k}\sum_{\mathbf{x}\in\Lambda_k} \hat{Y}^k(\mathbf{x}, \bar{\chi}_k)\Upsilon(\mathbf{x})}{\frac{1}{N_k}\sum_{\mathbf{x}\in\Lambda_k} \hat{Y}^k(\mathbf{x}, \bar{\chi}_k)} - \frac{\frac{1}{N_k}\sum_{\mathbf{x}\in\Lambda_k} \hat{Y}^k(\mathbf{x}, \bar{\chi}_{k-1})\Upsilon(\mathbf{x})}{\frac{1}{N_k}\sum_{\mathbf{x}\in\Lambda_k} \hat{Y}^k(\mathbf{x}, \bar{\chi}_{k-1})} = 0, \quad \forall k \geq \tilde{\mathcal{N}}(\omega).$$

To show that the second term also converges to zero, we denote by $\mathcal{V}_k$ the event $\mathcal{V}_k = \{\bar{\chi}_{k-1} > J(\mathbf{x}^*) - \varepsilon\}$. For any $\zeta > 0$, we also let $\mathcal{C}_k$ be the event

$\mathcal{C}_k = \{|\frac{1}{N_k}\sum_{\mathbf{x}\in\Lambda_k}\hat{Y}^k(\mathbf{x},\bar{\chi}_{k-1}) - \hat{Y}^k(\bar{\chi}_{k-1})| > \zeta\}$. We have

$$\begin{aligned} P(\mathcal{C}_k \text{ i.o.}) &= P\left(\{\mathcal{C}_k \cap \mathcal{V}_k\} \cup \left\{\mathcal{C}_k \cap \mathcal{V}_k^c\right\} \text{ i.o.}\right) \\ &= P(\mathcal{C}_k \cap \mathcal{V}_k \text{ i.o.}), \quad \text{since } P\left(\mathcal{V}_k^c \text{ i.o.}\right) = 0 \text{ by Part (i).} \end{aligned} \tag{4.32}$$

It is easy to see that conditioned on $\tilde{\theta}_k$ and $\bar{\chi}_{k-1}$, $\{\hat{Y}^k(\mathbf{x},\bar{\chi}_{k-1}),\ \mathbf{x}\in\Lambda_k\}$ are i.i.d. and $E[\hat{Y}^k(\mathbf{x},\bar{\chi}_{k-1})|\tilde{\theta}_k,\bar{\chi}_{k-1})] = \hat{Y}^k(\bar{\chi}_{k-1})\ \forall \mathbf{x}\in\Lambda_k$. Furthermore, by Assumption A3′, conditioned on the event $\mathcal{V}_k$, the support $[a_k,b_k]$ of the random variable $\hat{Y}^k(\mathbf{x},\bar{\chi}_{k-1}),\ \mathbf{x}\in\Lambda_k$ satisfies $[a_k,b_k]\subseteq[0,\frac{(\beta\mathcal{H}^*)^k}{\lambda f_*}]$. Therefore, we have, from the Hoeffding inequality [86],

$$\begin{aligned} &P(\mathcal{C}_k|\mathcal{V}_k,\tilde{\theta}_k=\theta,\bar{\chi}_{k-1}=\chi) \\ &= P\left(\left|\frac{1}{N_k}\sum_{\mathbf{x}\in\Lambda_k}\hat{Y}^k(\mathbf{x},\bar{\chi}_{k-1}) - \hat{Y}^k(\bar{\chi}_{k-1})\right| > \zeta \,\middle|\, \mathcal{V}_k,\tilde{\theta}_k=\theta,\bar{\chi}_{k-1}=\chi\right) \\ &\le 2\exp\left(\frac{-2N_k\zeta^2}{(b_k-a_k)^2}\right) \\ &\le 2\exp\left(\frac{-2N_k\zeta^2[\lambda f_*]^2}{(\beta\mathcal{H}^*)^{2k}}\right) \quad \forall k=1,2\ldots. \end{aligned} \tag{4.33}$$

Since

$$\begin{aligned} P(\mathcal{C}_k\cap\mathcal{V}_k) &= \int_{\theta,\chi} P(\mathcal{C}_k\cap\mathcal{V}_k|\tilde{\theta}_k=\theta,\bar{\chi}_{k-1}=\chi) f_{\tilde{\theta}_k,\bar{\chi}_{k-1}}(d\theta,d\chi) \\ &\le \int_{\theta,\chi} P(\mathcal{C}_k|\mathcal{V}_k,\tilde{\theta}_k=\theta,\bar{\chi}_{k-1}=\chi) f_{\tilde{\theta},\bar{\chi}_{k-1}}(d\theta,d\chi), \end{aligned}$$

where $f_{\tilde{\theta}_k,\bar{\chi}_{k-1}}(\cdot,\cdot)$ is the joint distribution of random variables $\tilde{\theta}_k$ and $\bar{\chi}_{k-1}$, we have, by inequality (4.33),

$$\begin{aligned} P(\mathcal{C}_k\cap\mathcal{V}_k) &\le 2\exp\left(\frac{-2N_k\zeta^2[\lambda f_*]^2}{(\beta\mathcal{H}^*)^{2k}}\right), \\ &\le 2\exp\left(\frac{-2(\alpha^{k-\mathcal{T}_\varepsilon}N_0)\zeta^2[\lambda f_*]^2}{(\beta\mathcal{H}^*)^{2k}}\right) \quad \forall k\ge\mathcal{T}_\varepsilon, \\ &= 2\exp\left(\frac{-2N_0\zeta^2\lambda^2 f_*^2}{\alpha^{\mathcal{T}_\varepsilon}}\left(\frac{\alpha}{(\beta\mathcal{H}^*)^2}\right)^k\right) \quad \forall k\ge\mathcal{T}_\varepsilon. \end{aligned}$$

Since $\alpha/(\beta\mathcal{H}^*)^2 > 1$ (by assumption), it follows that

$$\lim_{k\to\infty} P(\mathcal{C}_k\cap\mathcal{V}_k) = 0.$$

Furthermore, since $e^{-\mathbf{x}} < 1/\mathbf{x}\ \forall \mathbf{x} > 0$ we have

$$P(\mathcal{C}_k \cap \mathcal{V}_k) < \frac{\alpha^{\mathcal{T}_\varepsilon}}{N_0 \zeta^2 \lambda^2 f_*^2} \left(\frac{(\beta \mathcal{H}^*)^2}{\alpha} \right)^k \quad \forall k \geq \mathcal{T}_\varepsilon,$$

and because $(\beta \mathcal{H}^*)^2/\alpha < 1$, we have

$$\sum_{k=0}^{\infty} P(\mathcal{C}_k \cap \mathcal{V}_k) < \mathcal{T}_\varepsilon + \frac{\alpha^{\mathcal{T}_\varepsilon}}{N_0 \zeta^2 \lambda^2 f_*^2} \sum_{k=\mathcal{T}_\varepsilon}^{\infty} \left(\frac{(\beta \mathcal{H}^*)^2}{\alpha} \right)^k < \infty.$$

Finally by the Borel–Cantelli lemma and Eq. (4.32),

$$P(\mathcal{C}_k \text{ i.o.}) = P(\mathcal{C}_k \cap \mathcal{V}_k \text{ i.o.}) = 0.$$

Since this holds for any $\zeta > 0$, we have $\frac{1}{N_k} \sum_{\mathbf{x} \in \Lambda_k} \hat{Y}^k(\mathbf{x}, \bar{\chi}_{k-1}) \to \hat{Y}^k(\bar{\chi}_{k-1})$ w.p.1.

By following the same argument as before, we can also show that

$$\frac{1}{N_k} \sum_{\mathbf{x} \in \Lambda_k} \hat{Y}^k(\mathbf{x}, \bar{\chi}_{k-1}) \Upsilon(\mathbf{x}) \to \hat{Y}^k_\Upsilon(\bar{\chi}_{k-1}) \quad \text{w.p.1.}$$

Since $\liminf_{k \to \infty} \hat{Y}^k(\bar{\chi}_{k-1}) > 0\ \forall \omega \in \Omega_1^c$ from (4.31), we have

$$\frac{\frac{1}{N_k} \sum_{\mathbf{x} \in \Lambda_k} \hat{Y}^k(\mathbf{x}, \bar{\chi}_{k-1}) \Upsilon(\mathbf{x})}{\frac{1}{N_k} \sum_{\mathbf{x} \in \Lambda_k} \hat{Y}^k(\mathbf{x}, \bar{\chi}_{k-1})} \to \frac{\hat{Y}^k_\Upsilon(\bar{\chi}_{k-1})}{\hat{Y}^k(\bar{\chi}_{k-1})} \quad \text{a.s. on } \Omega_1^c.$$

Thus, in conclusion, we have

$$E_{\tilde{g}_k}\big[\Upsilon(\mathbf{X})\big] \to E_{\hat{g}_k}\big[\Upsilon(\mathbf{X})\big] \quad \text{w.p.1.}$$

On the other hand, by Assumptions A1, A2, and following the proof of Theorem 4.5, it is not difficult to show that

$$E_{\hat{g}_k}\big[\Upsilon(\mathbf{X})\big] \to \Upsilon\big(\mathbf{x}^*\big) \quad \text{w.p.1.}$$

Hence by Lemma 4.8, we have

$$\lim_{k \to \infty} m(\tilde{\theta}_k) := \lim_{k \to \infty} E_{\tilde{\theta}_k}\big[\Upsilon(\mathbf{X})\big] = \lim_{k \to \infty} E_{\tilde{g}_k}\big[\Upsilon(\mathbf{X})\big] = \Upsilon\big(\mathbf{x}^*\big) \quad \text{w.p.1.} \qquad \square$$

Roughly speaking, the second result in Theorem 4.10 can be understood as finite-time ε-optimality. To see this, consider the special case where $J(\mathbf{x})$ is locally concave on the set $\mathcal{O}_\varepsilon$. Let $\mathbf{x}, \mathbf{y} \in \mathcal{O}_\varepsilon$ and $\eta \in [0, 1]$ be arbitrary. By the definition of concavity, we will have $J(\eta \mathbf{x} + (1 - \eta)\mathbf{y}) \geq \eta J(\mathbf{x}) + (1 - \eta) J(\mathbf{y}) \geq J(\mathbf{x}^*) - \varepsilon$, which implies that the set $\mathcal{O}_\varepsilon$ is convex. If in addition $\Upsilon(\mathbf{x})$ is also convex and one-to-one on $\mathcal{O}_\varepsilon$ (e.g. multivariate normal p.d.f.), then $\text{CONV}\{\Upsilon(\mathcal{O}_\varepsilon)\} = \Upsilon(\mathcal{O}_\varepsilon)$, and it follows that $\Upsilon^{-1}(m(\tilde{\theta}_{k+1})) \in \mathcal{O}_\varepsilon$ for all $k \geq \mathcal{K}$ w.p.1.

The following results are now immediate.

Corollary 4.11 (Multivariate Normal) *For continuous optimization problems in* $\Re^n$, *if multivariate normal p.d.f.s are used in* MRAS_1, *i.e.*,

$$f(\mathbf{x}, \tilde{\theta}_k) = \frac{1}{\sqrt{(2\pi)^n |\tilde{\Sigma}_k|}} \exp\left(-\frac{1}{2}(\mathbf{x} - \tilde{\mu}_k)^T \tilde{\Sigma}_k^{-1}(\mathbf{x} - \tilde{\mu}_k)\right),$$

$\varepsilon > 0$, $\alpha > (\beta \mathcal{H}^*)^2$, *and Assumptions* A1, A2, A3′, *and* A4′ *are satisfied, then*

$$\lim_{k\to\infty} \tilde{\mu}_k = \mathbf{x}^*, \quad \textit{and} \quad \lim_{k\to\infty} \tilde{\Sigma}_k = 0_{n\times n} \quad w.p.1.$$

Corollary 4.12 (Independent Univariate) *If the components of the random vector* $\mathbf{X} = (\mathsf{X}_1, \ldots, \mathsf{X}_n)$ *are independent, each has a univariate density/mass of the form*

$$f(\mathsf{x}_i, \vartheta_i) = \exp\big(\mathsf{x}_i \vartheta_i - K(\vartheta_i)\big) h(\mathsf{x}_i), \quad \mathsf{x}_i \in \Re,\ \vartheta_i \in \Re,\ \forall i = 1, \ldots, n,$$

$\varepsilon > 0$, $\alpha > (\beta \mathcal{H}^*)^2$, *and Assumptions* A1, A2, A3′, *and* A4′ *are satisfied, then*

$$\lim_{k\to\infty} m(\tilde{\theta}_k) := \lim_{k\to\infty} E_{\tilde{\theta}_k}[\mathbf{X}] = \mathbf{x}^* \quad w.p.1, \ \textit{where}\ \tilde{\theta}_k := \big(\vartheta_1^k, \ldots, \vartheta_n^k\big).$$

4.2.3 MRAS_2 *Convergence*

We first make the following assumptions on the sample performances $\mathcal{J}_i(\mathbf{x})$.

Assumption L1 For any given $\epsilon > 0$, there exists a positive number n^* such that for all $n \geq n^*$,

$$\sup_{\mathbf{x}\in\mathcal{X}} P\left(\left|\frac{1}{n}\sum_{i=1}^{n} \mathcal{J}_i(\mathbf{x}) - J(\mathbf{x})\right| \geq \epsilon\right) \leq \phi(n, \epsilon),$$

where $\phi(\cdot, \cdot)$ is strictly decreasing in its first argument and non-increasing in its second argument. Moreover, $\phi(n, \epsilon) \to 0$ as $n \to \infty$.

Assumption L2 For any $\epsilon > 0$, there exist positive numbers m^* and n^* such that for all $m \geq m^*$ and $n \geq n^*$,

$$\sup_{\mathbf{x},\mathbf{y}\in\mathcal{X}} P\left(\left|\frac{1}{m}\sum_{i=1}^{m} \mathcal{J}_i(\mathbf{x}) - \frac{1}{n}\sum_{i=1}^{n} \mathcal{J}_i(\mathbf{y}) - J(\mathbf{x}) + J(\mathbf{y})\right| \geq \epsilon\right) \leq \phi\big(\min\{m, n\}, \epsilon\big),$$

where ϕ satisfies the conditions in L1.

Assumption L1 is satisfied by many random sequences, e.g., the sequence of i.i.d. random variables with (asymptotically) uniformly bounded variance, or a class of

random variables (not necessarily i.i.d.) that satisfy the large deviations principle (cf. [88, 189]). Assumption L2 can be viewed as a simple extension of Assumption L1. Most random sequences that satisfy Assumption L1 will also satisfy Assumption L2. For example, consider the particular case where $\mathcal{J}_i(\mathbf{x}),\ i = 1, 2, \ldots$ are i.i.d. with uniformly bounded variance $\sigma^2(\mathbf{x})$ and $E(\mathcal{J}_i(\mathbf{x})) = J(\mathbf{x}),\ \forall \mathbf{x} \in \mathcal{X}$. Thus the variance of the random variable $\frac{1}{m}\sum_{i=1}^{m}\mathcal{J}_i(\mathbf{x}) - \frac{1}{n}\sum_{i=1}^{n}\mathcal{J}_i(\mathbf{y})$ is $\frac{1}{m}\sigma^2(\mathbf{x}) + \frac{1}{n}\sigma^2(\mathbf{y})$, which is also uniformly bounded on $\mathcal{X}$. By Chebyshev's inequality, we have, for any $\mathbf{x}, \mathbf{y} \in \mathcal{X}$,

$$
\begin{aligned}
&P\left(\left|\frac{1}{m}\sum_{i=1}^{m}\mathcal{J}_i(\mathbf{x}) - \frac{1}{n}\sum_{i=1}^{n}\mathcal{J}_i(\mathbf{y}) - J(\mathbf{x}) + J(\mathbf{y})\right| \geq \epsilon\right) \\
&\leq \frac{\sup_{\mathbf{x},\mathbf{y}}[\frac{1}{m}\sigma^2(\mathbf{x}) + \frac{1}{n}\sigma^2(\mathbf{y})]}{\epsilon^2} \\
&\leq \frac{\sup_{\mathbf{x},\mathbf{y}}[\sigma^2(\mathbf{x}) + \sigma^2(\mathbf{y})]}{\min\{m,n\}\epsilon^2} = \phi\big(\min\{m,n\},\epsilon\big).
\end{aligned}
$$

We impose the following regularity condition on the observation allocation rule.

Assumption L3 The observation allocation rule $\{M_k, k = 0, 1, \ldots\}$ satisfies $M_k \geq M_{k-1}\ \forall k = 1, 2, \ldots$, and $M_k \to \infty$ as $k \to \infty$. Moreover, for any $\epsilon > 0$, there exist $\delta_\epsilon \in (0, 1)$ and $\mathcal{K}_\epsilon > 0$ such that $\alpha^{2k}\phi(M_{k-1}, \epsilon) \leq (\delta_\epsilon)^k,\ \forall k \geq \mathcal{K}_\epsilon$, where ϕ is defined as in Assumption L1.

Assumption L3 is a mild condition and is very easy to verify. For instance, if $\phi(n, \epsilon)$ takes the form $\phi(n, \epsilon) = \frac{\mathcal{C}(\epsilon)}{n}$, where $\mathcal{C}(\epsilon)$ is a constant depending on ϵ, then the condition on M_{k-1} becomes $M_{k-1} \geq \mathcal{C}(\epsilon)(\frac{\alpha^2}{\delta_\epsilon})^k\ \forall k \geq \mathcal{K}_\epsilon$. As another example, if $\mathcal{J}_i(\mathbf{x}), i = 1, 2 \ldots$ satisfies the large deviations principle and $\phi(n, \epsilon) = e^{-n\mathcal{C}(\epsilon)}$, then the condition becomes $M_{k-1} \geq [\ln(\frac{\alpha^2}{\delta_\epsilon})/\mathcal{C}(\epsilon)]k,\ \forall k \geq \mathcal{K}_\epsilon$.

The following lemma implies the probability one convergence of the sequence of stochastic thresholds $\{\bar{\chi}_k\}$ generated by MRAS$_2$.

Lemma 4.13 *If Assumptions L1–L3 are satisfied, then the sequence of random variables* $\{\mathbf{X}_k^*,\ k = 0, 1, \ldots\}$ *generated by* MRAS$_2$ *converges w.p.1 as* $k \to \infty$.

Proof Let $\mathcal{A}_k$ be the event that Steps 3a or 3b is visited at the kth iteration of the algorithm, $\mathcal{B}_k := \{J(\mathbf{X}_k^*) - J(\mathbf{X}_{k-1}^*) \leq \frac{\varepsilon}{2}\}$. Since each time Step 3a or 3b is visited, we have $\bar{\mathcal{J}}_k(\mathbf{X}_k^*) - \bar{\mathcal{J}}_{k-1}(\mathbf{X}_{k-1}^*) \geq \varepsilon$, we have

$$
\begin{aligned}
&P(\mathcal{A}_k \cap \mathcal{B}_k) \\
&\leq P\left(\left\{\bar{\mathcal{J}}_k(\mathbf{X}_k^*) - \bar{\mathcal{J}}_{k-1}(\mathbf{X}_{k-1}^*) \geq \varepsilon\right\} \cap \left\{J(\mathbf{X}_k^*) - J(\mathbf{X}_{k-1}^*) \leq \frac{\varepsilon}{2}\right\}\right)
\end{aligned}
$$

$$
\begin{aligned}
&\le P\Bigg(\bigcup_{\mathbf{x}\in\Lambda_k,\mathbf{y}\in\Lambda_{k-1}}\left\{\left\{\bar{\mathcal{J}}_k(\mathbf{x})-\bar{\mathcal{J}}_{k-1}(\mathbf{y})\ge\varepsilon\right\}\cap\left\{J(\mathbf{x})-J(\mathbf{y})\le\frac{\varepsilon}{2}\right\}\right\}\Bigg)\\
&\le \sum_{\mathbf{x}\in\Lambda_k,\mathbf{y}\in\Lambda_{k-1}} P\Bigg(\left\{\bar{\mathcal{J}}_k(\mathbf{x})-\bar{\mathcal{J}}_{k-1}(\mathbf{y})\ge\varepsilon\right\}\cap\left\{J(\mathbf{x})-J(\mathbf{y})\le\frac{\varepsilon}{2}\right\}\Bigg)\\
&\le |\Lambda_k||\Lambda_{k-1}|\sup_{\mathbf{x},\mathbf{y}\in\mathcal{X}} P\Bigg(\left\{\bar{\mathcal{J}}_k(\mathbf{x})-\bar{\mathcal{J}}_{k-1}(\mathbf{y})\ge\varepsilon\right\}\cap\left\{J(\mathbf{x})-J(\mathbf{y})\le\frac{\varepsilon}{2}\right\}\Bigg)\\
&\le |\Lambda_k||\Lambda_{k-1}|\sup_{\mathbf{x},\mathbf{y}\in\mathcal{X}} P\left(\bar{\mathcal{J}}_k(\mathbf{x})-\bar{\mathcal{J}}_{k-1}(\mathbf{y})-J(\mathbf{x})+J(\mathbf{y})\ge\frac{\varepsilon}{2}\right)\\
&\le |\Lambda_k||\Lambda_{k-1}|\phi\left(\min\{M_k,M_{k-1}\},\frac{\varepsilon}{2}\right)\quad\text{by Assumption L2}\\
&\le \alpha^{2k}N_0^2\phi\left(M_{k-1},\frac{\varepsilon}{2}\right)\le N_0^2(\delta_{\varepsilon/2})^k\quad\forall k\ge\mathcal{K}_{\varepsilon/2}\text{ by Assumption L3.}
\end{aligned}
$$

Therefore,

$$
\sum_{k=1}^{\infty}P(\mathcal{A}_k\cap\mathcal{B}_k)\le\mathcal{K}_{\varepsilon/2}+N_0^2\sum_{k=\mathcal{K}_{\varepsilon/2}}^{\infty}(\delta_{\varepsilon/2})^k\le\infty.
$$

By the Borel–Cantelli lemma, we have

$$
P(\mathcal{A}_k\cap\mathcal{B}_k\text{ i.o.})=0.
$$

It follows that if $\mathcal{A}_k$ happens infinitely often, then w.p.1, $\mathcal{B}_k^c$ will also happen infinitely often. Thus,

$$
\begin{aligned}
&\sum_{k=1}^{\infty}\left[J\left(\mathbf{X}_k^*\right)-J\left(\mathbf{X}_{k-1}^*\right)\right]\\
&\quad=\sum_{k:\ \mathcal{A}_k\text{ occurs}}\left[J\left(\mathbf{X}_k^*\right)-J\left(\mathbf{X}_{k-1}^*\right)\right]+\sum_{k:\ \mathcal{A}_k^c\text{ occurs}}\left[J\left(\mathbf{X}_k^*\right)-J\left(\mathbf{X}_{k-1}^*\right)\right]\\
&\quad=\sum_{k:\ \mathcal{A}_k\text{ occurs}}\left[J\left(\mathbf{X}_k^*\right)-J\left(\mathbf{X}_{k-1}^*\right)\right]\quad\text{since }\mathbf{X}_k^*=\mathbf{X}_{k-1}^*\text{ if Step 3c is visited}\\
&\quad=\sum_{k:\ \mathcal{A}_k\cap\mathcal{B}_k\text{ occurs}}\left[J\left(\mathbf{X}_k^*\right)-J\left(\mathbf{X}_{k-1}^*\right)\right]+\sum_{k:\ \mathcal{A}_k\cap\mathcal{B}_k^c\text{ occurs}}\left[J\left(\mathbf{X}_k^*\right)-J\left(\mathbf{X}_{k-1}^*\right)\right]\\
&\quad=\infty\quad\text{w.p.1, since }\varepsilon>0.
\end{aligned}
$$

However, this is a contradiction, since $J(\mathbf{x})$ is bounded from above by $J(\mathbf{x}^*)$. Therefore, w.p.1, $\mathcal{A}_k$ can only happen a finite number of times, which implies that the sequence $\{\mathbf{X}_k^*, k=0,1,\ldots\}$ converges w.p.1. □

Note that when the solution space $\mathcal{X}$ is finite, the set Λ_k will be finite for all k. Thus, Lemma 4.13 may still hold if we replace Assumption L3 by some milder conditions on M_k. One such condition is $\sum_{k=1}^{\infty} \phi(M_k, \varepsilon) < \infty$, for example, when the sequence $\mathcal{J}_i(\mathbf{x}), i = 1, 2 \ldots$, satisfies the large deviations principle and $\phi(n, \varepsilon)$ takes the form $\phi(n, \varepsilon) = e^{-n\mathcal{C}(\varepsilon)}$. A particular observation allocation rule that satisfies this condition is $M_k = M_{k-1} + 1 \; \forall k = 1, 2, \ldots$.

Define $\chi_k = J(\mathbf{X}_k^*)$, i.e., the true performance of the random sample $\mathbf{X}_k^*$. Lemma 4.13 implies that the sequence $\{\chi_k\}$ converges. It is easy to see that the sequence of stochastic thresholds $\{\bar{\chi}_k\}$ is just a sample average approximation of the sequence $\{\chi_k\}$. As we will see, by using a slightly stronger condition than Assumption L3, we can show that $\bar{\chi}_k$ not only converges to χ_k, but also does so at an exponential rate.

To establish the global convergence of MRAS$_2$, we make the following additional assumptions.

Assumption B1 There exists a compact set Ξ such that for the sequence of random variables $\{\mathbf{X}_k^*, k = 0, 1, \ldots\}$ generated by MRAS$_2$, $\exists \mathcal{N} < \infty$ w.p.1 such that $\{\mathbf{x} : J(\mathbf{x}) \geq J(\mathbf{X}_k^*) - \varepsilon\} \cap \mathcal{X} \subseteq \Xi \; \forall k \geq \mathcal{N}$.

Assumption B2 For any constant $\xi < J(\mathbf{x}^*)$, the set $\{\mathbf{x} : J(\mathbf{x}) \geq \xi\} \cap \mathcal{X}$ has a strictly positive Lebesgue or discrete measure.

Assumption B3 For any given constant $\delta > 0$, $\sup_{\mathbf{x} \in A_\delta} J(\mathbf{x}) < J(\mathbf{x}^*)$, where $A_\delta := \{\mathbf{x} : \|\mathbf{x} - \mathbf{x}^*\| > \delta\} \cap \mathcal{X}$, and we define the supremum over the empty set to be $-\infty$.

Assumption B4 For each point $z \leq J(\mathbf{x}^*)$, there exist $\Delta_k > 0$ and $L_k > 0$, such that $\frac{|(\mathcal{H}(z))^k - (\mathcal{H}(\bar{z}))^k|}{|(\mathcal{H}(z))^k|} \leq L_k |z - \bar{z}|$ for all $\bar{z} \in (z - \Delta_k, z + \Delta_k)$.

Assumption B5 The maximizer of (4.12) is an interior point of $\Theta \; \forall k$.

Assumption B6 $f(\mathbf{x}, \theta_0) > 0 \; \forall \mathbf{x} \in \mathcal{X}$ and $f_* := \inf_{\mathbf{x} \in \Xi} f(\mathbf{x}, \theta_0) > 0$, where Ξ is defined in Assumption B1.

Since the sequence $\{\mathbf{X}_k^*\}$ generated by MRAS$_2$ converges (see Lemma 4.13), Assumption B1 requires that the search of MRAS$_2$ will eventually end up in a compact set. The assumption is trivially satisfied if the solution space $\mathcal{X}$ is compact. Assumption B2 ensures that the neighborhood of the optimal solution $\mathbf{x}^*$ will be sampled with a strictly positive probability. Since $\mathbf{x}^*$ is the unique global optimizer of J, Assumption B3 is satisfied by many functions encountered in practice. Assumption B4 can be understood as a locally Lipschitz condition on $[\mathcal{H}(\cdot)]^k$; its suitability will be discussed later. In actual implementation of the algorithm, Step 4 is often posed as an unconstrained optimization problem, i.e., $\Theta = \Re^m$, in which case Assumption B5 is automatically satisfied. It is also easy to verify that Assumption B6 is satisfied by most NEFs.

The next lemma relates the sequence of sampling distributions $\{f(\cdot, \tilde{\theta}_k)\}$ to the sequence of reference models $\{\tilde{g}_k, k = 1, 2, \ldots\}$ defined by (4.13).

Lemma 4.14 *If Assumption B5 holds, then we have*

$$m(\tilde{\theta}_{k+1}) = E_{\tilde{\theta}_{k+1}}\big[\Upsilon(\mathbf{X})\big] = E_{\tilde{g}_{k+1}}\big[\Upsilon(\mathbf{X})\big], \quad \forall k = 0, 1, \ldots,$$

where $E_{\tilde{\theta}_{k+1}}$ and $E_{\tilde{g}_{k+1}}$ are the expectations taken with respect to the p.d.f./p.m.f. $f(\cdot, \tilde{\theta}_{k+1})$ and $\tilde{g}_{k+1}$, respectively.

Proof Similar to the proof of Lemma 4.4. □

We now construct a sequence of (idealized) distributions $\{\hat{g}_k\}$ as

$$\hat{g}_{k+1}(\mathbf{x}) = \frac{[\mathcal{H}(J(\mathbf{x}))]^k \tilde{I}(J(\mathbf{x}), \chi_{k-1})}{\int_{\mathbf{x}\in\mathcal{X}} [\mathcal{H}(J(\mathbf{x}))]^k \tilde{I}(J(\mathbf{x}), \chi_{k-1}) \nu(d\mathbf{x})} \quad \forall k = 1, 2, \ldots, \tag{4.34}$$

where $\chi_{k-1} := J(\mathbf{X}^*_{k-1})$.

The outline of the convergence proof is as follows. First we establish the convergence of the sequence of idealized distributions $\{\hat{g}_k\}$. Then we show that the reference models $\{\tilde{g}_k\}$ are in fact the (sample average) approximations of the sequence $\{\hat{g}_k\}$ by proving $E_{\tilde{g}_k}[\Upsilon(\mathbf{X})] \to E_{\hat{g}_k}[\Upsilon(\mathbf{X})]$ w.p.1 as $k \to \infty$. Thus, the convergence of the sequence $\{f(\cdot, \tilde{\theta}_k)\}$ follows immediately by applying Lemma 4.14.

The convergence of the sequence $\{\hat{g}_k\}$ is formalized in the following lemma.

Lemma 4.15 *If Assumptions L1–L3, B1–B3 are satisfied, then*

$$\lim_{k\to\infty} E_{\hat{g}_k}\big[\Upsilon(\mathbf{X})\big] = \Upsilon\big(\mathbf{x}^*\big) \quad w.p.1.$$

Proof Let Ω_2 be the set of all sample paths such that Step 3a or 3b of MRAS$_2$ is visited finitely often, and let Ω_3 be the set of sample paths such that $\lim_{k\to\infty}\{J(\mathbf{x}) \geq \chi_k - \varepsilon\} \subseteq \Xi$. By Lemma 4.13, we have $P(\Omega_2) = 1$, and for each $\omega \in \Omega_2$, there exists a finite $\mathcal{N}(\omega) > 0$ such that

$$\mathbf{X}^*_k(\omega) = \mathbf{X}^*_{k-1}(\omega) \quad \forall k \geq \mathcal{N}(\omega),$$

which implies that $\chi_k(\omega) = \chi_{k-1}(\omega)\ \forall k \geq \mathcal{N}(\omega)$. Furthermore, by Assumption B1, we have $P(\Omega_3) = 1$ and $\{J(\mathbf{x}) \geq \chi_{k-1}(\omega) - \varepsilon\} \subseteq \Pi,\ \forall k \geq \mathcal{N}(\omega)\ \forall \omega \in \Omega_2 \cap \Omega_3$. By following the same argument as in the proof of Theorem 4.5, it is easy to show that

$$\lim_{k\to\infty} E_{g_k(\omega)}\big[\Upsilon(\mathbf{X})\big] = \Upsilon\big(\mathbf{x}^*\big), \quad \forall \omega \in \Omega_2 \cap \Omega_3.$$

Since $P(\Omega_2 \cap \Omega_3) = 1$, the proof is thus completed. □

As mentioned earlier, the rest of the convergence proof now amounts to showing that $E_{\tilde{g}_k}[\Upsilon(\mathbf{X})] \to E_{\hat{g}_k}[\Upsilon(\mathbf{X})]$ w.p.1 as $k \to \infty$. However, there is one more complication: Since $\mathcal{H}$ is an increasing function and is raised to the kth power in both $\tilde{g}_{k+1}$ and $\hat{g}_{k+1}$ (see Eqs. (4.13) and (4.34)), the associated estimation error between $\bar{\mathcal{J}}_k(\mathbf{x})$ and $J(\mathbf{x})$ is exaggerated. Thus, even though we have $\lim_{k\to\infty} \bar{\mathcal{J}}_k(\mathbf{x}) = J(\mathbf{x})$ w.p.1, the quantities $[\mathcal{H}(\bar{\mathcal{J}}_k(\mathbf{x}))]^k$ and $[\mathcal{H}(J(\mathbf{x}))]^k$ may still differ considerably as k gets large. Therefore, the sequence $\{\bar{\mathcal{J}}_k(\mathbf{x})\}$ not only has to converge to $J(\mathbf{x})$, but it should also do so at a fast enough rate in order to keep the resultant approximation error between $[\mathcal{H}(\bar{\mathcal{J}}_k(\mathbf{x}))]^k$ and $[\mathcal{H}(J(\mathbf{x}))]^k$ at a manageable level. This requirement is summarized in the following assumption.

Assumption L4 For any given $\zeta > 0$, there exist $\delta^* \in (0, 1)$ and $\mathcal{K} > 0$ such that the observation allocation rule $\{M_k, k = 0, 1, \ldots\}$ satisfies

$$\alpha^k \phi\left(M_k, \min\left\{\Delta_k, \frac{\zeta}{\alpha^{k/2}}, \frac{\zeta}{\alpha^{k/2} L_k}\right\}\right) \leq \left(\delta^*\right)^k \quad \forall k \geq \mathcal{K},$$

where ϕ is defined as in Assumption L1, and Δ_k and L_k are defined as in Assumption B4.

Let $\mathcal{H}(z) = e^{\tau z}$, for some positive constant τ. We have $\mathcal{H}^k(z) = e^{\tau k z}$ and $[\mathcal{H}^k(z)]' = k\tau e^{\tau k z}$. It is easy to verify that $\frac{|\mathcal{H}^k(z) - \mathcal{H}^k(\bar{z})|}{\mathcal{H}^k(z)} \leq k\tau e^{\tau k \Delta_k} |z - \bar{z}|$ $\forall \bar{z} \in (z - \Delta_k, z + \Delta_k)$, and Assumption B4 is satisfied for $\Delta_k = 1/k$ and $L_k = \tau e^{\tau} k$. Thus, the condition in Assumption L4 becomes $\alpha^k \phi(M_k, \bar{\zeta}/\alpha^{k/2} k) \leq (\delta^*)^k$ $\forall k \geq \mathcal{K}$, where $\bar{\zeta} = \zeta/\tau e^{\tau}$. We consider the following two special cases of Assumption L4. Let $\mathcal{J}_i(\mathbf{x})$ be i.i.d. with $E(\mathcal{J}_i(\mathbf{x})) = J(\mathbf{x})$ and uniformly bounded variance $\sup_{\mathbf{x}\in\mathcal{X}} \sigma^2(\mathbf{x}) \leq \sigma^2$. By Chebyshev's inequality

$$P\left(\left|\bar{\mathcal{J}}_k(\mathbf{x}) - J(\mathbf{x})\right| \geq \frac{\bar{\zeta}}{\alpha^{k/2} k}\right) \leq \frac{\sigma^2 \alpha^k k^2}{M_k \bar{\zeta}^2}.$$

Thus, it is easy to check that Assumption L4 is satisfied by $M_k = (\mu\alpha^2)^k$ for any constant $\mu > 1$.

As a second example, consider the case where $\mathcal{J}_1(\mathbf{x}), \ldots, \mathcal{J}_{N_k}(\mathbf{x})$ are i.i.d. with $E(\mathcal{J}_i[\mathbf{x}]) = J(\mathbf{x})$ and bounded support $[a, b]$. By the Hoeffding inequality ([86])

$$P\left(\left|\bar{\mathcal{J}}_k(\mathbf{x}) - J(\mathbf{x})\right| \geq \frac{\bar{\zeta}}{\alpha^{k/2} k}\right) \leq 2\exp\left(\frac{-2M_k \bar{\zeta}^2}{(b-a)^2 \alpha^k k^2}\right).$$

In this case, Assumption L4 is satisfied by $M_k = (\mu\alpha)^k$ for any constant $\mu > 1$.

Again, as discussed earlier following Lemma 4.13, Assumption L4 can be replaced by the weaker condition

$$\sum_{k=1}^{\infty} \phi\left(M_k, \min\left\{\Delta_k, \frac{\zeta}{\alpha^{k/2}}, \frac{\zeta}{\alpha^{k/2} L_k}\right\}\right) < \infty$$

when the solution space $\mathcal{X}$ is finite.

The following result shows that under Assumption L4, the stochastic threshold $\bar{\chi}_k$ converges to χ_k exponentially fast.

Proposition 4.16 *If Assumptions L1–L4 are satisfied, then*

$$\lim_{k\to\infty} \alpha^{k/2}|\bar{\chi}_k - \chi_k| = 0 \quad w.p.1.$$

Proof Again, we consider the sequence $\{\mathbf{X}_k^*\}$ generated by MRAS$_2$.

We have for any $\zeta > 0$

$$\begin{aligned}
P\left(|\bar{\chi}_k - \chi_k| \geq \frac{\zeta}{\alpha^{k/2}}\right) &= P\left(\left|\bar{\mathcal{J}}_k\left(\mathbf{X}_k^*\right) - J\left(\mathbf{X}_k^*\right)\right| \geq \frac{\zeta}{\alpha^{k/2}}\right) \\
&\leq P\left(\bigcup_{\mathbf{x}\in\Lambda_k}\left\{\left|\bar{\mathcal{J}}_k(\mathbf{x}) - J(\mathbf{x})\right| \geq \frac{\zeta}{\alpha^{k/2}}\right\}\right) \\
&\leq \sum_{\mathbf{x}\in\Lambda_k} P\left(\left|\bar{\mathcal{J}}_k(\mathbf{x}) - J(\mathbf{x})\right| \geq \frac{\zeta}{\alpha^{k/2}}\right) \\
&\leq |\Lambda_k| \sup_{\mathbf{x}\in\mathcal{X}} P\left(\left|\bar{\mathcal{J}}_k(\mathbf{x}) - J(\mathbf{x})\right| \geq \frac{\zeta}{\alpha^{k/2}}\right) \\
&\leq \alpha^k N_0 \phi\left(M_k, \zeta/\alpha^{k/2}\right) \quad \text{by Assumption L1} \\
&\leq N_0\left(\delta^*\right)^k \quad \forall k \geq \mathcal{K} \text{ by Assumption L4.}
\end{aligned}$$

Thus

$$\sum_{k=1}^{\infty} P\left(|\bar{\chi}_k - \chi_k| \geq \frac{\zeta}{\alpha^{k/2}}\right) \leq \mathcal{K} + N_0 \sum_{k=\mathcal{K}}^{\infty}\left(\delta^*\right)^k < \infty.$$

And by Borel–Cantelli lemma,

$$P\left(\left\{|\bar{\chi}_k - \chi_k| \geq \frac{\zeta}{\alpha^{k/2}}\right\} \text{ i.o.}\right) = 0.$$

Since ζ is arbitrary, the proof is thus completed. □

We now state the main theorem.

Theorem 4.17 *Let $\varphi > 0$ be a positive constant satisfying the condition that the set $\{\mathbf{x} : \mathcal{H}(J(\mathbf{x})) \geq \frac{1}{\varphi}\}$ has a strictly positive Lebesgue/counting measure. If Assumptions L1–L4, B1–B6 are satisfied, and $\alpha > (\varphi\mathcal{H}^*)^2$, then*

$$\lim_{k\to\infty} m(\tilde{\theta}_k) = \lim_{k\to\infty} E_{\tilde{\theta}_k}\left[\Upsilon(\mathbf{X})\right] = \Upsilon\left(\mathbf{x}^*\right) \quad w.p.1, \tag{4.35}$$

where the limit above is component-wise.

By the monotonicity of $\mathcal{H}$ and Assumption B2, it is easy to see that such a positive constant φ in Theorem 4.17 always exists. Moreover, for continuous problems, φ can be chosen such that $\varphi\mathcal{H}^* \approx 1$; for discrete problems, if the counting measure is used, then we can choose $\varphi = 1/\mathcal{H}^*$.

Proof For brevity, we define the function

$$Y_k(Z, \chi) := \tilde{\mathcal{H}}_k(Z)\tilde{I}(Z, \chi),$$

where

$$\tilde{\mathcal{H}}_k(Z) = \begin{cases} [\mathcal{H}(J(\mathbf{x}))]^k / \tilde{f}(\mathbf{x}, \tilde{\theta}_k) & \text{if } Z = J(\mathbf{x}), \\ [\mathcal{H}(\bar{\mathcal{J}}_k(\mathbf{x}))]^k / \tilde{f}(\mathbf{x}, \tilde{\theta}_k) & \text{if } Z = \bar{\mathcal{J}}_k(\mathbf{x}). \end{cases}$$

By Assumption B6, the support of $\tilde{f}(\cdot, \tilde{\theta}_k)$ satisfies $\mathcal{X} \subseteq \text{supp}\{\tilde{f}(\cdot, \tilde{\theta}_k)\}\ \forall k$. Thus, we can write

$$E_{\hat{g}_{k+1}}\big[\Upsilon(\mathbf{X})\big] = \frac{\tilde{E}_{\tilde{\theta}_k}[Y_k(J(\mathbf{X}), \chi_{k-1})\Upsilon(\mathbf{X})]}{\tilde{E}_{\tilde{\theta}_k}[Y_k(J(\mathbf{X}), \chi_{k-1})]},$$

where $\tilde{E}_{\tilde{\theta}_k}$ is the expectation taken with respect to $\tilde{f}(\cdot, \tilde{\theta}_k)$. We now show $E_{\tilde{g}_{k+1}}[\Upsilon(\mathbf{X})] \to E_{\hat{g}_{k+1}}[\Upsilon(\mathbf{X})]$ w.p.1 as $k \to \infty$. Since we are only interested in the limiting behavior of $E_{\tilde{g}_{k+1}}[\Upsilon(\mathbf{X})]$, from the definition of $\tilde{g}_{k+1}$ (see Eq. (4.24)), it is sufficient to show that

$$\frac{\sum_{\mathbf{x}\in\Lambda_k} Y_k(\bar{\mathcal{J}}_k(\mathbf{x}), \bar{\chi}_k)\Upsilon(\mathbf{x})}{\sum_{\mathbf{x}\in\Lambda_k} Y_k(\bar{\mathcal{J}}_k(\mathbf{x}), \bar{\chi}_k)} \to E_{\hat{g}_{k+1}}\big[\Upsilon(\mathbf{X})\big] \quad \text{w.p.1},$$

where, and also hereafter, whenever $\{\mathbf{x} \in \Lambda_k : \bar{\mathcal{J}}_k(\mathbf{x}) > \bar{\chi}_k - \varepsilon\} = \emptyset$, we define $0/0 = 0$. We have

$$\begin{aligned}
&\frac{\sum_{\mathbf{x}\in\Lambda_k} Y_k(\bar{\mathcal{J}}_k(\mathbf{x}), \bar{\chi}_k)\Upsilon(\mathbf{x})}{\sum_{\mathbf{x}\in\Lambda_k} (\bar{\mathcal{J}}_k(\mathbf{x}), \bar{\chi}_k)} - E_{\hat{g}_{k+1}}\big[\Upsilon(\mathbf{X})\big] \\
&= \frac{\sum_{\mathbf{x}\in\Lambda_k} Y_k(\bar{\mathcal{J}}_k(\mathbf{x}), \bar{\chi}_k)\Upsilon(\mathbf{x})}{\sum_{\mathbf{x}\in\Lambda_k} Y_k(\bar{\mathcal{J}}_k(\mathbf{x}), \bar{\chi}_k)} - \frac{\tilde{E}_{\tilde{\theta}_k}[Y_k(J(\mathbf{X}), \chi_{k-1})\Upsilon(\mathbf{X})]}{\tilde{E}_{\tilde{\theta}_k}[Y_k(J(\mathbf{X}), \chi_{k-1})]} \\
&= \left\{ \frac{\frac{1}{N_k}\sum_{\mathbf{x}\in\Lambda_k} Y_k(J(\mathbf{x}), \chi_k)\Upsilon(\mathbf{x})}{\frac{1}{N_k}\sum_{\mathbf{x}\in\Lambda_k} Y_k(J(\mathbf{x}), \chi_k)} - \frac{\tilde{E}_{\tilde{\theta}_k}[Y_k(J(\mathbf{X}), \chi_{k-1})\Upsilon(\mathbf{X})]}{\tilde{E}_{\tilde{\theta}_k}[Y_k(J(\mathbf{X}), \chi_{k-1})]} \right\} \quad \text{[i]} \\
&\quad + \left\{ \frac{\frac{1}{N_k}\sum_{\mathbf{x}\in\Lambda_k} Y_k(\bar{\mathcal{J}}_k(\mathbf{x}), \bar{\chi}_k)\Upsilon(\mathbf{x})}{\frac{1}{N_k}\sum_{\mathbf{x}\in\Lambda_k} Y_k(\bar{\mathcal{J}}_k(\mathbf{x}), \bar{\chi}_k)} - \frac{\frac{1}{N_k}\sum_{\mathbf{x}\in\Lambda_k} Y_k(J(\mathbf{x}), \chi_k)\Upsilon(\mathbf{x})}{\frac{1}{N_k}\sum_{\mathbf{x}\in\Lambda_k} Y_k(J(\mathbf{x}), \chi_k)} \right\}. \quad \text{[ii]}
\end{aligned}$$

We now analyze the terms [i] and [ii].

First we show that [i] $\to 0$ w.p.1 as $k \to \infty$.

$$[\mathrm{i}] = \frac{\frac{1}{N_k}\sum_{\mathbf{x}\in\Lambda_k} \varphi^k \tilde{\mathcal{H}}_k(J(\mathbf{x}))\tilde{I}(J(\mathbf{x}), \chi_k)\Upsilon(\mathbf{x})}{\frac{1}{N_k}\sum_{\mathbf{x}\in\Lambda_k} \varphi^k \tilde{\mathcal{H}}_k(J(\mathbf{x}))\tilde{I}(J(\mathbf{x}), \chi_k)} - \frac{\tilde{E}_{\tilde{\theta}_k}[\varphi^k \tilde{\mathcal{H}}_k(J(\mathbf{X}))\tilde{I}(J(\mathbf{X}), \chi_{k-1})\Upsilon(\mathbf{X})]}{\tilde{E}_{\tilde{\theta}_k}[\varphi^k \tilde{\mathcal{H}}_k(J(\mathbf{X}))\tilde{I}(J(\mathbf{X}), \chi_{k-1})]}.$$

Since $\varepsilon > 0$, we have $\chi_k - \varepsilon \le J(\mathbf{x}^*) - \varepsilon$ for all k. Thus by Assumption B2, the set $\{\mathbf{x} : J(\mathbf{x}) \ge \chi_{k-1} - \varepsilon\} \cap \mathcal{X}$ has a strictly positive Lebesgue/discrete measure for all k. It follows from Fatou's lemma that

$$\begin{aligned} &\liminf_{k\to\infty} \tilde{E}_{\tilde{\theta}_k}\big[\varphi^k \tilde{\mathcal{H}}_k\big(J(\mathbf{X})\big)\tilde{I}\big(J(\mathbf{X}), \chi_{k-1}\big)\big] \\ &\quad \ge \int_{\mathcal{X}} \liminf_{k\to\infty} \big[\varphi \mathcal{H}\big(J(\mathbf{x})\big)\big]^k \tilde{I}\big(J(\mathbf{x}), \chi_{k-1}\big)\nu(d\mathbf{x}) > 0, \end{aligned} \tag{4.36}$$

where the last inequality follows from the fact that $\varphi\mathcal{H}(J(\mathbf{x})) \ge 1\ \forall \mathbf{x} \in \{\mathbf{x} : J(\mathbf{x}) \ge \max\{\mathcal{H}^{-1}(\frac{1}{\varphi}), J(\mathbf{x}^*) - \varepsilon\}\}$.

Note that

$$\begin{aligned} &\frac{1}{N_k}\sum_{\mathbf{x}\in\Lambda_k} \varphi^k \tilde{\mathcal{H}}_k\big(J(\mathbf{x})\big)\tilde{I}\big(J(\mathbf{x}), \chi_k\big) - \tilde{E}_{\tilde{\theta}_k}\big[\varphi^k \tilde{\mathcal{H}}_k\big(J(\mathbf{X})\big)\tilde{I}\big(J(\mathbf{X}), \chi_{k-1}\big)\big] \\ &= \left(\frac{1}{N_k}\sum_{\mathbf{x}\in\Lambda_k} \varphi^k \tilde{\mathcal{H}}_k\big(J(\mathbf{x})\big)\tilde{I}\big(J(\mathbf{x}), \chi_k\big) - \frac{1}{N_k}\sum_{\mathbf{x}\in\Lambda_k} \varphi^k \tilde{\mathcal{H}}_k\big(J(\mathbf{x})\big)\tilde{I}\big(J(\mathbf{x}), \chi_{k-1}\big)\right) \\ &\quad + \left(\frac{1}{N_k}\sum_{\mathbf{x}\in\Lambda_k} \varphi^k \tilde{\mathcal{H}}_k\big(J(\mathbf{x})\big)\tilde{I}\big(J(\mathbf{x}), \chi_{k-1}\big)\right. \\ &\quad \left. - \tilde{E}_{\tilde{\theta}_k}\big[\varphi^k \tilde{\mathcal{H}}_k\big(J(\mathbf{X})\big)\tilde{I}\big(J(\mathbf{X}), \chi_{k-1}\big)\big]\right). \end{aligned}$$

Let Ω_2 be defined as before (see the proof of Lemma 4.15). For each $\omega \in \Omega_2$, it is easy to see that there exists $\mathcal{N}(\omega)$ such that for all $k \ge \mathcal{N}(\omega)$,

$$\frac{1}{N_k}\sum_{\mathbf{x}\in\Lambda_k} \varphi^k \tilde{\mathcal{H}}_k\big(J(\mathbf{x})\big)\tilde{I}\big(J(\mathbf{x}), \chi_k\big) - \frac{1}{N_k}\sum_{\mathbf{x}\in\Lambda_k} \varphi^k \tilde{\mathcal{H}}_k\big(J(\mathbf{x})\big)\tilde{I}\big(J(\mathbf{x}), \chi_{k-1}\big) = 0. \tag{4.37}$$

We denote by $\mathcal{U}_k$ the event that the total number of visits to Step 3a or 3b of MRAS$_2$ is less than or equal to $\sqrt{k}$ at the kth iteration of the algorithm, and by $\mathcal{W}_k$ the event that $\{J(\mathbf{x}) \ge \chi_{k-1} - \varepsilon\} \cap \mathcal{X} \subseteq \Xi$. For any $\xi > 0$, let $\mathcal{Q}_k$ be the event

$$\left|\frac{1}{N_k}\sum_{\mathbf{x}\in\Lambda_k} \varphi^k \tilde{\mathcal{H}}_k\big(J(\mathbf{x})\big)\tilde{I}\big(J(\mathbf{x}), \chi_{k-1}\big) - \tilde{E}_{\tilde{\theta}_k}\big[\varphi^k \tilde{\mathcal{H}}_k\big(J(\mathbf{X})\big)\tilde{I}\big(J(\mathbf{X}), \chi_{k-1}\big)\big]\right| \ge \xi.$$

Note that we have $P(\mathcal{U}_k^c \text{ i.o.}) = 0$ by Lemma 4.13, and $P(\mathcal{W}_k^c \text{ i.o.}) = 0$ by Assumption B1. Therefore,

$$\begin{aligned} P(\mathcal{Q}_k \text{ i.o.}) &= P\big(\{\mathcal{Q}_k \cap \mathcal{U}_k\} \cup \big\{\mathcal{Q}_k \cap \mathcal{U}_k^c\big\} \text{ i.o.}\big) \\ &= P(\mathcal{Q}_k \cap \mathcal{U}_k \text{ i.o.}) \\ &= P\big(\{\mathcal{Q}_k \cap \mathcal{U}_k \cap \mathcal{W}_k\} \cup \big\{\mathcal{Q}_k \cap \mathcal{U}_k \cap \mathcal{W}_k^c\big\} \text{ i.o.}\big) \\ &= P(\mathcal{Q}_k \cap \mathcal{U}_k \cap \mathcal{W}_k \text{ i.o.}). \end{aligned} \tag{4.38}$$

From Assumption B6, it is easy to see that the event $\mathcal{W}_k$ implies that the support $[a_k, b_k]$ of the random variable $\varphi^k \tilde{\mathcal{H}}_k(J(\mathbf{x}))\tilde{I}(J(\mathbf{x}), \chi_{k-1})$, $\mathbf{x} \in \Lambda_k$ satisfies $[a_k, b_k] \subseteq [0, \frac{(\varphi \mathcal{H}^*)^k}{\lambda f_*}]$. Moreover, conditional on $\tilde{\theta}_k$ and χ_{k-1}, $\mathbf{X}_k^1, \ldots, \mathbf{X}_k^{N_k}$ are i.i.d. random variables with common density $\tilde{f}(\cdot, \tilde{\theta}_k)$, we have, by the Hoeffding inequality,

$$\begin{aligned} P(\mathcal{Q}_k | \mathcal{W}_k, \tilde{\theta}_k = \theta, \chi_{k-1} = \chi) &\le 2\exp\left(\frac{-2N_k\xi^2}{(b_k - a_k)^2}\right) \\ &\le 2\exp\left(\frac{-2N_k\xi^2\lambda^2 f_*^2}{(\varphi\mathcal{H}^*)^{2k}}\right) \quad \forall k = 1, 2, \ldots. \end{aligned}$$

Thus,

$$\begin{aligned} P(\mathcal{Q}_k \cap \mathcal{W}_k) &= \int_{\theta,\chi} P(\mathcal{Q}_k \cap \mathcal{W}_k | \tilde{\theta}_k = \theta, \chi_{k-1} = \chi) f_{\tilde{\theta}_k, \chi_{k-1}}(d\theta, d\chi) \\ &\le \int_{\theta,\chi} P(\mathcal{Q}_k | \mathcal{W}_k, \tilde{\theta}_k = \theta, \chi_{k-1} = \chi) f_{\tilde{\theta}_k, \chi_{k-1}}(d\theta, d\chi) \\ &\le 2\exp\left(\frac{-2N_k\xi^2\lambda^2 f_*^2}{(\varphi\mathcal{H}^*)^{2k}}\right), \end{aligned}$$

where $f_{\tilde{\theta}_k, \chi_{k-1}}(\cdot, \cdot)$ is the joint distribution of random variables $\tilde{\theta}_k$ and χ_{k-1}. It follows that

$$\begin{aligned} P(\mathcal{Q}_k \cap \mathcal{U}_k \cap \mathcal{W}_k) &\le P(\mathcal{Q}_k \cap \mathcal{W}_k | \mathcal{U}_k) \\ &\le 2\exp\left(\frac{-2\alpha^{k-\sqrt{k}} N_0 \xi^2 \lambda^2 f_*^2}{(\varphi\mathcal{H}^*)^{2k}}\right) \\ &\le 2\exp\left(\frac{-2N_0\xi^2 f_*^2\lambda^2}{\alpha^{\sqrt{k}}}\left(\frac{\alpha}{(\varphi\mathcal{H}^*)^2}\right)^k\right), \end{aligned}$$

where the second inequality above follows because the event $\mathcal{U}_k$ implies that the total number of visits to Step 3c is greater than $k - \sqrt{k}$.

Moreover, since $e^{-\mathbf{x}} < 1/\mathbf{x}\ \forall \mathbf{x} > 0$, we have

$$P(\mathcal{Q}_k \cap \mathcal{U}_k \cap \mathcal{W}_k) < \frac{\alpha^{\sqrt{k}}}{N_0 \xi^2 f_*^2 \lambda^2} \left(\frac{(\varphi \mathcal{H}^*)^2}{\alpha} \right)^k = \frac{1}{N_0 \xi^2 f_*^2 \lambda^2} \left(\frac{\alpha^{\sqrt{k}/k} (\varphi \mathcal{H}^*)^2}{\alpha} \right)^k.$$

By assumption, we have $\frac{(\varphi \mathcal{H}^*)^2}{\alpha} < 1$. Thus, there exist $\delta < 1$ and $\mathcal{T}_\delta > 0$ such that $\alpha^{\sqrt{k}/k} \frac{(\varphi \mathcal{H}^*)^2}{\alpha} \leq \delta\ \forall k \geq \mathcal{T}_\delta$. Therefore,

$$\sum_{k=1}^{\infty} P(\mathcal{Q}_k \cap \mathcal{U}_k \cap \mathcal{W}_k) < \mathcal{T}_\delta + \frac{1}{N_0 \xi^2 f_*^2 \lambda^2} \sum_{k=\mathcal{T}_\delta}^{\infty} \delta^k < \infty.$$

Thus, we have by the Borel–Cantelli lemma

$$P(\mathcal{Q}_k \cap \mathcal{U}_k \cap \mathcal{W}_k \text{ i.o.}) = 0,$$

which implies that $P(\mathcal{Q}_k \text{ i.o.}) = 0$ by Eq. (4.38). Since $\xi > 0$ is arbitrary, we have

$$\left| \frac{1}{N_k} \sum_{\mathbf{x} \in \Lambda_k} \varphi^k \tilde{\mathcal{H}}_k\big(J(\mathbf{x})\big) \tilde{I}\big(J(\mathbf{x}), \chi_{k-1}\big) - \tilde{E}_{\tilde{\theta}_k}\big[\varphi^k \tilde{\mathcal{H}}_k\big(J(\mathbf{X})\big) \tilde{I}\big(J(\mathbf{X}), \chi_{k-1}\big)\big] \right| \to 0 \quad \text{w.p.1.} \tag{4.39}$$

Therefore, by combining (4.37) and (4.39), we have

$$\frac{1}{N_k} \sum_{\mathbf{x} \in \Lambda_k} \varphi^k \tilde{\mathcal{H}}_k\big(J(\mathbf{x})\big) \tilde{I}\big(J(\mathbf{x}), \chi_k\big) \to \tilde{E}_{\tilde{\theta}_k}\big[\varphi^k \tilde{\mathcal{H}}_k\big(J(\mathbf{X})\big) \tilde{I}\big(J(\mathbf{X}), \chi_{k-1}\big)\big] \quad \text{w.p.1.} \tag{4.40}$$

The same argument can also be used to show that

$$\frac{1}{N_k} \sum_{\mathbf{x} \in \Lambda_k} \varphi^k \tilde{\mathcal{H}}_k\big(J(\mathbf{x})\big) \tilde{I}\big(J(\mathbf{x}), \chi_k\big) \Upsilon(\mathbf{x}) \to \tilde{E}_{\tilde{\theta}_k}\big[\varphi^k \tilde{\mathcal{H}}_k\big(J(\mathbf{X})\big) \tilde{I}\big(J(\mathbf{X}), \chi_{k-1}\big) \Upsilon(\mathbf{X})\big] \to 0 \quad \text{w.p.1.}$$

And because $\liminf_{k \to \infty} \tilde{E}_{\tilde{\theta}_k}[\varphi^k \tilde{\mathcal{H}}_k(J(\mathbf{X})) \tilde{I}(J(\mathbf{X}), \chi_{k-1})] > 0$ (see (4.36)), we have [i] $\to 0$ w.p.1 as $k \to \infty$.

To complete the proof, we still need to show that the second term [ii] $\to 0$ w.p.1 as $k \to \infty$. Note that

$$[\text{ii}] = \frac{\frac{1}{N_k} \sum_{\mathbf{x} \in \Lambda_k} \varphi^k \tilde{\mathcal{H}}_k(\bar{\mathcal{J}}_k(\mathbf{x})) \tilde{I}(\bar{\mathcal{J}}_k(\mathbf{x}), \bar{\chi}_k) \Upsilon(\mathbf{x})}{\frac{1}{N_k} \sum_{\mathbf{x} \in \Lambda_k} \varphi^k \tilde{\mathcal{H}}_k(\bar{\mathcal{J}}_k(\mathbf{x})) \tilde{I}(\bar{\mathcal{J}}_k(\mathbf{x}), \bar{\chi}_k)} - \frac{\frac{1}{N_k} \sum_{\mathbf{x} \in \Lambda_k} \varphi^k \tilde{\mathcal{H}}_k(J(\mathbf{x})) \tilde{I}(J(\mathbf{x}), \chi_k) \Upsilon(\mathbf{x})}{\frac{1}{N_k} \sum_{\mathbf{x} \in \Lambda_k} \varphi^k \tilde{\mathcal{H}}_k(J(\mathbf{x})) \tilde{I}(J(\mathbf{x}), \chi_k)}.$$

From (4.36) and (4.40), it is easy to see that

$$\liminf_{k\to\infty} \frac{1}{N_k} \sum_{\mathbf{x}\in\Lambda_k} \varphi^k \tilde{\mathcal{H}}_k\big(J(\mathbf{x})\big)\tilde{I}\big(J(\mathbf{x}), \chi_k\big) > 0 \quad \text{w.p.1.}$$

Therefore, to prove that [ii] $\to 0$ w.p.1, it is sufficient to show that w.p.1

$$\frac{1}{N_k} \sum_{\mathbf{x}\in\Lambda_k} \varphi^k \tilde{\mathcal{H}}_k\big(\bar{\mathcal{J}}_k(\mathbf{x})\big)\tilde{I}\big(\bar{\mathcal{J}}_k(\mathbf{x}), \bar{\chi}_k\big) \to \frac{1}{N_k} \sum_{\mathbf{x}\in\Lambda_k} \varphi^k \tilde{\mathcal{H}}_k\big(J(\mathbf{x})\big)\tilde{I}\big(J(\mathbf{x}), \chi_k\big)$$

and

$$\begin{aligned}&\frac{1}{N_k} \sum_{\mathbf{x}\in\Lambda_k} \varphi^k \tilde{\mathcal{H}}_k\big(\bar{\mathcal{J}}_k(\mathbf{x})\big)\tilde{I}\big(\bar{\mathcal{J}}_k(\mathbf{x}), \bar{\chi}_k\big)\Upsilon(\mathbf{x})\\ &\quad\to \frac{1}{N_k} \sum_{\mathbf{x}\in\Lambda_k} \varphi^k \tilde{\mathcal{H}}_k\big(J(\mathbf{x})\big)\tilde{I}\big(J(\mathbf{x}), \chi_k\big)\Upsilon(\mathbf{x}).\end{aligned}$$

We have

$$\begin{aligned}&\left|\frac{1}{N_k} \sum_{\mathbf{x}\in\Lambda_k} \varphi^k \tilde{\mathcal{H}}_k\big(\bar{\mathcal{J}}_k(\mathbf{x})\big)\tilde{I}\big(\bar{\mathcal{J}}_k(\mathbf{x}), \bar{\chi}_k\big) - \frac{1}{N_k} \sum_{\mathbf{x}\in\Lambda_k} \varphi^k \tilde{\mathcal{H}}_k\big(J(\mathbf{x})\big)\tilde{I}\big(J(\mathbf{x}), \chi_k\big)\right|\\ &\le \left|\frac{1}{N_k} \sum_{\mathbf{x}\in\Lambda_k} \varphi^k \tilde{\mathcal{H}}_k\big(\bar{\mathcal{J}}_k(\mathbf{x})\big)\tilde{I}\big(\bar{\mathcal{J}}_k(\mathbf{x}), \bar{\chi}_k\big) - \frac{1}{N_k} \sum_{\mathbf{x}\in\Lambda_k} \varphi^k \tilde{\mathcal{H}}_k\big(J(\mathbf{x})\big)\tilde{I}\big(\bar{\mathcal{J}}_k(\mathbf{x}), \bar{\chi}_k\big)\right|\\ &\quad + \left|\frac{1}{N_k} \sum_{\mathbf{x}\in\Lambda_k} \varphi^k \tilde{\mathcal{H}}_k\big(J(\mathbf{x})\big)\tilde{I}\big(\bar{\mathcal{J}}_k(\mathbf{x}), \bar{\chi}_k\big)\right.\\ &\quad \left. - \frac{1}{N_k} \sum_{\mathbf{x}\in\Lambda_k} \varphi^k \tilde{\mathcal{H}}_k\big(J(\mathbf{x})\big)\tilde{I}\big(J(\mathbf{x}), \chi_k\big)\right|. \qquad (4.41)\end{aligned}$$

The first term on the right-hand side of inequality (4.41) is bounded by

$$\begin{aligned}&\frac{\varphi^k}{N_k} \sum_{\mathbf{x}\in\Lambda_k} \frac{|\tilde{\mathcal{H}}_k(\bar{\mathcal{J}}_k(\mathbf{x})) - \tilde{\mathcal{H}}_k(J(\mathbf{x}))|}{\tilde{\mathcal{H}}_k(J(\mathbf{x}))} \tilde{\mathcal{H}}_k\big(J(\mathbf{x})\big)\tilde{I}\big(\bar{\mathcal{J}}_k(\mathbf{x}), \bar{\chi}_k\big)\\ &= \frac{\varphi^k}{N_k} \sum_{\mathbf{x}\in\Lambda_k} \frac{|[\mathcal{H}(\bar{\mathcal{J}}_k(\mathbf{x}))]^k - [\mathcal{H}(J(\mathbf{x}))]^k|}{[\mathcal{H}(J(\mathbf{x}))]^k} \frac{[\mathcal{H}(J(\mathbf{x}))]^k}{\tilde{f}(\mathbf{x}, \tilde{\theta}_k)} \tilde{I}\big(\bar{\mathcal{J}}_k(\mathbf{x}), \bar{\chi}_k\big)\\ &\le \frac{(\varphi\mathcal{H}^*)^k}{N_k \lambda f_*} \sum_{\mathbf{x}\in\Lambda_k} \frac{|[\mathcal{H}(\bar{\mathcal{J}}_k(\mathbf{x}))]^k - [\mathcal{H}(J(\mathbf{x}))]^k|}{[\mathcal{H}(J(\mathbf{x}))]^k} \quad \forall k \ge \mathcal{N}(\omega), \qquad (4.42)\end{aligned}$$

for all $\omega \in \Omega_3$ by Assumptions B1 and B6, where we recall from the proof of Lemma 4.15 that Ω_3 is the set of sample paths such that $\lim_{k\to\infty}\{\mathbf{x} : J(\mathbf{x}) \ge \chi_k - \varepsilon\} \cap \mathcal{X} \subseteq \Xi$.

Letting Δ_k be defined as in Assumption B4, we have

$$\begin{aligned} P\Big(\max_{\mathbf{x}\in\Lambda_k}\big|\bar{\mathcal{J}}_k(\mathbf{x})-J(\mathbf{x})\big|\geq\Delta_k\Big) &\leq P\Big(\bigcup_{\mathbf{x}\in\Lambda_k}\big\{\big|\bar{\mathcal{J}}_k(\mathbf{x})-J(\mathbf{x})\big|\geq\Delta_k\big\}\Big)\\ &\leq \sum_{\mathbf{x}\in\Lambda_k} P\big(\big|\bar{\mathcal{J}}_k(\mathbf{x})-J(\mathbf{x})\big|\geq\Delta_k\big)\\ &\leq |\Lambda_k|\sup_{\mathbf{x}\in\mathcal{X}} P\big(\big|\bar{\mathcal{J}}_k(\mathbf{x})-J(\mathbf{x})\big|\geq\Delta_k\big)\\ &\leq \alpha^k N_0\phi(M_k,\Delta_k)\quad \text{by Assumption L1}\\ &\leq N_0\big(\delta^*\big)^k\quad \forall k\geq\mathcal{K}\text{ by Assumption L4.}\end{aligned}$$

Furthermore,

$$\sum_{k=1}^{\infty} P\Big(\max_{\mathbf{x}\in\Lambda_k}\big|\bar{\mathcal{J}}_k(\mathbf{x})-J(\mathbf{x})\big|\geq\Delta_k\Big)\leq\mathcal{K}+N_0\sum_{k=\mathcal{K}}^{\infty}\big(\delta^*\big)^k<\infty,$$

which implies that $P(\max_{\mathbf{x}\in\Lambda_k}|\bar{\mathcal{J}}_k(\mathbf{x})-J(\mathbf{x})|\geq\Delta_k \text{ i.o.})=0$ by the Borel–Cantelli lemma.

Let $\Omega_4:=\{\omega:\max_{\mathbf{x}\in\Lambda_k}|\bar{\mathcal{J}}_k(\mathbf{x})-J(\mathbf{x})|<\Delta_k \text{ i.o.}\}$. For each $\omega\in\Omega_3\cap\Omega_4$, we see that the right-hand side of inequality (4.42) is further bounded by

$$\begin{aligned} &\frac{(\varphi\mathcal{H}^*)^k}{\lambda f_* N_k}\sum_{\mathbf{x}\in\Lambda_k} L_k\big|\bar{\mathcal{J}}_k(\mathbf{x})-J(\mathbf{x})\big|\quad \text{for sufficiently large } k\text{, by Assumption B4}\\ &\leq \frac{(\varphi\mathcal{H}^*)^k}{\lambda f_*}L_k\max_{\mathbf{x}\in\Lambda_k}\big|\bar{\mathcal{J}}_k(\mathbf{x})-J(\mathbf{x})\big|\\ &\leq \frac{\alpha^{k/2}L_k}{\lambda f_*}\max_{\mathbf{x}\in\Lambda_k}\big|\bar{\mathcal{J}}_k(\mathbf{x})-J(\mathbf{x})\big|,\quad \text{since } \alpha>\big(\varphi\mathcal{H}^*\big)^2.\end{aligned}$$

Note that for any given $\zeta>0$,

$$P\Big\{\alpha^{k/2}L_k\max_{\mathbf{x}\in\Lambda_k}\big|\bar{\mathcal{J}}_k(\mathbf{x})-J(\mathbf{x})\big|\geq\zeta\Big\}\leq\sum_{\mathbf{x}\in\Lambda_k}P\Big\{\big|\bar{\mathcal{J}}_k(\mathbf{x})-J(\mathbf{x})\big|\geq\frac{\zeta}{\alpha^{k/2}L_k}\Big\}.$$

And by using Assumption L4 and a similar argument as in the proof for Proposition 4.16, it is easy to show that

$$\alpha^{k/2}L_k\max_{\mathbf{x}\in\Lambda_k}\big|\bar{\mathcal{J}}_k(\mathbf{x})-J(\mathbf{x})\big|\to 0\quad\text{w.p.1.}$$

Let $\Omega_5:=\{\omega:\alpha^{k/2}L_k\max_{\mathbf{x}\in\Lambda_k}|\bar{\mathcal{J}}_k(\mathbf{x})-J(\mathbf{x})|\to 0\}$. Since $P(\Omega_3\cap\Omega_4\cap\Omega_5)\geq 1-P(\Omega_3^c)-P(\Omega_4^c)-P(\Omega_5^c)=1$, it follows that the first term on the right-hand side of inequality (4.41) $\to 0$ as $k\to\infty$ w.p.1.

On the other hand, the second term on the right-hand side of inequality (4.41) is bounded by

$$\begin{aligned}
&\frac{\varphi^k}{N_k}\sum_{\mathbf{x}\in\Lambda_k}\tilde{\mathcal{H}}_k\big(J(\mathbf{x})\big)\big|\tilde{I}\big(\bar{\mathcal{J}}_k(\mathbf{x}),\bar{\chi}_k\big)-\tilde{I}\big(J(\mathbf{x}),\chi_k\big)\big| \\
&\quad\le\frac{(\varphi\mathcal{H}^*)^k}{N_k\lambda f_*}\sum_{\mathbf{x}\in\Lambda_k}\big|\tilde{I}\big(\bar{\mathcal{J}}_k(\mathbf{x}),\bar{\chi}_k\big)-\tilde{I}\big(J(\mathbf{x}),\bar{\chi}_k\big)\big|+\big|\tilde{I}\big(J(\mathbf{x}),\bar{\chi}_k\big)-\tilde{I}\big(J(\mathbf{x}),\chi_k\big)\big| \\
&\qquad\forall k\ge\mathcal{N}(\omega),\ \omega\in\Omega_3 \\
&\quad\le\frac{(\varphi\mathcal{H}^*)^k}{N_k\lambda f_*}\frac{1}{\varepsilon}\sum_{\mathbf{x}\in\Lambda_k}\big[\big|\bar{\mathcal{J}}_k(\mathbf{x})-J(\mathbf{x})\big|+|\bar{\chi}_k-\chi_k|\big]\quad\text{by the definition of }\tilde{I}(\cdot,\cdot) \\
&\quad\le\frac{\alpha^{k/2}}{\lambda f_*\varepsilon}\max_{\mathbf{x}\in\Lambda_k}\big|\bar{\mathcal{J}}_k(\mathbf{x})-J(\mathbf{x})\big|+\frac{\alpha^{k/2}}{\lambda f_*\varepsilon}|\bar{\chi}_k-\chi_k|\to 0\quad\text{w.p.1},
\end{aligned}$$

by Assumption L4 and Proposition 4.16.

By repeating the above argument, we can also show that

$$\begin{aligned}
&\frac{1}{N_k}\sum_{\mathbf{x}\in\Lambda_k}\varphi^k\tilde{\mathcal{H}}_k\big(\bar{\mathcal{J}}_k(\mathbf{x})\big)\tilde{I}\big(\bar{\mathcal{J}}_k(\mathbf{x}),\bar{\chi}_k\big)\Upsilon(\mathbf{x}) \\
&\qquad\to\frac{1}{N_k}\sum_{\mathbf{x}\in\Lambda_k}\varphi^k\tilde{\mathcal{H}}_k\big(J(\mathbf{x})\big)\tilde{I}\big(J(\mathbf{x}),\chi_k\big)\Upsilon(\mathbf{x})\quad\text{w.p.1}.
\end{aligned}$$

Thus, we have [ii] $\to 0$ as $k\to\infty$ w.p.1.

Hence the proof is completed by applying Lemmas 4.14 and 4.15. □

Proofs of the following results for the multivariate normal and independent univariate cases are straightforward and hence omitted.

Corollary 4.18 (Multivariate Normal) *For continuous optimization problems in* $\Re^n$, *if multivariate normal p.d.f.s are used in* MRAS$_2$, *i.e.*,

$$f(\mathbf{x},\tilde{\theta}_k)=\frac{1}{\sqrt{(2\pi)^n|\tilde{\Sigma}_k|}}\exp\left(-\frac{1}{2}(\mathbf{x}-\tilde{\mu}_k)^T\tilde{\Sigma}_k^{-1}(\mathbf{x}-\tilde{\mu}_k)\right),$$

where $\tilde{\theta}_k:=(\tilde{\mu}_k;\ \tilde{\Sigma}_k)$, *Assumptions* L1–L4, B1–B5 *are satisfied, and* $\alpha>(\varphi\mathcal{H}^*)^2$, *then*

$$\lim_{k\to\infty}\tilde{\mu}_k=\mathbf{x}^*,\quad\textit{and}\quad\lim_{k\to\infty}\tilde{\Sigma}_k=0_{n\times n}\quad w.p.1,$$

where $0_{n\times n}$ *represents an n-by-n zero matrix.*

Corollary 4.19 (Independent Univariate) *If the components of the random vector* $\mathbf{X}$ *are independent, each has a univariate density/mass function of the form*

$$f(\mathsf{x}_i, \tilde{\vartheta}_i) = \exp\big(\mathsf{x}_i \tilde{\vartheta}_i - K(\tilde{\vartheta}_i)\big) h(\mathsf{x}_i), \quad \mathsf{x}_i \in \Re,\ \tilde{\vartheta}_i \subset \Re,\ \forall i = 1, \ldots, n,$$

Assumptions L1–L4, B1–B6 *are satisfied, and* $\alpha > (\varphi \mathcal{H}^*)^2$, *then*

$$\lim_{k\to\infty} m(\tilde{\theta}_k) = \lim_{k\to\infty} E_{\tilde{\theta}_k}[\mathbf{X}] = \mathbf{x}^* \quad w.p.1,\ \textit{where}\ \tilde{\theta}_k := \big(\tilde{\vartheta}_1^k, \ldots, \tilde{\vartheta}_n^k\big).$$

4.3 Application of MRAS to MDPs via Direct Policy Learning

In this section, we apply the MRAS method to solving MDPs with finite state spaces. Central to the context of this section are that the underlying transition distribution and/or the one-stage cost function are unknown and that only simulation samples are available. Thus, standard techniques like value iteration and policy iteration are not directly applicable. There is a class of well-established approaches, often referred to as reinforcement learning (see Chap. 1 for a description and references), that have proven useful for attacking these types of problem. Many reinforcement learning algorithms, including the well-known Q-learning, work directly with the value function or Q-function, and iteratively update/learn these quantities in order to better approximate the true optimal value function. Thus, these algorithms can be viewed as variants of value iteration in simulation settings. In contrast, the idea of this section is to interpret an MDP as a stochastic optimization problem on the (randomized) policy space, and then use MRAS as a specific optimization strategy to directly search the policy space to find good policies. We focus on the MRAS_2 algorithm proposed in Sect. 4.1.3, and consider both finite- and infinite-horizon MDPs. The key in applying MRAS_2 to both settings is to define an appropriate (parameterized) distribution on the policy space, so that random samples (policies) can be generated efficiently.

4.3.1 Finite-Horizon MDPs

In the finite-horizon setting, since the optimal policy is in general non-stationary, we need to define the parameterized sampling distributions on the space of non-stationary policies. This step can be carried out by specifying at each stage $t \in \{0, \ldots, H-1\}$, a parameterized marginal distribution $f(a, \theta^{t,x})\ \forall a \in A(x)$ for each state $x \in X$, where $\theta^{t,x}$ is a parameter that depends on both t and x. Note that when the action space is finite, $f(\cdot, \theta^{t,x})$ becomes a p.m.f. with $\sum_{a\in A(x)} f(a, \theta^{t,x}) = 1\ \forall x, t$, where $f(a, \theta^{t,x})$ represents the probability of taking action a at state x and stage t. Once these marginal distributions are specified, we can generate N_k policies at the kth iteration of MRAS_2, simulate these N_k policies, and calculate their

corresponding cumulative rewards. A detailed description of these steps is given in Fig. 4.6, where for simplicity, we have assumed that all simulations start from a given initial state x_0.

Given N_k deterministic policies $\pi^i,\ i = 1, \ldots, N_k$, and their corresponding value function estimates $\bar{V}_k^{\pi^i}$, we can directly apply the MRAS$_2$ algorithm with $\bar{\mathcal{J}}_k(\mathbf{X}_k^i)$ in Fig. 4.5 replaced by $\bar{V}_k^{\pi^i}$ to update the parameters associated with the marginal distributions. As discussed in Sect. 4.1.1, we restrict $f(\cdot, \theta^{t,x})$ to natural exponential families, so that the parameter updating (4.43) can be carried out in analytical form, which makes the algorithm easy to implement. It is worth mentioning that when applying MRAS$_2$, the distribution $\mathbf{f}(\cdot, \tilde{\theta}_k)$ that appears in Eq. (4.43) is a joint distribution, which is the product of marginal distributions $\tilde{f}(\cdot, \tilde{\theta}_k^{t,x})$. Thus, in the finite action space case, $\mathbf{f}(\pi^i, \tilde{\theta}_k)$ would actually indicate the probability that policy π^i is generated given the current parameter $\tilde{\theta}_k = \{\theta_k^{t,x},\ \forall t, x\}$.

This algorithm can be used in an on-line manner, in the same way as the Simulated Annealing Multiplicative Weights (SAMW) algorithm presented in Sect. 5.1, to approximately solve large or infinite-horizon MDPs.

4.3.2 Infinite-Horizon MDPs

In the infinite-horizon setting, since we are working with stationary policies, the policy generation and reward collection step is easier than that in the finite-horizon case. However, except for shortest path problems, there is a slight complication in simulating a given policy, because there is no explicit terminal stage/state in a general infinite-horizon setting. One simple way to address this issue is to use the cumulative reward of a finite but large horizon problem to approximate the reward of the infinite-horizon problem. Precisely, for a given (stationary) policy π, let $\{x_0, \pi(x_0), w_0, x_1, \pi(x_1), w_1 \ldots\}$ be a particular sample path of an infinite-horizon MDP with cumulative reward $\sum_{t=0}^{\infty} \gamma^t R'(x_t, \pi(x_t), w_t)$. For any given $\delta > 0$, it can be easily verified that if we simulate for at least $T_\delta \geq \lceil \ln \frac{\delta(1-\gamma)}{R_{\max}} / \ln \gamma \rceil$ periods, then the reward accumulated on the finite path $\{x_0, \pi(x_0), w_0, \ldots, x_{T_\delta}, \pi(x_{T_\delta}), w_{T_\delta}\}$ can at most differ from $\sum_{t=0}^{\infty} \gamma^t R'(x_t, \pi(x_t), w_t)$ by δ. Thus, when estimating value functions in Fig. 4.7, each sample path will only be simulated for a finite horizon length T_δ. All other components in Fig. 4.7 essentially remain the same as in the finite-horizon case, except that the marginal distributions $f(\cdot, \theta^x)$ are now stationary.

4.3.3 MDPs with Large State Spaces

For MDPs with large state spaces, the aforementioned policy learning approach may lead to computational inefficiency, since the size of the policy space to be searched

MRAS$_2$ **for Finite-Horizon MDPs**

Input: $\rho_0 \in (0,1]$, $N_0 > 1$, $\varepsilon > 0$, $\alpha > 1$, $\lambda \in (0,1)$, strictly increasing function $\mathcal{H}$: $\Re \to \Re^+$; simulation allocation rule $\{M_k\}$, initial state $x_0 \in X$, family of distributions $\{f(\cdot, \theta^{t,x}),\ x \in X,\ t = 0, \ldots, H-1\}$ with $\theta_0^{t,x}$ s.t. $f(a, \theta_0^{t,x}) > 0\ \forall a \in A(x),\ x \in X,$ $t = 0, \ldots, H-1$.

Initialization: Set iteration count $k = 0$; $\tilde{\theta}_0^{t,x} = \theta_0^{t,x}\ \forall t, x$.

Loop until Stopping Rule is satisfied:

- Construct N_k non-stationary policies $\pi^i = \{\pi_0^i, \ldots, \pi_{H-1}^i\}$, $i = 1, \ldots, N_k$.
  - With probability $1-\lambda$, generate π^i by sampling actions
    $$\pi_t^i(x) \sim f(\cdot, \tilde{\theta}_k^{t,x})\ \forall x \in X,\ t = 0, \ldots, H-1.$$
  - With probability λ, generate π^i by sampling actions
    $$\pi_t^i(x) \sim f(\cdot, \tilde{\theta}_0^{t,x})\ \forall x \in X,\ t = 0, \ldots, H-1.$$

  Let $\Lambda_k = \{\pi^1, \ldots, \pi^{N_k}\}$ be the set of N_k generated policies. Simulate for M_k paths, each policy $\pi \in \Lambda_k$ starting from x_0; calculate the averaged reward $\bar{V}_k^{\pi} = \frac{1}{M_k}\sum_{j=1}^{M_k} V_{k,j}^{\pi}$, where $V_{k,j}^{\pi}$ is the cumulative reward obtained at the jth simulation of policy π.
- Compute the sample $(1-\rho_k)$-quantile:
  $$\tilde{\chi}_k(\rho_k, N_k) = \bar{V}_{(\lceil (1-\rho_k)N_k \rceil)},$$
  where $\bar{V}_{(i)}$ is the ith order statistic of $\{\bar{V}_k^{\pi^i},\ i = 1, \ldots, N_k\}$.
- **if** $k = 0$ **or** $\tilde{\chi}_k(\rho_k, N_k) \geq \bar{\chi}_{k-1} + \varepsilon$, **then**
  - set $\bar{\chi}_k = \tilde{\chi}_k(\rho_k, N_k),\ \rho_{k+1} = \rho_k,\ N_{k+1} = N_k,$ $\mathbf{X}_k^* = \mathbf{X}_{1-\rho_k}$, where $\mathbf{X}_{1-\rho_k} \in \{\pi \in \Lambda_k : \bar{V}_k^{\pi} = \tilde{\chi}_k(\rho_k, N_k)\}$;

  else, find the largest $\bar{\rho} \in (0, \rho_k)$ such that $\tilde{\chi}_k(\bar{\rho}, N_k) \geq \bar{\chi}_{k-1} + \varepsilon$;
  - **if** $\bar{\rho}$ exists, **then** set $\bar{\chi}_k = \tilde{\chi}_k(\bar{\rho}, N_k),\ \rho_{k+1} = \bar{\rho},\ N_{k+1} = N_k,$ $\mathbf{X}_k^* = \mathbf{X}_{1-\bar{\rho}} \in \{\pi \in \Lambda_k :\ \bar{V}_k^{\pi} = \tilde{\chi}_k(\bar{\rho}, N_k)\}$;
  - **else**, set $\bar{\chi}_k = \bar{\mathcal{J}}_k(\mathbf{X}_{k-1}^*),\ \rho_{k+1} = \rho_k,\ N_{k+1} = \lceil \alpha N_k \rceil,$ $\mathbf{X}_k^* = \mathbf{X}_{k-1}^*$.

  endif
- Update parameter vector:
  $$\tilde{\theta}_{k+1}^{t,x} \in \arg\max_{\theta^{t,x} \in \Theta} \frac{1}{N_k} \sum_{\pi \in \Lambda_k} \frac{[\mathcal{H}(\bar{V}_k^{\pi})]^k}{\mathbf{f}(\pi, \tilde{\theta}_k)} \tilde{I}\left(\bar{V}_k^{\pi}, \bar{\chi}_k\right) \ln f\left(\pi_t(x), \theta^{t,x}\right) \quad \forall t, x, \tag{4.43}$$
  where $\mathbf{f}(\pi, \tilde{\theta}_k) = (1-\lambda) \prod_{x \in X} \prod_{t=0}^{H-1} f\left(\pi_t(x), \tilde{\theta}_k^{t,x}\right) + \lambda \prod_{x \in X} \prod_{t=0}^{H-1} f\left(\pi_t(x), \tilde{\theta}_0^{t,x}\right)$ and $\tilde{\theta}_k = \{\tilde{\theta}_k^{t,x},\ \forall t, x\}$.
- $k \leftarrow k+1$.

Output: $\tilde{\theta}_k$.

Fig. 4.6 Application of MRAS$_2$ to finite-horizon MDPs

MRAS$_2$ **for Infinite-Horizon MDPs**

Input: $\rho_0 \in (0,1]$, $N_0 > 1$, $\varepsilon > 0$, $\alpha > 1$, $\lambda \in (0,1)$, strictly increasing function $\mathcal{H}\colon \Re \to \Re^+$, simulation allocation rule $\{M_k\}$, initial state $x_0 \in X$, simulation horizon T_δ, family of (univariate) distributions $\{f(\cdot,\theta^x), x \in X\}$ with θ_0^x s.t. the initial marginal distributions $f(a,\theta_0^x) > 0\ \forall a \in A(x),\ x \in X$.

Initialization: Set iteration count $k = 0$; $\tilde{\theta}_0^x = \theta_0^x\ \forall x$.

Loop until Stopping Rule is satisfied:

- Construct a population of N_k stationary policies $\Lambda_k = \{\pi^1 \ldots, \pi^{N_k}\}$.
  - With probability $1-\lambda$, generate π^i by sampling actions
    $$\pi^i(x) \sim f(\cdot, \tilde{\theta}_k^x)\ \forall x \in X.$$
  - With probability λ, generate π^i by sampling actions
    $$\pi^i(x) \sim f(\cdot, \tilde{\theta}_0^x)\ \forall x \in X.$$

  Simulate for M_k paths, each policy $\pi \in \Lambda_k$, where each simulation starts from x_0 and lasts for T_δ periods. Calculate the averaged reward $\bar{V}_k^\pi = \frac{1}{M_k}\sum_{j=1}^{M_k} V_{k,j}^\pi$, where $V_{k,j}^\pi$ is the cumulative reward obtained at the jth simulation of policy π.
- Compute the sample $(1-\rho_k)$-quantile:
  $$\tilde{\chi}_k(\rho_k, N_k) = \bar{V}_{(\lceil (1-\rho_k)N_k \rceil)},$$
  where $\bar{V}_{(i)}$ is the ith order statistic of $\{\bar{V}_k^{\pi^i},\ i = 1, \ldots, N_k\}$.
- **if** $k = 0$ **or** $\tilde{\chi}_k(\rho_k, N_k) \geq \bar{\chi}_{k-1} + \varepsilon$, **then**
  - set $\bar{\chi}_k = \tilde{\chi}_k(\rho_k, N_k),\ \rho_{k+1} = \rho_k,\ N_{k+1} = N_k,$
    $\mathbf{X}_k^* = \mathbf{X}_{1-\rho_k}$, where $\mathbf{X}_{1-\rho_k} \in \{\pi \in \Lambda_k : \bar{V}_k^\pi = \tilde{\chi}_k(\rho_k, N_k)\}$;

  else, find the largest $\bar{\rho} \in (0, \rho_k)$ such that $\tilde{\chi}_k(\bar{\rho}, N_k) \geq \bar{\chi}_{k-1} + \varepsilon$;
  - **if** $\bar{\rho}$ exists, **then** set $\bar{\chi}_k = \tilde{\chi}_k(\bar{\rho}, N_k),\ \rho_{k+1} = \bar{\rho},\ N_{k+1} = N_k,$
    $\mathbf{X}_k^* = \mathbf{X}_{1-\bar{\rho}} \in \{\pi \in \Lambda_k :\ \bar{V}_k^\pi = \tilde{\chi}_k(\bar{\rho}, N_k)\}$;
  - **else**, set $\bar{\chi}_k = \bar{\mathcal{J}}_k(\mathbf{X}_{k-1}^*),\ \rho_{k+1} = \rho_k,\ N_{k+1} = \lceil \alpha N_k \rceil,$
    $\mathbf{X}_k^* = \mathbf{X}_{k-1}^*$.

  endif
- Update parameter vector:
  $$\tilde{\theta}_{k+1}^x \in \arg\max_{\theta^x \in \Theta} \frac{1}{N_k} \sum_{\pi \in \Lambda_k} \frac{[\mathcal{H}(\bar{V}_k^\pi)]^k}{\mathbf{f}(\pi, \tilde{\theta}_k)} \tilde{I}\left(\bar{V}_k^\pi, \bar{\chi}_k\right) \ln f\left(\pi(x), \theta^x\right) \quad \forall x, \tag{4.44}$$
  where $\mathbf{f}(\pi, \tilde{\theta}_k) = (1-\lambda) \prod_{x \in X} f\left(\pi(x), \tilde{\theta}_k^x\right) + \lambda \prod_{x \in X} f\left(\pi(x), \tilde{\theta}_0^x\right)$
  and $\tilde{\theta}_k = \{\tilde{\theta}_k^x,\ \forall x\}$.
- $k \leftarrow k+1$.

Output: $\tilde{\theta}_k$.

Fig. 4.7 Application of MRAS$_2$ to infinite-horizon MDPs

by the algorithm (assume the action space is finite) grows exponentially with the state space size. Moreover, in the finite-horizon setting, the policy space size will also increase exponentially with the horizon length. In these situations, it is useful to consider various compact representations of the policy space, which can be achieved either by explicit policy parameterization or by restricting the search to a subset of structured policies. In the former case, policies are parameterized, and thus determined, by a small number of parameters, and the policy learning via MRAS simply becomes optimization over the smaller parameter space. One such example is the classical (s, S) inventory control problem, where we parameterize control policies by a threshold pair (s, S), and search for the best thresholds (s^*, S^*) over all possible combinations of (s, S) rather than for the best policy among the set of all admissible policies. In the latter case, we often need to have an idea of what form good policies might take, either from practical experience or from some analysis; for example, in designing the optimal feedback control law for a linear quadratic Gaussian (LQG) system, if we know that a good control law is linear, then we can restrict the search to the set of linear controllers instead of the general non-linear controllers.

4.3.4 Numerical Examples

We now illustrate the performance of MRAS_2 on both finite- and infinite-horizon MDPs. In the former case, we consider the inventory control problem of Sect. 2.1.5. In the latter case, we apply the algorithm to the controlled queueing example introduced in Sect. 4.6.2.2. For MDPs with large state spaces, we also consider an inventory control problem with continuous demand, where the optimal policy is of (s, S)-type.

Note that since MRAS_2 was presented in a maximization context, the following slight modifications are required before it can be applied to minimization problems: (i) The performance function $\mathcal{H}$ needs to be initialized as a strictly decreasing function. Throughout this section, we take $\mathcal{H}(z) := e^{-\kappa z}$ for $z \in \Re$, where $\kappa > 0$ is a constant. (ii) The sample $(1-\rho_k)$-quantile $\tilde{\chi}_k$ will now be calculated by first ordering the sample performances $\bar{\mathcal{J}}_k(\mathbf{X}_k^i),\ i = 1, \ldots, N_k$ from largest to smallest, and then taking the $\lceil (1-\rho_k)N_k \rceil$th order statistic. (iii) The threshold function should now be modified as

$$\tilde{I}(z, \chi) := \begin{cases} 0 & \text{if } z \geq \chi + \varepsilon, \\ (\chi + \varepsilon - z)/\varepsilon & \text{if } \chi < z < \chi + \varepsilon, \\ 1 & \text{if } z \leq \chi. \end{cases}$$

(iv) The inequalities at Step 3 of MRAS_2 need to be replaced with $\tilde{\chi}_k(\rho_k, N_k) \leq \bar{\chi}_k - \varepsilon$ and $\tilde{\chi}_k(\bar{\rho}, N_k) \leq \bar{\chi}_k - \varepsilon$.

In actual implementation of MRAS_2, a smoothed parameter updating procedure is used, i.e., first a smoothed parameter vector $\hat{\theta}_{k+1}$ is computed at each iteration k

according to

$$\hat{\theta}_{k+1} := \upsilon\tilde{\theta}_{k+1} + (1-\upsilon)\hat{\theta}_k, \quad \forall k = 0, 1, \ldots, \text{ and } \hat{\theta}_0 := \tilde{\theta}_0,$$

where $\tilde{\theta}_{k+1}$ is the parameter computed at Step 4 of MRAS$_2$, and $\upsilon \in (0, 1]$ is a smoothing parameter; then $f(\cdot, \hat{\theta}_{k+1})$ (instead of $f(\cdot, \tilde{\theta}_{k+1})$) is used in Step 1 to generate new samples. Although this modification will not affect the theoretical convergence results, it may improve the numerical performance of the algorithm.

For numerical comparison purposes, we also applied a simple stochastic version of the standard simulated annealing (SAN) algorithm to all test cases, where each time the algorithm visits a solution, we allocate L simulation observations to that solution, estimate the performance of that solution by averaging over L replications, and then use standard SAN to solve the underlying problem. In the algorithm, the value of L is chosen large, so that a relatively precise estimate of the value function can be obtained during the search.

4.3.4.1 An Inventory Control Example

We consider the inventory control example of Sect. 2.1.5 with the following parameter settings: horizon $H = 3$; capacity $M = 20$; initial inventory $x_0 = 5$; demand $D_t \sim DU(0, 9)$; holding cost $h = 1$; penalty cost $p = 1$ and $p = 10$; fixed order cost $K = 5$; and the order amount can be any integer value up to the capacity level.

For this problem, we specify the marginal distributions in MRAS$_2$ via the use of an $|X| \times |A| \times H$ stochastic matrix $\mathbf{P}$, where the entry $\mathbf{P}(i, j, t)$ $i = 1, \ldots, |X|$, $j = 1, \ldots, |A|$, $t \in \{0, \ldots, H-1\}$ indicates the probability of taking action $a_j \in A$ at state $x_i \in X$ and stage t, and we set $\mathbf{P}(i, j, t) = 0$ for $a_j \notin A(x_i)$. Thus, at each iteration k of the algorithm, given the stochastic matrix $\mathbf{P}_k$, we can then sample N_k policies according to $\mathbf{P}_k$, and simulate their corresponding cumulative rewards. Note that when parameterized by the stochastic matrix $\mathbf{P}_k$, the probability of generating a non-stationary policy π is given by

$$f(\pi, \mathbf{P}_k) = \prod_{t=0}^{H-1} \prod_{i=1}^{|X|} \prod_{j=1}^{|A|} [\mathbf{P}(i, j, t)]^{I\{\pi \in \Pi_{i,j}(t)\}} = \prod_{t=0}^{H-1} \prod_{i=1}^{|X|} e^{(\theta^{i,t})^T \Upsilon^{i,t}(\pi)},$$

where $\Pi_{i,j}(t)$ denotes the set of policies which take action a_j at state x_i and stage t, $\theta^{i,t} := [\ln \mathbf{P}(i, 1, t), \ldots, \ln \mathbf{P}(i, |A|, t)]^T$, and $\Upsilon^{i,t}(\pi) := \{I\{\pi \in \Pi_{i,1}(t)\}, \ldots, I\{\pi \in \Pi_{i,|A|}(t)\}\}^T$. Thus, by Definition 4.2, the parameterized distribution $f(\cdot, \mathbf{P}_k)$ belongs to a NEF. Moreover, it is not difficult to show that the entries of $\mathbf{P}$ are updated in (4.43) as

$$\mathbf{P}_{k+1}(i, j, t) := \frac{\sum_{\pi \in \Lambda_k} \frac{[\mathcal{H}(\bar{V}_k^\pi)]^k}{\mathbf{f}(\pi, \mathbf{P}_k)} \tilde{I}(\bar{V}_k^\pi, \bar{\chi}_k) I\{\pi \in \Pi_{i,j}(t)\}}{\sum_{\pi \in \Lambda_k} \frac{[\mathcal{H}(\bar{V}_k^\pi)]^k}{\mathbf{f}(\pi, \mathbf{P}_k)} \tilde{I}(\bar{V}_k^\pi, \bar{\chi}_k)},$$

where $\Lambda_k := \{\pi^i, \ldots, \pi^{N_k}\}$ is the set of N_k policies generated, and $\mathbf{f}(\pi, \mathbf{P}_k) = (1-\lambda) f(\pi, \mathbf{P}_k) + \lambda f(\pi, \mathbf{P}_0)$. Thus, if there exists a unique optimal policy π^* and all the relevant conditions in Theorem 4.17 are satisfied, then a straightforward interpretation of the convergence result (4.35) yields

$$\lim_{k\to\infty} \mathbf{P}_k(i, j, t) = I\left\{\pi^* \in \Pi_{i,j}(t)\right\}, \quad \forall i, j, t,$$

which indicates that the sequence of stochastic matrices $\{\mathbf{P}_k\}$ will converge to a matrix $\mathbf{P}^*$ with all probability mass at the optimal policy π^*.

For both test cases, i.e., $p = 1$ and $p = 10$, our numerical results are based on the following parameter setting: $\mathbf{P}_0(i, j, t) = 1/|A(x_i)|\ \forall a_j \in A(x_i), t = 0, \ldots, H-1$ and $\mathbf{P}_0(i, j, t) = 0\ \forall a_j \notin A(x_i)$, $\varepsilon = 0.1$, initial sample size $N_0 = 100$, $\rho_0 = 0.1$, $\lambda = 0.01$, $\alpha = 1.04$, $\kappa = 0.1$, smoothing parameter $\upsilon = 0.5$, and the simulation allocation rule is chosen to be $M_k = \lceil 1.05 M_{k-1} \rceil$ with $M_0 = 50$. For SAN, we have used the parameters suggested in [49]: initial temperature $T = 50{,}000$, temperature reduction factor $r_T = 0.85$, number of simulation observations $L = 100$, the search neighborhood of a policy π is taken to be $\mathcal{N}(\pi) = \{\pi' \in \Pi : \max_x |\pi_t(x) - \pi'_t(x)| \leq 1,\ \forall t = 0, \ldots, H-1\}$, and the initial policy is randomly selected (uniform distribution) from the policy space Π.

For each case, we performed 30 independent simulation runs of both algorithms, and their average performance is shown in Fig. 4.8, where we plotted the value function estimates of the current best sampled policies given the number of simulations used. We see that MRAS_2 outperforms SAN and yields smaller variance than SAN does. In particular, MRAS_2 shows a reduction in the standard deviation over SAN by a factor of 3 and 2 for the respective cases $p = 1$ and $p = 10$.

4.3.4.2 A Controlled Queueing Example

For the infinite-horizon setting, we consider the one-dimensional queueing example introduced in Sect. 4.6.2.2 with continuous action space, i.e., the service completion probability a can take any value between 0 and 1, and the non-linear one-stage cost function is

$$R(x, a) = x + 5\left[\frac{|X|}{2}\sin(2\pi a) - x\right]^2.$$

The objective is to minimize the expected total discounted cost from a given initial state x, i.e.,

$$\inf_{\pi\in\Pi_s} E\left[\sum_{t=0}^{\infty} \gamma^t R\big(x_t, \pi(x_t)\big)\,\middle|\, x_0 = x\right].$$

We consider two cases: $x_0 = 5$ and $x_0 = 45$.

In MRAS_2, we use the univariate independent normal distribution as the marginal distribution (truncated between 0 and 1). Initially, a mean $\mu_0(x)$ and a variance

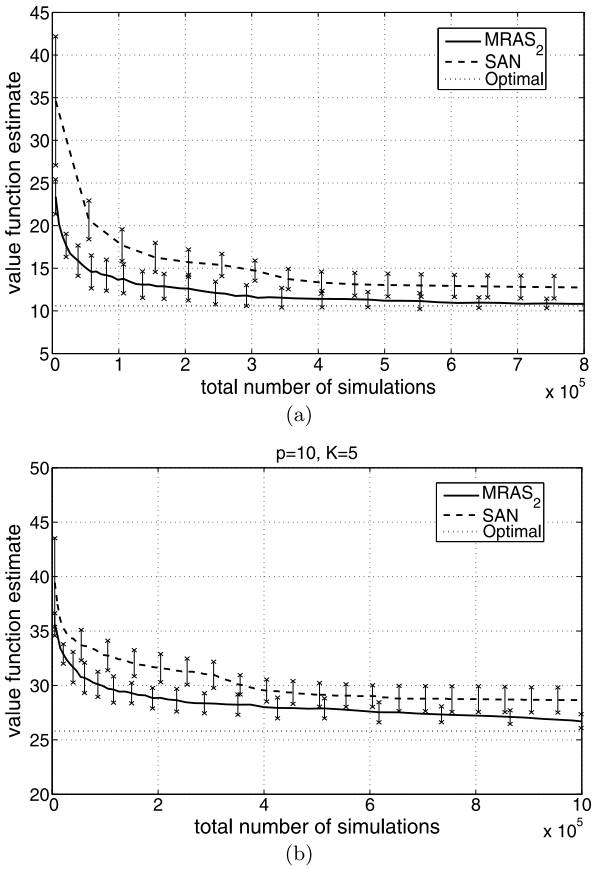


Fig. 4.8 Average performance of MRAS_2 and SAN (mean and standard deviation) on an inventory control problem, based on 30 independent simulation runs, (**a**) $p = 1,\ K = 5$; (**b**) $p = 10,\ K = 5$

$\sigma_0^2(x)$ are specified for all $x \in X$; then at each iteration the new parameters $\mu_{k+1}(x)$ and $\sigma_{k+1}^2(x)$ are updated iteratively in (4.44) as

$$\mu_{k+1}(x) = \frac{\sum_{\pi\in\Lambda_k} \frac{[\mathcal{H}(\bar{V}_k^\pi)]^k}{\mathbf{f}(\pi;\mu_k,\sigma_k^2)} \tilde{I}(\bar{V}_k^\pi, \bar{\chi}_k)\pi(x)}{\sum_{\pi\in\Lambda_k} \frac{[\mathcal{H}(\bar{V}_k^\pi)]^k}{\mathbf{f}(\pi;\mu_k,\sigma_k^2)} \tilde{I}(\bar{V}_k^\pi, \bar{\chi}_k)},$$

$$\sigma_{k+1}^2(x) = \frac{\sum_{\pi\in\Lambda_k} \frac{[\mathcal{H}(\bar{V}_k^\pi)]^k}{\mathbf{f}(\pi;\mu_k,\sigma_k^2)} \tilde{I}(\bar{V}_k^\pi, \bar{\chi}_k)[\pi(x) - \mu_{k+1}(x)]^2}{\sum_{\pi\in\Lambda_k} \frac{[\mathcal{H}(\bar{V}_k^\pi)]^k}{\mathbf{f}(\pi;\mu_k,\sigma_k^2)} \tilde{I}(\bar{V}_k^\pi, \bar{\chi}_k)}, \quad \forall x \in X,$$

where $\Lambda_k = \{\pi^i,\ i = 1, \ldots, N_k\}$ is the set of N_k stationary policies independently generated according to the distribution $\mathbf{f}(\cdot; \mu_k, \sigma_k^2) = (1-\lambda) f(\cdot; \mu_k, \sigma_k^2) + \lambda f(\cdot; \mu_0, \sigma_0^2)$, and $f(\pi; \mu_k, \sigma_k^2) = \prod_{x\in X} f(\pi(x); \mu_k(x), \sigma_k^2(x))$ is the (truncated) joint normal density. Again, if there exists a unique optimal policy, a simple inter-

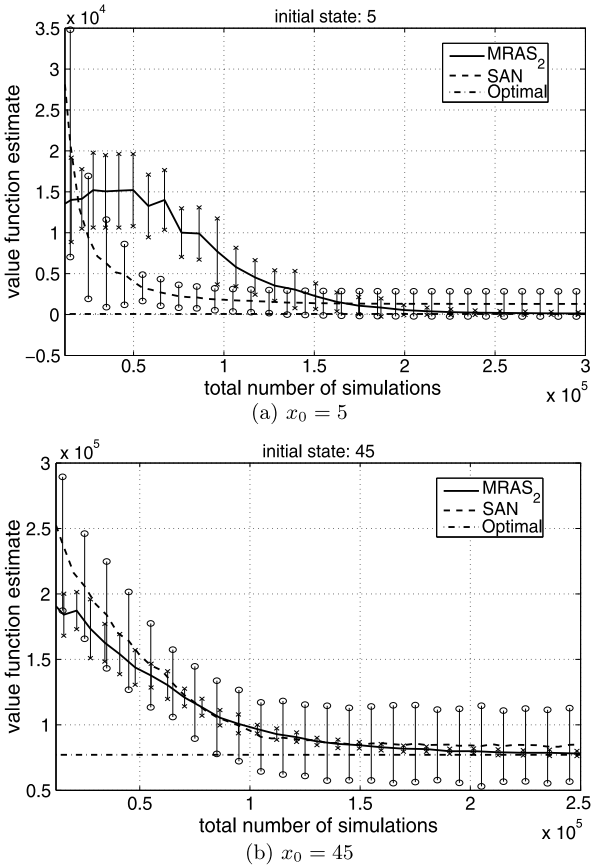


Fig. 4.9 Average performance of MRAS_2 and SAN (mean and standard deviation) on a controlled queueing problem, based on 30 independent simulation runs, (**a**) $x_0 = 5$; (**b**) $x_0 = 45$

pretation of Theorem 4.17 gives $\lim_{k\to\infty} \mu_k(x) = \pi^*(x)$ and $\lim_{k\to\infty} \sigma_k^2(x) = 0$ for all $x \in X$.

The following parameter settings are used in MRAS_2: $\varepsilon = 0.1$, initial sample size $N_0 = 100$, $\rho_0 = 0.1$, $\lambda = 0.01$, $\alpha = 1.04$, $\kappa = 0.01$, smoothing parameter $\upsilon = 0.5$, $M_k = \lceil 1.05 M_{k-1} \rceil$ with $M_0 = 50$. The initial mean $\mu_0(x)$ is randomly selected from $[0, 1]$ according to the uniform distribution for all $x \in X$, and $\sigma_0^2(x) = 1$ for all $x \in X$. For SAN: initial temperature $T = 50{,}000$, temperature reduction factor $r_T = 0.85$, number of simulation observations $L = 100$, the search neighborhood of a (stationary) policy π is taken to be $\mathcal{N}(\pi) = \{\pi' \in \Pi_s : \max_x |\pi(x) - \pi'(x)| \leq 0.02\}$, and the initial policy is uniformly selected from the policy space Π. In both algorithms, each generated/sampled policy is simulated for 500 periods so that the truncation error of the cumulative rewards on any sample path is less than 10.

Figure 4.9 shows the performances of MRAS_2 and SAN for both test cases. For the $x_0 = 5$ case, since SAN uses local search, it may quickly locate a good policy. Thus, the algorithm shows a relatively faster initial convergence rate. However, we see that the algorithm may frequently get trapped at some local optimal solutions.

MRAS_2 outperforms SAN after about 170,000 simulations, and converges to the global optimum in all 30 simulation runs. For the $x_0 = 45$ case, MRAS_2 slightly outperforms SAN and converges to the global optimum in all runs. SAN does not always converge to the global optimum within the allowed total number of simulations. Note that, as in the finite-horizon case, one salient feature of MRAS_2 is that it shows a significant variance reduction over SAN. We see that, in both cases, the reduction in the standard deviation of MRAS_2 over SAN is a factor of more than 10.

4.3.4.3 An Inventory Control Problem with Continuous Demand

To further illustrate the algorithm, we consider an inventory control problem with i.i.d. exponentially distributed continuous demands. At the beginning of period t, the inventory level x_t is reviewed, then an order in the amount of $a_t \geq 0$ is placed and received immediately, and a demand D_t is realized. We assume that unsatisfied demands are fully backlogged. Thus, the inventory level $\{x_t\}$ evolves according to the following dynamics:

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

For a given ordering policy π, we define the H-period average cost

$$\bar{V}_H^\pi := \frac{1}{H} \sum_{i=0}^{H-1} \left[I\{\pi(x_i) > 0\} \big(K + c\pi(x_i)\big) + h x_{i+1}^+ + p x_{i+1}^- \right],$$

where p is the per period per unit penalty cost, h is the per period per unit inventory holding cost, c is the per unit ordering cost, and K is the set-up cost. The goal is to minimize, over the set of all policies Π, the long-run average cost per period, i.e.,

$$\inf_{\pi \in \Pi} \bar{V}^\pi := \inf_{\pi \in \Pi} \liminf_{H \to \infty} \bar{V}_H^\pi.$$

Note that the above cost function is convex, but we will not exploit this property in MRAS_2; however, we do take advantage of the fact that the optimal policy is of (s, S) threshold form, i.e., an order is placed if the inventory level falls below the level s, and the amount of the order is the difference between S and the current inventory level. Therefore, in MRAS_2, the problem becomes searching for the best policy (s^*, S^*) over the set of all admissible (s, S) policies to minimize the average cost per period.

The following two test cases, taken from [66], are used to test the performance of MRAS_2: (i) $p = 10$, $K = 100$; (ii) $p = 10$, $K = 10{,}000$. In both cases, we take $c = h = 1$ and mean demand $E[D] = 200$.

In our experiments with MRAS_2, we take normal distributions as the marginal distributions, and use the same set of parameters as in the previous section, except that the initial mean is now uniformly selected from $[0, 2000] \times [0, 4000]$ and the initial variance is taken to be 10^6. For SAN, $T = 50{,}000$, $r_T = 0.85$, $L = 100$, the neighborhood of a (s, S) policy is chosen to be $N((s, S)) = \{(s', S') :$

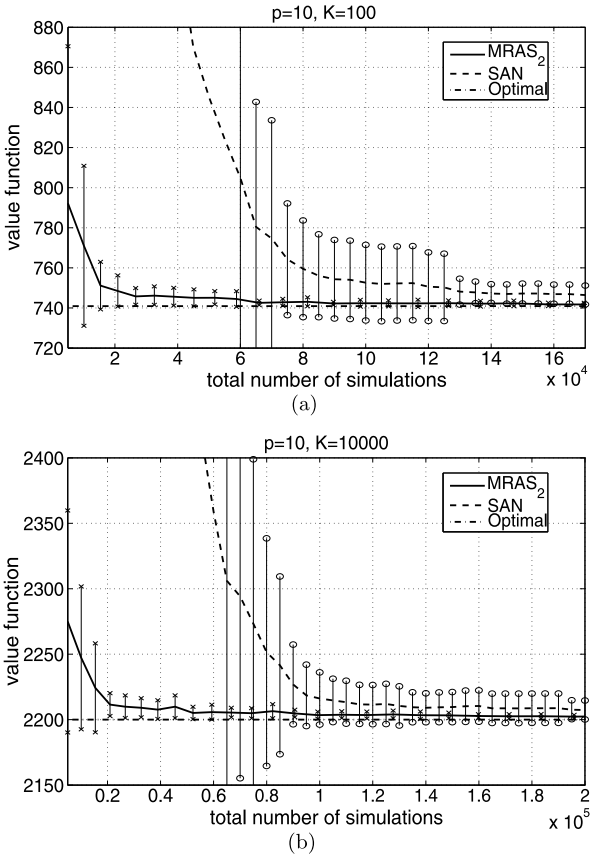


Fig. 4.10 Average performance of MRAS_2 and SAN (mean and standard deviation) on an inventory control problem, based on 30 independent simulation runs, (**a**) $p = 10,\ K = 100$; (**b**) $p = 10,\ K = 10{,}000$

$|s' - s| \leq 50,\ |S' - S| \leq 50\}$, and the initial (s, S) policy is uniformly selected from $[0, 2000] \times [0, 4000]$. In simulation, the average cost per period is estimated by averaging the accumulated cost over 50 periods after a warm-up length of 50 periods.

We performed 30 independent simulation runs for each test case. The average performance of both algorithms are given in Fig. 4.10, where we plotted the average value functions of the current policies found by both algorithms given the total number of simulations used. We see that in both cases, MRAS_2 consistently outperforms SAN and yields much smaller variance than SAN does.

4.4 Application of MRAS to Infinite-Horizon MDPs in Population-Based Evolutionary Approaches

In Sect. 3.2, the construction of sub-MDPs in ERPS is based on information obtained from local search and random sampling from a pre-specified fixed action selection distribution $\mathcal{P}$. The idea is to use local search to fine tune the solution, and use $\mathcal{P}$ to keep the search at a global level in order to avoid local optima. In ERPS,

the balance between these two types of search is maintained by an extra parameter called the exploitation probability. However, the choice of the exploitation probability is clearly problem dependent, and how to choose the most appropriate value of this parameter for a given problem is an open issue. One possible way to get around this, and to further improve the performance of ERPS, is to adaptively update the underlying action selection distribution by using the past sampling information, so that more promising actions will have larger probabilities of being sampled in the future. In this section, we use MRAS as a specific mechanism for updating the action selection distribution, and by combining it with the PIRS step, propose an algorithm with balanced explorative and exploitative search.

4.4.1 Algorithm Description

The algorithm, called evolutionary random policy search with adaptive action selection (ERPS-AAS) (see Fig. 4.11), is an extension of the ERPS algorithm introduced in Sect. 3.2. Basically, we replace the original action selection distribution $\mathcal{P}$ in ERPS by a set of parameterized distributions, which are updated iteratively after evaluating the performance of the population of policies at each iteration. These updated distributions are then used to generate the next population of policies, from which a new sub-MDP is constructed. Note that the PIRS step is still retained in the algorithm to preserve the monotonicity through an elite policy.

During the initialization of ERPS-AAS, the action selection distributions are specified as a set of state-dependent parameterized distributions, i.e., we assign for each state $x \in X$ a different distribution $f(\cdot, \theta^x)$. Then an initial population of policies $\Lambda_0 = \{\pi^1, \ldots, \pi^n\}$ is constructed by sampling an action $\pi^i(x)$ from $f(\cdot, \theta_0^x)$ for all $x \in X$, $i = 1, \ldots, n$. As in ERPS, the initial sub-MDP $\mathcal{G}_{\Lambda_0}$ is thus obtained by restricting the original MDP to the subset of actions $\Gamma_0(x) = \{\pi^1(x), \ldots, \pi^n(x)\}\ \forall x \in X$.

In updating the action selection distributions (i.e., Step 1 in Exploration), the number of samples used to calculate the new parameter is equal to the population size, which is fixed throughout the algorithm. Since there is no performance selection involved in parameter updating, Eq. (4.45) is essentially a sample average approximation of Eq. (4.4) in the MRAS_0 algorithm with the selection parameter $\rho = 1$. The reason for choosing a simple stochastic counterpart of MRAS_0 as opposed to the MRAS_1 algorithm is mainly of practical concern, since a straightforward application of MRAS_1 would require the population size to increase, which makes the policy evaluation step very expensive to compute.

In ERPS-AAS, no explicit local search is used in constructing the next population of policies. Roughly speaking, from our experience with MRAS, what happens is that the action selection distributions will be more "peaked" around those elite actions (i.e., actions corresponding to an elite policy), so their neighboring actions will have larger probabilities of being sampled; whereas those actions farther away will be sampled less frequently. Therefore, the balance between local and global searches is automatically maintained by sampling from the action selection distributions.

ERPS with Adaptive Action Selection (ERPS-AAS)

Input: MDP $(X, A, A(\cdot), P, R)$, population size $n > 1$, strictly increasing function $\mathcal{H} : \Re \to \Re^+$, family of distributions $\{f(\cdot, \theta^x), x \in X\}$ with θ_0^x s.t. the initial action selection distribution $f(a, \theta_0^x) > 0 \; \forall a \in A(x) \; \forall x \in X$.

Initialization: Set iteration count $k = 0$.
Construct an initial population of policies $\Lambda_0 = \{\pi^1, \ldots, \pi^n\}$ by sampling from $f(\cdot, \theta_0^x)$. Construct the initial sub-MDP as $\mathcal{G}_{\Lambda_0} := (X, \Gamma_0, \Gamma_0(\cdot), P, R)$, where $\Gamma_0(x) = \{\pi^1(x), \ldots, \pi^n(x)\}$ and $\Gamma_0 = \bigcup_x \Gamma_0(x)$.

Loop until a specified stopping rule is satisfied:

- **Elite Policy** via policy improvement with reward swapping (PIRS):
  1. Obtain the value function V^π for each $\pi \in \Lambda_k$.
  2. Generate an elite policy of Λ_k using sub-MDP $\mathcal{G}_{\Lambda_k}$:

$$\hat{\pi}^k(x) \in \arg\max_{u \in \Gamma_k(x)} \left\{ R(x,u) + \gamma \sum_{y \in X} P(x,u)(y) \left[\max_{\pi \in \Lambda_k} V^\pi(y) \right] \right\}, \quad x \in X.$$

- **Exploration** via adaptive action selection (AAS) sampling:
  1. Update parameter in action selection distribution:

$$\theta_{k+1}^x \in \arg\max_{\theta^x \in \Theta} \frac{1}{n} \sum_{\pi \in \Lambda_k} \frac{[\mathcal{H}(V^\pi(x))]^k}{f(\pi(x), \theta_k^x)} \ln f\big(\pi(x), \theta^x\big), \quad \forall x \in X. \tag{4.45}$$

  2. Generate $n-1$ policies $\tilde{\pi}^i$, $i = 1, \ldots, n-1$, by sampling actions

$$\tilde{\pi}^i(x) \sim f(\cdot, \theta_{k+1}^x) \quad \forall x \in X.$$

- **Next Population Generation:** $\Lambda_{k+1} = \{\hat{\pi}^k, \tilde{\pi}^1, \ldots, \tilde{\pi}^{n-1}\}$.
- Next sub-MDP: $\mathcal{G}_{\Lambda_{k+1}} := (X, \Gamma_{k+1}, \Gamma_{k+1}(\cdot), P, R)$,
  where $\Gamma_{k+1}(x) = \{\hat{\pi}^k(x), \tilde{\pi}^1(x), \ldots, \tilde{\pi}^{n-1}(x)\}$, $\Gamma_{k+1} = \bigcup_x \Gamma_{k+1}(x)$.
- $k \leftarrow k + 1$.

Output: $\hat{\pi}^k$ an estimated optimal policy.

Fig. 4.11 Evolutionary random policy search with adaptive action selection

Note that throughout the algorithm, both the parameter updating and the action sampling are carried out independently across the states. This is equivalent to using independent univariate distributions in MRAS_0.

4.4.2 Numerical Examples

Again, we consider the infinite-horizon queueing example of Sect. 4.6.2.2 with continuous action space and the one-stage cost function

$$R(x,a) = x + 5\left[\frac{|X|}{2}\sin(2\pi a) - x\right]^2,$$

and the objective is to minimize the expected total discounted cost:

$$\min_{\pi \in \Pi_s} E\left[\sum_{t=0}^{\infty} \gamma^t R\big(x_t, \pi(x_t)\big)\right].$$

In ERPS-AAS, we choose the population size $n = 25$ and the performance function $\mathcal{H}(z) := e^{-z}$ for $z \in \Re$. The action selection distribution at state $x \in X$ is specified as a truncated (between 0 and 1) univariate normal distribution $N(\mu(x), \sigma^2(x))$, with the initial mean $\mu_0(x)$ randomly selected according to the uniform distribution $U(0, 1)$ and the initial variance $\sigma_0^2(x) = 1$ for all $x \in X$. Thus, parameters are updated iteratively in (4.45) as

$$\mu_{k+1}(x) = \frac{\sum_{\pi \in \Lambda_k} \frac{[\mathcal{H}(V^\pi(x))]^k}{f(\pi(x); \mu_k(x), \sigma_k^2(x))} \pi(x)}{\sum_{\pi \in \Lambda_k} \frac{[\mathcal{H}(V^\pi(x))]^k}{f(\pi(x); \mu_k(x), \sigma_k^2(x))}},$$

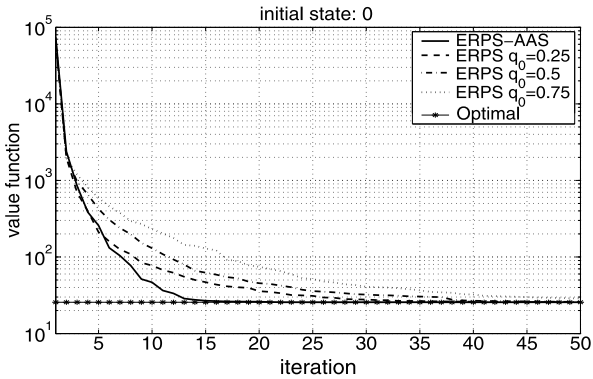
$$\sigma_{k+1}^2(x) = \frac{\sum_{\pi \in \Lambda_k} \frac{[\mathcal{H}(V^\pi(x))]^k}{f(\pi(x); \mu_k(x), \sigma_k^2(x))} (\pi(x) - \mu_{k+1}(x))^2}{\sum_{\pi \in \Lambda_k} \frac{[\mathcal{H}(V^\pi(x))]^k}{f(\pi(x); \mu_k(x), \sigma_k^2(x))}} \quad \forall x \in X,$$

where $f(\pi(x); \mu_k(x), \sigma_k^2(x))$ indicates the probability that action $\pi(x)$ is generated from the truncated normal distribution $N(\mu_k(x), \sigma_k^2(x))$, i.e., the action selection distribution.

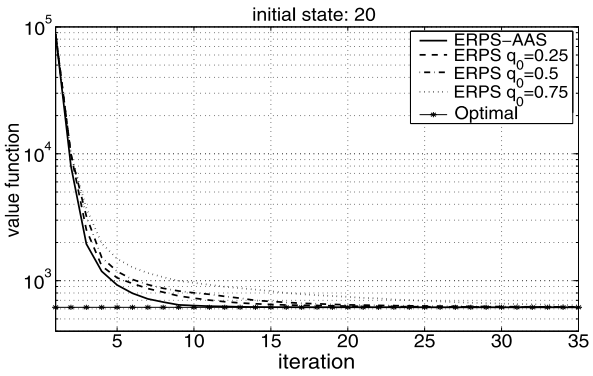
Figure 4.12 shows the convergence of the value functions at the initial states $x_0 = 0$ and $x_0 = 20$ (averaged over 30 independent replication runs) versus the number of algorithm iterations, where we have included for comparison the performance of the original ERPS algorithm under the following parameter setting: population size $n = 25$, search range $r = 10^{-4}$, and exploitation probability $q_0 = 0.25,\ 0.5$, and 0.75. We see that at the minor expense of only an extra parameter updating step (4.45), ERPS-AAS provides even better performance than the original ERPS algorithm, as indicated by the superior convergence rate of the algorithm over ERPS in both cases $x_0 = 0$ and $x_0 = 20$. To illustrate how the algorithm performs at different initial states, we also plotted in Fig. 4.13 the typical convergence behavior of the value functions of the generated elite policies for both algorithms, which clearly shows the uniform superior convergence rate of ERPS-AAS over ERPS across the states.

4.5 Application of MRAS to Finite-Horizon MDPs Using Adaptive Sampling

For finite-horizon MDPs in which both the state space and the action space are huge (e.g., uncountable), we now outline an approach that incorporates the sampling approach of MRAS for the action space into the adaptive multi-stage sampling simulation-based framework of Chap. 2, as summarized in Fig. 2.1. MRAS



(a) $x_0 = 0$



(b) $x_0 = 20$

Fig. 4.12 Average performance of ERPS-AAS and the original ERPS algorithm (mean based on 30 independent replication runs), (**a**) $x_0 = 0$; (**b**) $x_0 = 20$

provides yet another alternative adaptive sampling mechanism in the approach of Chap. 2, replacing the upper confidence bound (UCB) approach from multi-armed bandit models and pursuit learning automata (PLA) approach for deciding on the next action to sample. As in Chap. 2, the resulting algorithms will be independent of the size of the state space, but now it will also be independent of the size of the action space. However, as in Chap. 2, the computation will increase exponentially with the horizon length, and this will be further exacerbated by the additional requirements of the MRAS algorithm for requiring *multiple* samples of the action space at each iteration. Figure 4.14 outlines the framework for the MRAS approach applied to the adaptive multi-stage sampling setting.

The algorithm uses all of the samples to update the Q-function estimates, but possibly only a portion to update the sampling parameter vector θ. Note the use of common random numbers by using the same w_k over all actions in an iteration.

The main steps in each iteration (cf. **Loop** in Fig. 4.14) are the following:

- generation—sampling actions from the probability distribution;
- estimation—updating Q-function estimates for each action sampled;

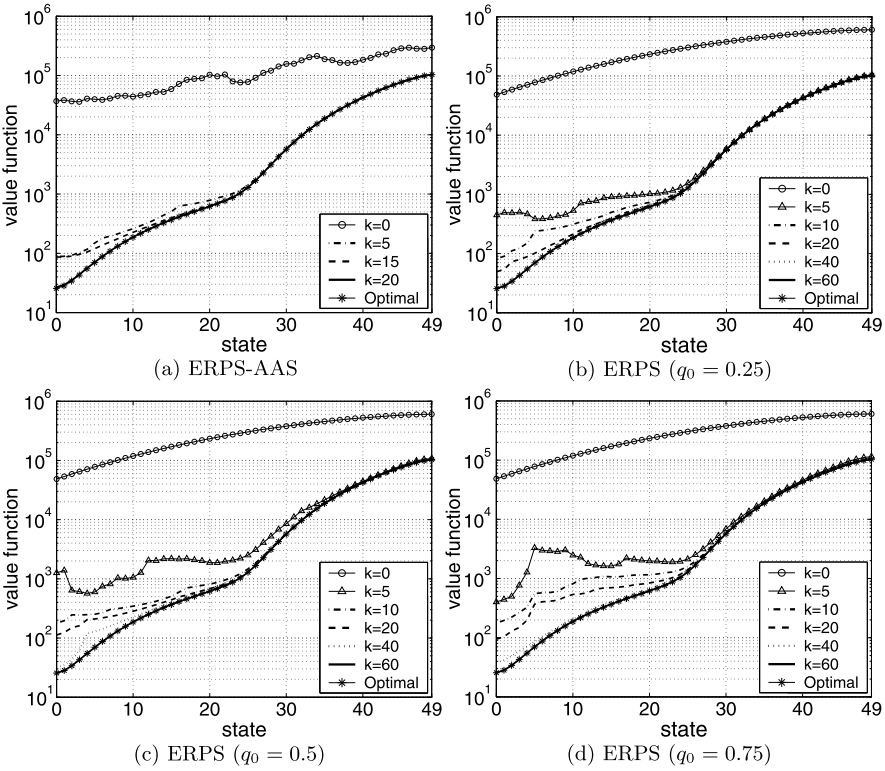


Fig. 4.13 Typical performance of ERPS-AAS and ERPS, (**a**) ERPS-AAS ($n = 25$); (**b**) ERPS ($n = 25$, $r = 10^{-4}$, $q_0 = 0.25$); (**c**) ERPS ($n = 25$, $r = 10^{-4}$, $q_0 = 0.5$); (**d**) ERPS ($n = 25$, $r = 10^{-4}$, $q_0 = 0.75$)

- selection and update—choosing the elite population, which is then used for updating the parameter (vector) in the probability distribution.

The selection step in MRAS is generally carried out using quantile estimation, which would involve the Q-function estimates in this setting.

To summarize the approach, we reiterate its features here, contrasting them with various concepts and approaches contained in the rest of the book:

- It is *simulation* based, in that it uses simulation to generate the next state in estimating the Q-function (cf. $f(x, a_j(k), w_k)$ in second step in Loop in Fig. 4.14).
- It uses multi-stage *adaptive sampling* as in Chap. 2, in that it uses a sampling mechanism for selecting the action that generates the next state transition via simulation (cf. first step in Loop in Fig. 4.14).
- It is *population*-based as in Chap. 3, in that it iteratively generates a *set* of candidate solutions rather than a single estimate as in Chap. 2; however, as in Chap. 2 and unlike Chap. 3, the algorithm works with actions for a single state at a time

MRAS Multi-Stage Sampling Framework

Input: stage $i < H$, state $x \in X$ (for $i = H$, $\hat{V}_i^{N_i}(x) = \hat{V}_H^{N_H}(x) = 0$),
initial population size n_0, simulation budget $N_i > 0$, $\lambda \in (0, 1]$,
parameterized distribution family $f(\cdot, \theta)$, initial θ_0, strictly increasing function $\mathcal{H} : \Re \to \Re^+$,
simulation allocation rule $\{M_k, k = 0, 1, \ldots\}$.

Initialization: Set $N_a^i(x) = 0$, $C_i(x, a) = 0 \; \forall a \in A(x)$; $k = 0$.

Loop until $\sum_{r=0}^{k} n_r M_r \geq N_i$:

- Sample $a_j(k) \sim (1 - \lambda) f(\cdot, \theta_k) + \lambda f(\cdot, \theta_0)$, $j = 1, \ldots, n_k$,
  $w_k^l \sim U(0, 1)$, $l = 1, \ldots, M_k$.
- Update Q-function estimate for each $a_j(k)$, $j = 1, \ldots, n_k$:

$$C_i\big(x, a_j(k)\big) \leftarrow C_i\big(x, a_j(k)\big) + \sum_{l=1}^{M_k} \big[R'\big(x, a_j(k), w_k^l\big) + \hat{V}_{i+1}^{N_{i+1}}\big(f\big(x, a_j(k), w_k^l\big)\big)\big],$$

$$N_{a_j(k)}^i(x) \leftarrow N_{a_j(k)}^i(x) + M_k,$$

$$\hat{Q}_i^{N_i}\big(x, a_j(k)\big) \leftarrow \frac{C_i(x, a_j(k))}{N_{a_j(k)}^i(x)}.$$

- Update population size n_{k+1} (non-decreasing) and probability distribution parameter (vector):

$$\theta_{k+1} \in \arg\max_{\theta \in \Theta} \frac{1}{n_k} \sum_{j \in \Lambda_k^*} \frac{[\mathcal{H}(\hat{Q}_i^{N_i}(x, a_j(k)))]^k}{f(a_j(k), \theta_k)} \ln f\big(a_j(k), \theta\big), \quad \forall i = 1, \ldots, |X|, \quad (4.46)$$

  where Λ_k^* is the elite subset of actions selected from $\{a_j(k), j = 1, \ldots, n_k\}$.
- $k \leftarrow k + 1$.

Output: $\hat{V}_i^{N_i}(x)$ some function of $\{\hat{Q}_i^{N_i}(x, a)\}$.

Fig. 4.14 Framework for applying MRAS to multi-stage sampling

(initially given, then recursively generated in time through simulated next stages) rather than with (stationary) policies over the entire state space (cf. n_k in first step in Loop in Fig. 4.14).

- It uses a model-based global optimization algorithm, in that it updates a *probability distribution* over the action space, which is the same approach as the PLA algorithm of Chap. 2. However, unlike the PLA algorithm, it can handle infinite and uncountable action spaces, generates a set of actions in each iteration (as mentioned in the previous item), and employs a compact representation via a *parameterized* distribution rather than updating each action individually, which is impractical for large action spaces (cf. third step in Loop in Fig. 4.14, where the version of (4.10) from MRAS$_2$ used here and given by (4.46) replaces (2.28) in the PLA algorithm).

4.6 A Stochastic Approximation Framework

In this section, we present a stochastic approximation framework to study model-based optimization algorithms. The framework is based on the MRAS method presented in Sect. 4.1. It is intended to combine the robust features of model-based algorithms encountered in practice with rigorous theoretical convergence guarantees. Specifically, we exploit a natural connection between a class of model-based algorithms and the well-known stochastic approximation (SA) method. We show that, regardless of the type of decision variables involved in (4.1), algorithms conforming to the framework can be equivalently formulated in the form of a generalized stochastic approximation procedure on a transformed continuous parameter space for solving a sequence of stochastic optimization problems with differentiable structures. This viewpoint allows us to study the asymptotic properties of these algorithms, both convergence and rate, by using existing theory and tools from SA.

The key idea that leads to the stochastic approximation framework is based on replacing the reference sequence $\{g_k\}$ in the original MRAS method by a more general distribution sequence in the recursive form:

$$\hat{g}_{k+1}(\mathbf{x}) = \alpha_k g_{k+1}(\mathbf{x}) + (1-\alpha_k) f(\mathbf{x}, \theta_k) \quad \forall k, \tag{4.47}$$

where $\alpha_k \in (0, 1]$ is a smoothing parameter, so that $\hat{g}_{k+1}$ is a mixture of the reference distribution g_{k+1} and the sampling distribution $f(\mathbf{x}, \theta_k)$ obtained at the kth iteration. Intuitively, such a mixture retains the properties of g_{k+1} while ensuring that its difference from $f(\mathbf{x}, \theta_k)$ is only incremental, so that the new sampling distribution $f(\mathbf{x}, \theta_{k+1})$ obtained by minimizing $\mathcal{D}(\hat{g}_{k+1}, f(\cdot, \theta))$ will not deviate significantly from the current sampling distribution $f(\mathbf{x}, \theta_k)$. This is especially useful in actual implementation, when g_k and $f(\cdot, \theta_k)$ can only be estimated and calculated based on the set of sampled solutions.

When $\{\hat{g}_{k+1}\}$ is used in (4.2) to minimize the KL-divergence with f, the following lemma states a key link between the two successive mean vector functions of the projected (exponential family) probability distributions.

Lemma 4.20 *If $f(\cdot, \theta)$ belongs to a NEF and the new parameter vector θ_{k+1} obtained via minimizing $\mathcal{D}(\hat{g}_{k+1}, f(\cdot, \theta))$ is an interior point of Θ, i.e., $\theta_{k+1} \in \text{int}(\Theta)$ for all k, then*

$$m(\theta_{k+1}) - m(\theta_k) = -\alpha_k \nabla_\theta \mathcal{D}\big(g_{k+1}, f(\cdot, \theta)\big)\big|_{\theta=\theta_k} \quad \forall k = 0, 1, 2, \ldots, \tag{4.48}$$

where recall that $m(\theta) := E_\theta[\Upsilon(\mathbf{X})]$ and $\Upsilon(\mathbf{x})$ is the sufficient statistic of the NEF.

Proof Since $\theta_{k+1} \in \text{int}(\Theta)$, it satisfies the first order necessary condition for optimality. By using an argument similar to the proof of Lemma 4.4, it can be shown that $m(\theta_{k+1}) = E_{\theta_{k+1}}[\Upsilon(\mathbf{X})] = E_{\hat{g}_{k+1}}[\Upsilon(\mathbf{X})]$. Therefore, by (4.47),

$$m(\theta_{k+1}) = E_{\hat{g}_{k+1}}\big[\Upsilon(\mathbf{X})\big] = \alpha_k E_{g_{k+1}}\big[\Upsilon(\mathbf{X})\big] + (1-\alpha_k) m(\theta_k).$$

Thus, the difference between the two successive mean vector functions can be written as

$$m(\theta_{k+1}) - m(\theta_k) = \alpha_k \big(E_{g_{k+1}}\big[\Upsilon(\mathbf{X})\big] - m(\theta_k) \big) = -\alpha_k \nabla_\theta \mathcal{D}\big(g_{k+1}, f(\cdot,\theta)\big)\big|_{\theta=\theta_k},$$

where the interchange of derivative and expectation in the last step follows from the properties of NEFs and the fact that $\mathcal{X}$ does not depend on θ. □

Lemma 4.20 states that regardless of the specific form of the reference distribution g_k, the mean vector function $m(\theta_k)$ (i.e., a one-to-one transformation of the parameter θ_k) is updated at each step along the gradient descent direction of the *time-varying* objective function for the *minimization* problem $\min_{\theta\in\Theta} \mathcal{D}(g_{k+1}, f(\cdot,\theta))\ \forall k$. Note that the parameter sequence $\{\alpha_k\}$ turns out to be the gain sequence for the gradient iteration, so that the special case $\alpha_k \equiv 1$ corresponds to the original MRAS method. This suggests that all model-based algorithms that fall under the framework can be equivalently viewed as gradient-based recursions on the parameter space Θ for solving a sequence of optimization problems with differentiable structures. This new interpretation of model-based algorithms provides a key insight to understand how these algorithms address hard optimization problems with little structure.

In actual implementation, when expectations are replaced by sample averages based on Monte Carlo sampling, (4.48) becomes a recursive algorithm of stochastic approximation type with direct gradient estimation. Thus, the rich body of tools and results from stochastic approximation can be incorporated into the framework to analyze model-based algorithms.

4.6.1 Model-Based Annealing Random Search

To illustrate the stochastic approximation framework, we present a specific algorithm instantiation called model-based annealing random search (MARS) [92, 93]. MARS can be viewed as an implementable version of the Annealing Adaptive Search (AAS) algorithm, which was originally introduced in [147] as a useful means to understand the behavior of simulated annealing.

In the idealized AAS algorithm, solutions are generated at each iteration k by sampling from a Boltzmann distribution characterized by a temperature parameter T_k. Within the context of problem (4.1), the density/mass function of the Boltzmann distribution can be written as

$$g_k(\mathbf{x}) = \frac{e^{J(\mathbf{x})/T_k}}{\int_{\mathcal{X}} e^{J(\mathbf{x})/T_k} \nu(d\mathbf{x})}. \tag{4.49}$$

It is well-known that as the temperature T_k decreases to some small constant $T^* \geq 0$, the sequence $\{g_k\}$ will converge to a limiting density/mass function g^* that concentrates its mass around the optimal solution $\mathbf{x}^*$. Consequently, as the sampling

Algorithm MARS$_0$—Idealized Version

Input: annealing schedule $\{T_k\}$, gain sequence $\{\alpha_k\}$, initial parameterized density/mass function $f(\mathbf{x}, \theta_0)$, with θ_0 s.t. $f(\mathbf{x}, \theta_0) > 0 \; \forall \mathbf{x} \in \mathcal{X}$.

Initialization: Set iteration count $k = 0$.

Loop until Stopping Rule is satisfied:

- Update parameter vector:

$$\theta_{k+1} \in \arg\min_{\theta \in \Theta} \mathcal{D}\big(\hat{g}_{k+1}, f(\cdot, \theta)\big), \tag{4.50}$$

  where $\hat{g}_{k+1} = \alpha_k g_{k+1}(\mathbf{x}) + (1 - \alpha_k) f(\mathbf{x}, \theta_k)$ and g_{k+1} is given by (4.49).
- $k \leftarrow k + 1$.

Output: θ_k.

Fig. 4.15 Description of MARS$_0$ algorithm

process proceeds, solutions generated from Boltzmann distributions with small T_k values will be close to the global optimum with high probability. Unfortunately, the algorithm is not readily implementable for solving optimization problems, because the practical problem of sampling exactly from the Boltzmann distribution g_k is known to be extremely difficult. Prior work to address this issue has primarily focused on embedding Markov chain Monte Carlo sampling techniques within the AAS algorithm (e.g., [190]).

The MARS algorithm provides an alternative approach to address the implementation difficulty of AAS. The basic idea is to use a sequence of NEF distributions to approximate the target Boltzmann distributions and then use the sequence as surrogate distributions to generate candidate points. Thus, by treating Boltzmann distributions as reference distributions, candidate solutions are drawn at each iteration of MARS *indirectly* from a Boltzmann distribution by sampling exactly from its approximation. This is in contrast to Markov chain-based techniques [190] that aim to *directly* sample from the Boltzmann distributions. The idealized MARS algorithm we call MARS$_0$ is stated in Fig. 4.15.

Under the gradient interpretation of MARS$_0$, Lemma 4.20 implies that the mean vector function $m(\theta_{k+1})$ of the new sampling distribution $f(\cdot, \theta_{k+1})$ obtained in (4.50) can be viewed as an iterate generated by a gradient descent algorithm for solving the iteration-varying stochastic minimization problem $\min_{\theta \in \Theta} \mathcal{D}(g_{k+1}, f(\cdot, \theta))$ on the transformed parameter space Θ. The solution to this problem, as k goes to infinity, is an optimal parameter θ^* that provides the best possible approximation to the limiting Boltzmann distribution g^*.

Note that to implement MARS$_0$ would still require the full information about the Boltzmann distribution g_{k+1}, which is generally unavailable unless the entire solution space $\mathcal{X}$ can be enumerated. Therefore, a rational approach in practice is to use only a finite number of samples generated at each iteration k to construct an

Algorithm MARS$_1$—Implementable Version

Input: annealing schedule $\{T_k\}$, gain sequence $\{\alpha_k\}$, sample size $\{N_k\}$, exploration parameter sequence $\{\lambda_k\}$, initial parameterized density/mass function $f(\mathbf{x}, \hat{\theta}_0)$, with $\hat{\theta}_0$ s.t. $f(\mathbf{x}, \hat{\theta}_0) > 0 \; \forall \mathbf{x} \in \mathcal{X}$.

Initialization: Set iteration count $k = 0$.

Loop until Stopping Rule is satisfied:

- Generate a population of N_k i.i.d. solutions $\Lambda_k = \{\mathbf{X}_k^1, \ldots, \mathbf{X}_k^{N_k}\}$ from $\hat{f}(\mathbf{x}, \hat{\theta}_k) := (1 - \lambda_k) f(\mathbf{x}, \hat{\theta}_k) + \lambda_k f(\mathbf{x}, \hat{\theta}_0)$.
- Update parameter vector:

$$\hat{\theta}_{k+1} \in \arg\min_{\theta \in \Theta} \mathcal{D}\big(\hat{g}_{k+1}, f(\cdot, \theta)\big), \tag{4.51}$$

  where $\hat{g}_{k+1}$ is given by (4.52).
- $k \leftarrow k + 1$.

Output: $\hat{\theta}_k$.

Fig. 4.16 Description of MARS$_1$ algorithm

empirical distribution $\bar{g}_{k+1}$, and then use $\bar{g}_{k+1}$ to approximate g_{k+1}. This results in the implementable version of MARS$_0$ presented in Fig. 4.16.

In addition to $\{T_k\}$ and $\{\alpha_k\}$, the MARS$_1$ algorithm requires specifications of two parameter sequences $\{N_k\}$ and $\{\lambda_k\}$, where N_k specifies the number of candidate solutions to be generated at each iteration, and the exploration parameter λ_k allows the algorithm to explore the entire feasible region by occasionally sampling from the initial distribution $f(\mathbf{x}, \hat{\theta}_0)$, so that there is a positive probability for the algorithm to reach anywhere in $\mathcal{X}$ at each iteration. In (4.51), the KL divergence is with respect to $\hat{g}_{k+1}$, an estimate of $\hat{g}_{k+1}$ in (4.50) of Algorithm MARS$_0$ based on the sampled solutions in Λ_k, i.e.,

$$\hat{g}_{k+1}(\mathbf{x}) = \alpha_k \sum_{y \in \Lambda_k} \bar{g}_{k+1}(\mathbf{x}) \delta(\mathbf{x} - \mathbf{y}) + (1 - \alpha_k) f(\mathbf{x}, \hat{\theta}_k), \quad \mathbf{x} \in \mathcal{X}, \tag{4.52}$$

where δ is the Dirac delta function and we have replaced the Boltzmann distribution g_{k+1} in (4.49) by a discrete empirical distribution

$$\bar{g}_{k+1}(\mathbf{x}) := \frac{e^{\frac{J(\mathbf{x})}{T_{k+1}}} / \hat{f}(\mathbf{x}, \hat{\theta}_k)}{\sum_{\mathbf{y} \in \Lambda_k} e^{\frac{J(\mathbf{y})}{T_{k+1}}} / \hat{f}(\mathbf{y}, \hat{\theta}_k)} \quad \forall \mathbf{x} \in \Lambda_k. \tag{4.53}$$

Intuitively, the division by $\hat{f}(\mathbf{x}, \hat{\theta}_k)$ in $\bar{g}_{k+1}$ is used to compensate for solutions that are unlikely to be chosen, which makes $\bar{g}_{k+1}$ a good approximation of the Boltzmann distribution g_{k+1}.

We assume that the new parameter obtained in (4.51) of Algorithm MARS$_1$ satisfies the following condition:

Assumption C1 The parameter $\hat{\theta}_{k+1}$ computed in (4.51) of Algorithm MARS$_1$ satisfies $\hat{\theta}_{k+1} \in \text{int}(\Theta)$ for all k.

Similar to Lemma 4.20, the following result shows the connection between the successive mean vector functions obtained in Algorithm MARS$_1$. The proof is similar to that of Lemma 4.20 and is thus omitted.

Lemma 4.21 *If Assumption C1 holds, then the mean vector function $m(\hat{\theta}_{k+1})$ of $f(\mathbf{x}, \hat{\theta}_{k+1})$ satisfies*

$$m(\hat{\theta}_{k+1}) - m(\hat{\theta}_k) = -\alpha_k \big(m(\hat{\theta}_k) - E_{\tilde{g}_{k+1}}\big[\Upsilon(\mathbf{X})\big] \big) \quad \forall k = 0, 1, 2, \ldots. \tag{4.54}$$

Finally, to recapitulate the connection of Algorithm MARS$_1$ to stochastic gradient search, we rewrite (4.54) as follows:

$$\begin{aligned}
& m(\hat{\theta}_{k+1}) - m(\hat{\theta}_k) \\
&= -\alpha_k \big(m(\hat{\theta}_k) - E_{g_{k+1}}\big[\Upsilon(\mathbf{X})\big] + E_{g_{k+1}}\big[\Upsilon(\mathbf{X})\big] - E_{\tilde{g}_{k+1}}\big[\Upsilon(\mathbf{X})\big] \big), \\
&= -\alpha_k \nabla_\theta \mathcal{D}\big(g_{k+1}, f(\cdot, \theta)\big)\big|_{\theta=\hat{\theta}_k} \\
&\quad - \alpha_k \left(\frac{\int_{\mathcal{X}} e^{\frac{J(\mathbf{x})}{T_{k+1}}} \Upsilon(\mathbf{x}) \nu(d\mathbf{x})}{\int_{\mathcal{X}} e^{\frac{J(\mathbf{x})}{T_{k+1}}} \nu(d\mathbf{x})} - \frac{\frac{1}{N_k} \sum_{\mathbf{x} \in \Lambda_k} e^{\frac{J(\mathbf{x})}{T_{k+1}}} \Upsilon(\mathbf{x}) / \hat{f}(\mathbf{x}, \hat{\theta}_k)}{\frac{1}{N_k} \sum_{\mathbf{x} \in \Lambda_k} e^{\frac{J(\mathbf{x})}{T_{k+1}}} / \hat{f}(\mathbf{x}, \hat{\theta}_k)} \right).
\end{aligned} \tag{4.55}$$

This becomes a Robbins–Monro type stochastic approximation algorithm in terms of the true gradient of $\mathcal{D}(g_{k+1}, f(\cdot, \theta))$ with respect to θ and an error term due to the combined effect of bias and noise caused by Monte-Carlo random sampling in MARS$_1$.

4.6.1.1 Global Convergence of MARS$_1$

Note that (4.55) generalizes a typical stochastic approximation recursion in that the function $\mathcal{D}(g_{k+1}, f(\cdot, \theta))$ may change shape with k. Such a generalization has been studied in [56] under a scalar setting and later extended by [168] to a multivariate setting in the context of non-linear adaptive control. The basic idea is that while the underlying objective function varies with k, the optimal solutions to the sequence of time-varying optimization problems will reach a limit as k goes to infinity. This desired property is justified in our setting, because the idealized sequence of Boltzmann distributions $\{g_k\}$ converges to a limiting distribution g^* as k goes to infinity. This will in turn imply the convergence of the sequence of the (idealized) optimal solutions $\{\theta_k\}$ to a global optimizer θ^*.

Since the MARS$_1$ algorithm is randomized, it induces a probability distribution over the set of all sequences of sampled solutions. We denote by $P(\cdot)$ and $E[\cdot]$ the probability and expectation taken with respect to this distribution, and denote by $I\{A\}$ the indicator function of set A. In the rest of the section, probability one convergence of random vectors and matrices is to be understood with respect to P. We define $\mathcal{F}_k = \sigma\{\Lambda_0, \Lambda_1, \ldots, \Lambda_{k-1}\}$, $k = 1, 2, \ldots$ as the sequence of increasing σ-fields generated by the set of all sampled solutions up to iteration $k-1$. Note that given $\mathcal{F}_k$, the parameter $\hat{\theta}_k$ is completely determined and the set of solutions Λ_k generated at the kth iteration is conditionally independent of the past. We use $\hat{P}_{\hat{\theta}_k}(\cdot|\mathcal{F}_k)$ and $\hat{E}_{\hat{\theta}_k}[\cdot|\mathcal{F}_k]$ to denote the conditional probability and expectation with respect to the mixture distribution $\hat{f}(\cdot, \hat{\theta}_k)$. The following shorthand notations will also be frequently used:

$$\begin{aligned}
\mathbf{U}_k &= \frac{1}{N_k} \sum_{\mathbf{x} \in \Lambda_k} e^{\frac{J(\mathbf{x})}{T_{k+1}}} \Upsilon(\mathbf{x}) / \hat{f}(\mathbf{x}, \hat{\theta}_k), \\
\bar{\mathbf{U}}_k &= \hat{E}_{\hat{\theta}_k}[\mathbf{U}_k|\mathcal{F}_k] = \int_{\mathcal{X}} e^{\frac{J(\mathbf{x})}{T_{k+1}}} \Upsilon(\mathbf{x}) \nu(d\mathbf{x}) \\
\mathbf{V}_k &= \frac{1}{N_k} \sum_{\mathbf{x} \in \Lambda_k} e^{\frac{J(\mathbf{x})}{T_{k+1}}} / \hat{f}(\mathbf{x}, \hat{\theta}_k), \\
\bar{\mathbf{V}}_k &= \hat{E}_{\hat{\theta}_k}[\mathbf{V}_k|\mathcal{F}_k] = \int_{\mathcal{X}} e^{\frac{J(\mathbf{x})}{T_{k+1}}} \nu(d\mathbf{x}).
\end{aligned} \tag{4.56}$$

To simplify the presentation and analysis of the algorithm, we assume throughout this section that the solution space $\mathcal{X}$ is compact and that $J(\mathbf{x}) > 0$ $\forall \mathbf{x} \in \mathcal{X}$. In addition, we make the following assumptions on the objective function, which are essentially identical to conditions A1 and A2 in Sect. 4.2.

Assumption C2 For any constant $\varepsilon < J(\mathbf{x}^*)$, the set $\{\mathbf{x} \in \mathcal{X} : J(\mathbf{x}) \geq \varepsilon\}$ has a strictly positive Lebesgue or discrete measure.

Assumption C3 For any $\delta > 0$, $\sup_{\mathbf{x} \in A_\delta} J(\mathbf{x}) < J(\mathbf{x}^*)$, where $A_\delta := \{\mathbf{x} \in \mathcal{X} : \|\mathbf{x} - \mathbf{x}^*\| \geq \delta\}$, and we use the convention that the supremum over the empty set is $-\infty$.

To present the main convergence result, we will also need the following conditions on the input parameters of MARS$_1$.

Assumption C4 The mapping $\Upsilon(\mathbf{x})$ given in Definition 4.2 is bounded on $\mathcal{X}$. Moreover, for any $\xi > 0$, there exists $\delta > 0$ such that $\|\Upsilon(\mathbf{x}) - \Upsilon(\mathbf{x}^*)\| \leq \xi$ whenever $\|\mathbf{x} - \mathbf{x}^*\| \leq \delta$.

Assumption C5 The gain sequence $\{\alpha_k\}$ satisfies $\alpha_k > 0$ $\forall k$, $\sum_{k=0}^{\infty} \alpha_k = \infty$, and $\sum_{k=0}^{\infty} \alpha_k^2 < \infty$.

Assumption C6

(a) The sequence $\{T_k\}$ satisfies $T_k > 0\ \forall k$ and $T_k \to T^* \geq 0$ as $k \to \infty$;
(b) The sequence $\{\lambda_k\}$ satisfies $\lambda_k > 0\ \forall k$ and $\lambda_k \to \lambda^* \in [0, 1)$ as $k \to \infty$;
(c) Moreover, $\frac{e^{J^*/T_k}}{N_k \lambda_k} \to 0$ as $k \to \infty$, where $J^* := J(\mathbf{x}^*)$.

Since $\mathcal{X}$ is compact, Assumption C4 is satisfied for natural exponential distributions with continuous mappings Υ, e.g., normal, exponential, and Gamma distribution with fixed shape parameters. When $\mathcal{X}$ is discrete, the assumption also holds trivially for many mass functions encountered in practice, e.g., Bernoulli, binomial, and Poisson. Note that since $\Upsilon(\mathbf{x})$ is bounded, the initial parameterized density/mass function $f(\mathbf{x}, \hat{\theta}_0)$ is bounded away from zero on $\mathcal{X}$ for any given $\hat{\theta}_0 \in \text{int}(\Theta)$, i.e., $f_* := \inf_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x}, \hat{\theta}_0) > 0$. Assumption C5 is a typical SA condition; it requires that the gain α_k should decrease to zero at a rate that is neither too fast nor to slow (see, e.g., [167]). Assumption C6 requires that both $\{T_k\}$ and $\{\lambda_k\}$ should converge to some limits. However, the sequence $\{T_k\}$ is not necessarily monotone. This allows the use of non-monotonic cooling schedules in MARS, including various adaptive schedules, which have been observed to have superior performance than monotone schedules on some problems in both SAN and AAS (e.g., [118, 190]). Assumption C6(c) states that the three parameter sequences $\{T_k\}$, $\{N_k\}$, and $\{\lambda_k\}$ should be chosen in balance. Roughly speaking, the annealing schedule $\{T_k\}$ determines the convergence speed of the sequence of (idealized) Boltzmann distribution $\{g_k\}$ to the limiting distribution g^*, whereas $\{N_k\}$ determines whether $\{g_k\}$ can be closely approximated by the surrogate distributions $\{f(\cdot, \hat{\theta}_k)\}$. Thus, if the temperature T_k decays to zero at a fast rate, then the sample size N_k should also be increased sufficiently fast to ensure that the sequence $\{f(\cdot, \hat{\theta}_k)\}$ can properly "track" the sequence of convergent Boltzmann distributions. We consider the following special cases of Assumption C6(c): (i) If $T^* > 0$ and $\lambda^* > 0$, then for Assumption C6(c) to hold, it is sufficient to let $N_k \to \infty$ as $k \to \infty$. (ii) If $T_k = \frac{T_0}{\ln(1+k)}$, $N_k = \Theta(k^\beta)$, and $\lambda_k = \Omega(k^{-\gamma})$ for some constants $T_0 > 0$, $\beta > 0$, and $\gamma > 0$, then Assumption C6(c) is satisfied for $\beta > \gamma$, provided that T_0 is sufficiently large. (iii) If $T_k = \frac{T_0}{1+ck}$, $N_k = \Theta(\beta^k)$, and $\lambda_k = \Omega(k^{-\gamma})$ for constants $T_0 > 0$, $c > 0$, $\beta > 1$, and $\gamma > 0$, then it is easy to verify that Assumption C6(c) is satisfied by taking $\beta > e^{J^* c/T_0}$.

We need the following intermediate results. First, we establish that for the class of optimization problems characterized by conditions C2 and C3, the sequence of Boltzmann distributions converges (in a weak sense) to a limiting distribution. Define $\Upsilon^* := \Upsilon(\mathbf{x}^*)$ if $T^* = 0$, $\Upsilon^* := E_{g^*}[\Upsilon(\mathbf{X})]$ whenever $T^* > 0$, where g^* is the limiting Boltzmann distribution parameterized by $T^* > 0$.

Lemma 4.22 *If Assumptions* C2, C3, C4, *and* C6(a) *are satisfied, then*

$$E_{g_k}\big[\Upsilon(\mathbf{X})\big] \to \Upsilon^* \quad \text{as } k \to \infty,$$

where the limit is taken component-wise.

Proof The $T^* > 0$ case is trivial and is thus omitted. In the $T^* = 0$ case, we have $\Upsilon^* = \Upsilon(\mathbf{x}^*)$. By Assumption C4, for any $\xi > 0$, we can find a $\delta > 0$ such that $\|\mathbf{x} - \mathbf{x}^*\| < \delta$ implies $\|\Upsilon(\mathbf{x}) - \Upsilon^*\| < \xi$. Define $A_\delta = \{\mathbf{x} \in \mathcal{X} : \|\mathbf{x} - \mathbf{x}^*\| \geq \delta\}$. We have by Assumption C3 that $\bar{J} = \sup_{\mathbf{x} \in A_\delta} J(\mathbf{x}) < J^*$. Take $\varepsilon = \frac{\bar{J} + J^*}{2}$. By Assumption C2, the set $B_\epsilon := \{\mathbf{x} \in \mathcal{X} : J(\mathbf{x}) > \varepsilon\}$ has a positive Lebesgue/discrete measure. Thus,

$$
\begin{aligned}
&\left\| E_{g_k}[\Upsilon(\mathbf{X})] - \Upsilon^* \right\| \\
&\quad \leq E_{g_k}\left[\left\| \Upsilon(\mathbf{X}) - \Upsilon^* \right\| \right] \\
&\quad = \int_{A_\delta^c} \left\| \Upsilon(\mathbf{x}) - \Upsilon^* \right\| g_k(\mathbf{x}) \nu(d\mathbf{x}) + \int_{A_\delta} \left\| \Upsilon(\mathbf{x}) - \Upsilon^* \right\| g_k(\mathbf{x}) \nu(d\mathbf{x}) \\
&\quad \leq \xi + \sup_{\mathbf{x} \in \mathcal{X}} \left\| \Upsilon(\mathbf{x}) - \Upsilon^* \right\| \frac{\int_{A_\delta} e^{\frac{J(\mathbf{x})}{T_k}} \nu(d\mathbf{x})}{\int_{\mathcal{X}} e^{\frac{J(\mathbf{x})}{T_k}} \nu(d\mathbf{x})} \\
&\quad \leq \xi + \sup_{\mathbf{x} \in \mathcal{X}} \left\| \Upsilon(\mathbf{x}) - \Upsilon^* \right\| \frac{\int_{A_\delta} e^{\frac{J(\mathbf{x})}{T_k}} \nu(d\mathbf{x})}{\int_{B_\varepsilon} e^{\frac{J(\mathbf{x})}{T_k}} \nu(d\mathbf{x})} \\
&\quad \leq \xi + \sup_{\mathbf{x} \in \mathcal{X}} \left\| \Upsilon(\mathbf{x}) - \Upsilon^* \right\| e^{\frac{-(J^* - \bar{J})}{2T_k}} \frac{\nu(A_\delta)}{\nu(B_\varepsilon)}.
\end{aligned}
$$

Since $\Upsilon(\mathbf{x})$ is bounded, ξ is arbitrary, and $J^* > \bar{J}$, we have $E_{g_k}[\Upsilon(\mathbf{X})] \to \Upsilon^*$ as $T_k \to 0$. □

The next result shows that the conditional bias of the error term in (4.55) converges to zero w.p.1.

Lemma 4.23 *If Assumptions C4 and C6 hold, then*

$$
\left| \hat{E}_{\hat{\theta}_k} \left[\frac{\mathbf{U}_k}{\mathbf{V}_k} \middle| \mathcal{F}_k \right] - \frac{\bar{\mathbf{U}}_k}{\bar{\mathbf{V}}_k} \right| \to 0 \quad \text{as } k \to \infty \text{ w.p.1},
$$

where the limit is component-wise.

Proof To simplify exposition, we focus on the ith components of $\mathbf{U}_k$ and $\bar{\mathbf{U}}_k$ ($i = 1, \ldots, m$), and define $\mathbf{U}_k^i = \frac{1}{N_k} \sum_{\mathbf{x} \in \Lambda_k} e^{\frac{J(\mathbf{x})}{T_{k+1}}} \Upsilon_i(\mathbf{x}) / \hat{f}_{\hat{\theta}_k}(\mathbf{x})$, $\bar{\mathbf{U}}_k^i = \hat{E}_{\hat{\theta}_k}[\mathbf{U}_k^i | \mathcal{F}_k] = \int_{\mathcal{X}} e^{\frac{J(\mathbf{x})}{T_{k+1}}} \Upsilon_i(\mathbf{x}) \nu(d\mathbf{x})$, where $\Upsilon_i(\mathbf{x})$ is the ith component of Υ. Denote by $\mathcal{V}$ the volume of $\mathcal{X}$. Note that since $J(\mathbf{x}) > 0 \ \forall \mathbf{x}$, we have $\bar{\mathbf{V}}_k > \mathcal{V}$ (cf. Eq. (4.56)). Moreover, by Assumption C4, since the mapping Υ is bounded on $\mathcal{X}$, there exist constants C_1 and C_2 such that $C_1 \leq \Upsilon_i(\mathbf{x}) \leq C_2 \ \forall \mathbf{x}$. We have

$$
\frac{\mathbf{U}_k^i}{\mathbf{V}_k} - \frac{\bar{\mathbf{U}}_k^i}{\bar{\mathbf{V}}_k} = \frac{\mathbf{U}_k^i}{\mathbf{V}_k} - \frac{\mathbf{U}_k^i}{\bar{\mathbf{V}}_k} + \frac{\mathbf{U}_k^i}{\bar{\mathbf{V}}_k} - \frac{\bar{\mathbf{U}}_k^i}{\bar{\mathbf{V}}_k} = \frac{\mathbf{U}_k^i}{\mathbf{V}_k} \frac{\bar{\mathbf{V}}_k - \mathbf{V}_k}{\bar{\mathbf{V}}_k} + \frac{1}{\bar{\mathbf{V}}_k} \left(\mathbf{U}_k^i - \bar{\mathbf{U}}_k^i \right). \tag{4.57}
$$

Taking conditional expectations on both sides of (4.57) yields, w.p.1.,

$$\begin{aligned}
&\left|\hat{E}_{\hat{\theta}_k}\left[\frac{\mathbf{U}_k^i}{\mathbf{V}_k}\middle|\mathcal{F}_k\right]-\frac{\bar{\mathbf{U}}_k^i}{\bar{\mathbf{V}}_k}\right| \\
&\quad=\left|\hat{E}_{\hat{\theta}_k}\left[\frac{\mathbf{U}_k^i}{\mathbf{V}_k}\frac{\bar{\mathbf{V}}_k-\mathbf{V}_k}{\bar{\mathbf{V}}_k}\middle|\mathcal{F}_k\right]\right| \\
&\quad\le\frac{C}{|\bar{\mathbf{V}}_k|}\hat{E}_{\hat{\theta}_k}\left[|\bar{\mathbf{V}}_k-\mathbf{V}_k||\mathcal{F}_k\right],\quad \text{where } C=\max\left\{|C_1|,|C_2|\right\} \\
&\quad\le\frac{C}{|\bar{\mathbf{V}}_k|}\hat{E}_{\hat{\theta}_k}\left[(\bar{\mathbf{V}}_k-\mathbf{V}_k)^2|\mathcal{F}_k\right]^{1/2}\quad \text{by Hölder's inequality} \\
&\quad=\frac{C}{\sqrt{N_k}|\bar{\mathbf{V}}_k|}\hat{E}_{\hat{\theta}_k}\left[e^{2J(\mathbf{X})/T_{k+1}}\hat{f}^{-2}(\mathbf{X},\hat{\theta}_k)|\mathcal{F}_k\right]^{1/2} \\
&\quad=\frac{C}{\sqrt{N_k}}E_{g_{k+1}}\left[\frac{e^{J(\mathbf{X})/T_{k+1}}}{\bar{\mathbf{V}}_k}\hat{f}^{-1}(\mathbf{X},\hat{\theta}_k)\middle|\mathcal{F}_k\right]^{1/2} \\
&\quad\le\frac{C}{\sqrt{\mathcal{V}f_*}}\sqrt{\frac{e^{\frac{J^*}{T_{k+1}}}}{N_k\lambda_k}}.
\end{aligned}$$

Thus, the desired result follows by applying Assumption C6(c). □

We have the following convergence theorem for MARS$_1$.

Theorem 4.24 *If Assumptions C1 to C6 hold, then*

$$m(\hat{\theta}_k)\to\Upsilon^*\quad \text{as } k\to\infty\ w.p.1.$$

Proof We rewrite (4.54) in the following recursive form:

$$\eta_{k+1}=\eta_k-\xi_k,$$

where $\eta_k:=m(\hat{\theta}_k)-\Upsilon^*$, and $\xi_k=\alpha_k(m(\hat{\theta}_k)-\frac{\mathbf{U}_k}{\mathbf{V}_k})$. Let $M_k=\hat{E}_{\hat{\theta}_k}[\xi_k|\mathcal{F}_k]$ and $Z_k=\xi_k-M_k$. To show the desired convergence result, we establish that the multivariate versions of conditions (i)–(iv) in [56] hold.

[i] First we show that for every $\epsilon>0$, $P(\|\eta_k\|>\epsilon,\ \eta_k^TM_k<0\text{ i.o.})=0$. We write M_k as

$$M_k=\alpha_k\left(m(\hat{\theta}_k)-\Upsilon^*+\Upsilon^*-E_{g_{k+1}}[\Upsilon(\mathbf{X})]+E_{g_{k+1}}[\Upsilon(\mathbf{X})]-\hat{E}_{\hat{\theta}_k}\left[\frac{\mathbf{U}_k}{\mathbf{V}_k}\middle|\mathcal{F}_k\right]\right).\tag{4.58}$$

It follows that

$$\eta_k^T M_k = \alpha_k \Bigg(\|\eta_k\|^2 + \eta_k^T \left(\Upsilon^* - E_{g_{k+1}}[\Upsilon(\mathbf{X})] \right) + \eta_k^T \left(E_{g_{k+1}}[\Upsilon(\mathbf{X})] - \hat{E}_{\hat{\theta}_k}\left[\frac{\mathbf{U}_k}{\mathbf{V}_k} \middle| \mathcal{F}_k \right] \right) \Bigg).$$

Since η_k is bounded, by Lemma 4.22, the second term in the parentheses above vanishes to zero as $k \to \infty$, whereas Lemma 4.23 implies that the third term also vanishes to zero w.p.1. as $k \to \infty$. Therefore, for almost every sample path generated by MARS$_1$, we must have $\eta_k^T M_k > 0$ whenever $\|\eta_k\| > \epsilon$ for k sufficiently large, i.e., $P(\|\eta_k\| > \epsilon,\ \eta_k^T M_k < 0 \text{ i.o.}) = 0$.

[**ii**] Note that $m(\hat{\theta}_k) = E_{\hat{\theta}_k}[\Upsilon(\mathbf{X})]$ and $\frac{\mathbf{U}_k}{\mathbf{V}_k} = E_{\bar{g}_{k+1}}[\Upsilon(\mathbf{X})]$, where $\bar{g}_{k+1}$ is defined in (4.53). Since the mapping Υ is bounded on $\mathcal{X}$ by Assumption C4, both $m(\hat{\theta}_k)$ and $\frac{\mathbf{U}_k}{\mathbf{V}_k}$ are bounded. Moreover, we have from Assumption C5 that $\alpha_k \to 0$ as $k \to \infty$. Therefore, $\|M_k\|(1 + \|\eta_k\|)^{-1} \to 0$ as $k \to \infty$ w.p.1, which establishes condition (ii) in [56].

[**iii**] By definition, we have $Z_k = \alpha_k(\hat{E}_{\hat{\theta}_k}[\frac{\mathbf{U}_k}{\mathbf{V}_k}|\mathcal{F}_k] - \frac{\mathbf{U}_k}{\mathbf{V}_k})$. Therefore,

$$\sum_{k=1}^{\infty} E\left[\|Z_k\|^2\right] = \sum_{k=1}^{\infty} \alpha_k^2 E\left[\left(\hat{E}_{\hat{\theta}_k}\left[\frac{\mathbf{U}_k}{\mathbf{V}_k} \middle| \mathcal{F}_k \right] - \frac{\mathbf{U}_k}{\mathbf{V}_k} \right)^T \left(\hat{E}_{\hat{\theta}_k}\left[\frac{\mathbf{U}_k}{\mathbf{V}_k} \middle| \mathcal{F}_k \right] - \frac{\mathbf{U}_k}{\mathbf{V}_k} \right) \right] < \infty,$$

since $\frac{\mathbf{U}_k}{\mathbf{V}_k}$ is bounded and $\sum_{k=1}^{\infty} \alpha_k^2 < \infty$ by Assumption C5.

[**iv**] Finally, we establish that $P(\liminf_{k\to\infty} \|\eta_k\| > 0, \sum_{k=1}^{\infty} \|M_k\| < \infty) = 0$. From (4.58), we have $\|M_k\| \geq \alpha_k(\|\eta_k\| - \|\Upsilon^* - E_{g_{k+1}}[\Upsilon(\mathbf{X})]\| - \|E_{g_{k+1}}[\Upsilon(\mathbf{X})] - \hat{E}_{\hat{\theta}_k}[\frac{\mathbf{U}_k}{\mathbf{V}_k}|\mathcal{F}_k]\|)$. Let $\Omega_1 = \{\liminf_{k\to\infty} \|\eta_k\| > 0\}$ and $\Omega_2 = \{\sum_{k=1}^{\infty} \|M_k\| < \infty\}$. For every sample point $\omega \in \Omega_1$, we can find a $\delta > 0$ such that $\liminf_{k\to\infty} \|\eta_k\| > \delta > 0$. This implies that there exists a $K_\delta(\omega)$ such that $\|\eta_k\| \geq \delta\ \forall k \geq K_\delta(\omega)$. In addition, let $\Omega_3 = \{\|E_{g_{k+1}}[\Upsilon(\mathbf{X})] - \hat{E}_{\hat{\theta}_k}[\frac{\mathbf{U}_k}{\mathbf{V}_k}|\mathcal{F}_k]\| \to 0\}$. Note that Lemma 4.23 implies $P(\Omega_3) = 1$. Since $E_{g_{k+1}}[\Upsilon(\mathbf{X})] \to \Upsilon^*$ as $k \to \infty$, there exists a $\bar{K}_{\delta/2}(\omega)$ for every $\omega \in \Omega_3$ such that

$$\left\| \Upsilon^* - E_{g_{k+1}}\left[\Upsilon(\mathbf{X})\right] \right\| + \left\| E_{g_{k+1}}\left[\Upsilon(\mathbf{X})\right] - \hat{E}_{\hat{\theta}_k}\left[\frac{\mathbf{U}_k}{\mathbf{V}_k} \middle| \mathcal{F}_k \right] \right\| < \frac{\delta}{2}$$

for all $k \geq \bar{K}_{\delta/2}(\omega)$. Consequently, we have, for every $\omega \in \Omega_1 \cap \Omega_3$, $\|M_k\| > \frac{\delta}{2}\alpha_k$ for all $k \geq K^*(\omega) := \max\{K_\delta(\omega), \bar{K}_{\delta/2}(\omega)\}$. Thus by Assumption C5,

$$\sum_{k=1}^{\infty} \|M_k\| \geq \sum_{k=K^*(\omega)}^{\infty} \|M_k\| \geq \frac{\delta}{2} \sum_{k=K^*(\omega)}^{\infty} \alpha_k = \infty \quad \forall \omega \in \Omega_1 \cap \Omega_3,$$

which implies $P(\Omega_1 \cap \Omega_2 \cap \Omega_3) = 0$. Hence, it follows that $P(\Omega_1 \cap \Omega_2) = P(\Omega_1 \cap \Omega_2 \cap \Omega_3) + P(\Omega_1 \cap \Omega_2 \cap \Omega_3^c) \leq P(\Omega_3^c) = 0$, this shows condition (iv) in [56], which completes the proof. □

The interpretation of Theorem 4.24 depends on the parameterized distribution family used in MARS$_1$ and, in particular, on the specific form of the function $\Upsilon(\mathbf{x})$. For example, in continuous optimization, if $T^* = 0$ and normal distributions are used as parameterized family, then Theorem 4.24 implies that the sequence of sampling distributions $\{f_{\hat{\theta}_k}\}$ in MARS will converge to a degenerate distribution with all mass concentrated at the global optimizer $\mathbf{x}^*$, in the sense that $\lim_{k\to\infty} E_{\hat{\theta}_k}[\mathbf{X}] = \mathbf{x}^*$ and $\lim_{k\to\infty} \mathrm{Cov}_{\hat{\theta}_k}[\mathbf{X}] = 0$ w.p.1. Another case of interest is when independent univariate density/mass functions are used and the parameterized family takes the form $f(\mathbf{x}, \theta) = \prod_{i=1}^{n} \exp(\mathbf{x}_i \vartheta_i - K(\vartheta_i))$, where $\mathbf{x}_i$ and ϑ_i are the respective ith components of $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)^T$ and the parameter vector $\theta = (\vartheta_1, \ldots, \vartheta_n)^T$, in which case, we have $\Upsilon(\mathbf{x}) = \mathbf{x}$ and $m(\hat{\theta}_k) = E_{\hat{\theta}_k}[\mathbf{X}]$. Thus, if $T^* = 0$, then the result of Theorem 4.24 reduces to $\lim_{k\to\infty} E_{\hat{\theta}_k}[\mathbf{X}] = \mathbf{x}^*$ w.p.1, i.e., the means of the sequence of sampling distributions converge to $\mathbf{x}^*$ w.p.1. As a third example, consider a discrete optimization problem with a feasible region $\mathcal{X}$ that contains m distinct values $\{a_1, \ldots, a_m\}$. To approach the problem, we can specify an m-by-1 probability vector Q, whose ith entry q_i indicates the probability that a solution will take the ith value $a_i \in \mathcal{X}$. When parameterized by Q, the probability of sampling a solution $\mathbf{x}$ can be written as $f(\mathbf{x}, \theta) = \prod_{i=1}^{m} q_i^{I\{\mathbf{x}=a_i\}} := e^{\theta^T \Upsilon(\mathbf{x})}$, where $\theta = [\ln q_1, \ldots, \ln q_m]^T$ and $\Upsilon(\mathbf{x}) = [I\{\mathbf{x} = a_1\}, \ldots, I\{\mathbf{x} = a_m\}]^T$. Note that $\Upsilon(\mathbf{x})$ satisfies Assumption C4. Thus, when $T^* = 0$, a straightforward interpretation of Theorem 4.24 yields

$$\lim_{k\to\infty} \sum_{\mathbf{x}\in\mathcal{X}} \prod_{i=1}^{m} \left(q_i^k\right)^{I\{\mathbf{x}=a_i\}} I\{\mathbf{x} = a_j\} = I\left\{\mathbf{x}^* = a_j\right\} \quad \forall j \text{ w.p.1},$$

where q_i^k is the ith entry of the vector Q_k obtained at the kth iteration of MARS. This implies that $\lim_{k\to\infty} q_i^k = I\{\mathbf{x}^* = a_i\}$ w.p.1 $\forall i$. In other words, the sequence of probability vectors Q_k will converge to a limiting vector that assigns unit mass to the global optimum $\mathbf{x}^*$.

4.6.1.2 Asymptotic Normality of MARS$_1$

To fix ideas, we consider a sample size sequence $N_k = \Theta(k^\beta)$ and a gain sequence of the form $\alpha_k = c/k^\alpha$ for some constants $\beta > 0$, $c > 0$, and $\alpha \in (\frac{1}{2}, 1)$. Note that such a choice of α_k satisfies Assumption C5. In addition, we require that $\{T_k\}$ and $\{\lambda_k\}$ satisfy the following strengthened version of Assumption C6.

Assumption D1 For a given sample size sequence $N_k = \Theta(k^\beta)$ and a gain sequence $\alpha_k = \Theta(k^{-\alpha})$, the sequence $\{T_k\}$ satisfies $T_k > T^* > 0$ $\forall k$ and $\lim_{k\to\infty} k^{\frac{\alpha+\beta}{2}}(\frac{1}{T^*} - \frac{1}{T_k}) = 0$. In addition, the sequence $\{\lambda_k\}$ satisfies $\lambda_k > 0$ $\forall k$, $\lambda_k \to \lambda^* \in [0, 1)$ as $k \to \infty$, and $\lambda_k = \Omega(k^{-\gamma})$ for some positive constant $\gamma < \frac{\beta}{2}$.

It is easy to see that Theorem 4.24 still holds with Assumption C6 replaced by Assumption D1. Thus, by the invertibility of $m(\cdot)$, the sequence of parameters $\{\hat{\theta}_k\}$

generated by MARS$_1$ will settle down to some limit (possibly infinite) as $k \to \infty$ w.p.1. We shall also assume the following regularity condition:

Assumption D2 The limit of the sequence of parameters $\{\hat{\theta}_k\}$ generated by MARS$_1$ as $k \to \infty$ is an interior point of Θ, i.e., $m^{-1}(\Upsilon^*) \in \text{int}(\Theta)$.

Since $m(\cdot)$ is continuously differentiable on int(Θ), Assumption D2 implies that the Jacobian of $m(\cdot)$ at $m^{-1}(\Upsilon^*)$ is strictly positive definite. Therefore, by the inverse function theorem, there exists an open neighborhood $\mathcal{N}(\Upsilon^*)$ of Υ^* such that $m^{-1}(\cdot)$ is continuously differentiable on $\mathcal{N}(\Upsilon^*)$. This, together with the boundedness of Υ, further implies that the sequence of sampling distributions $\{f(\cdot, \hat{\theta}_k)\}$ in MARS$_1$ will converge point-wise to a limiting distribution $f(\cdot, m^{-1}(\Upsilon^*))$ w.p.1.

Given the specific forms of N_k and α_k, we can rewrite (4.54) in the form of a recursion in [58]:

$$\eta_{k+1} = \big(1 - ck^{-\alpha}\big)\eta_k + k^{-(2\alpha+\beta)/2} R_k + k^{-(3\alpha+\beta)/2} W_k,$$

where $\eta_k = m(\hat{\theta}_k) - \Upsilon^*$,

$$R_k = ck^{\beta/2}\left(\frac{\mathbf{U}_k}{\mathbf{V}_k} - \hat{E}_{\hat{\theta}_k}\left[\frac{\mathbf{U}_k}{\mathbf{V}_k}\middle|\mathcal{F}_k\right]\right), \qquad W_k = ck^{(\alpha+\beta)/2}\left[\hat{E}_{\hat{\theta}_k}\left[\frac{\mathbf{U}_k}{\mathbf{V}_k}\middle|\mathcal{F}_k\right] - \Upsilon^*\right].$$

The term R_k has the following properties.

Lemma 4.25 *If Assumptions C1–C4, D1, and D2 hold, then there exists a symmetric positive semi-definite matrix Σ such that $\hat{E}_{\hat{\theta}_k}[R_k R_k^T | \mathcal{F}_k] \to \Sigma$ as $k \to \infty$ w.p.1. In addition, the sequence $\{R_k\}$ is uniformly square integrable in the sense that*

$$\lim_{k\to\infty} E\big[I\big\{\|R_k\|^2 \geq rk^{\alpha}\big\}\|R_k\|^2\big] = 0 \quad \forall r > 0.$$

Proof Let $\mathbf{U}_k^i = N_k^{-1} \sum_{\mathbf{x}\in\Lambda_k} e^{\frac{J(\mathbf{x})}{T_{k+1}}} \Upsilon_i(\mathbf{x}) / \hat{f}(\mathbf{x}, \hat{\theta}_k)$ and $\bar{\mathbf{U}}_k^i = \hat{E}_{\hat{\theta}_k}[\mathbf{U}_k | \mathcal{F}_k] = \int_{\mathcal{X}} e^{\frac{J(\mathbf{x})}{T_{k+1}}} \Upsilon_i(\mathbf{x}) \nu(d\mathbf{x})$ be the ith components of $\mathbf{U}_k$ and its conditional expectation. Denote by $\Sigma_{i,j}^k$ the (i, j)th entry of the matrix $\Sigma^k := \hat{E}_{\hat{\theta}_k}[R_k R_k^T | \mathcal{F}_k]$ and by $\mathcal{V}$ the volume of $\mathcal{X}$. Define $\epsilon = \frac{\mathcal{V}}{2}$, and let $\Omega_k = \{|\mathbf{U}_k^i - \bar{\mathbf{U}}_k^i| < \epsilon \cap |\mathbf{V}_k - \bar{\mathbf{V}}_k| < \epsilon\}$ and Ω_k^c be the complement of Ω_k.

By using a second order two-variable Taylor expansion of $\frac{\mathbf{U}_k^i}{\mathbf{V}_k}$ around the neighborhood Ω_k of $(\bar{\mathbf{U}}_k^i, \bar{\mathbf{V}}_k)$, we can write, for every sample path generated by MARS$_1$,

$$\begin{aligned}\frac{\mathbf{U}_k^i}{\mathbf{V}_k} = \Bigg[&\frac{\bar{\mathbf{U}}_k^i}{\bar{\mathbf{V}}_k} - \frac{\bar{\mathbf{U}}_k^i}{(\bar{\mathbf{V}}_k)^2}(\mathbf{V}_k - \bar{\mathbf{V}}_k) + \frac{1}{\bar{\mathbf{V}}_k}\big(\mathbf{U}_k^i - \bar{\mathbf{U}}_k^i\big) + \frac{\tilde{\mathbf{U}}_k}{\tilde{\mathbf{V}}_k^3}(\mathbf{V}_k - \bar{\mathbf{V}}_k)^2 \\ &- \frac{1}{\tilde{\mathbf{V}}_k^2}(\mathbf{V}_k - \bar{\mathbf{V}}_k)\big(\mathbf{U}_k^i - \bar{\mathbf{U}}_k^i\big)\Bigg] I\{\Omega_k\} + \frac{\mathbf{U}_k^i}{\mathbf{V}_k} I\big\{\Omega_k^c\big\}, \end{aligned} \tag{4.59}$$

where $\tilde{\mathbf{U}}_k$ and $\tilde{\mathbf{V}}_k$ are on the respective line segments from $\bar{\mathbf{U}}_k^i$ to $\mathbf{U}_k^i$ and from $\bar{\mathbf{V}}_k$ to $\mathbf{V}_k$. By rearranging terms in (4.59), we have

$$\begin{aligned}\frac{\mathbf{U}_k^i}{\mathbf{V}_k}-\frac{\bar{\mathbf{U}}_k^i}{\bar{\mathbf{V}}_k}&=\frac{1}{\bar{\mathbf{V}}_k}\big(\mathbf{U}_k^i-\bar{\mathbf{U}}_k^i\big)-\frac{\bar{\mathbf{U}}_k^i}{(\bar{\mathbf{V}}_k)^2}(\mathbf{V}_k-\bar{\mathbf{V}}_k)\\&\quad+\left[\frac{\tilde{\mathbf{U}}_k}{\tilde{\mathbf{V}}_k^3}(\mathbf{V}_k-\bar{\mathbf{V}}_k)^2-\frac{1}{\tilde{\mathbf{V}}_k^2}(\mathbf{V}_k-\bar{\mathbf{V}}_k)\big(\mathbf{U}_k^i-\bar{\mathbf{U}}_k^i\big)\right]I\{\Omega_k\}\\&\quad+\left[\frac{\mathbf{U}_k^i}{\mathbf{V}_k}-\frac{\bar{\mathbf{U}}_k^i}{\bar{\mathbf{V}}_k}-\frac{1}{\bar{\mathbf{V}}_k}\big(\mathbf{U}_k^i-\bar{\mathbf{U}}_k^i\big)+\frac{\bar{\mathbf{U}}_k^i}{(\bar{\mathbf{V}}_k)^2}(\mathbf{V}_k-\bar{\mathbf{V}}_k)\right]I\big\{\Omega_k^c\big\}.\end{aligned}\tag{4.60}$$

It follows that

$$\begin{aligned}\Sigma_{i,j}^k&=c^2k^\beta\hat{E}_{\hat{\theta}_k}\left[\left(\frac{\mathbf{U}_k^i}{\mathbf{V}_k}-\hat{E}_{\hat{\theta}_k}\left[\frac{\mathbf{U}_k^i}{\mathbf{V}_k}\middle|\mathcal{F}_k\right]\right)\left(\frac{\mathbf{U}_k^j}{\mathbf{V}_k}-\hat{E}_{\hat{\theta}_k}\left[\frac{\mathbf{U}_k^j}{\mathbf{V}_k}\middle|\mathcal{F}_k\right]\right)\middle|\mathcal{F}_k\right]\\&=c^2k^\beta\frac{1}{\bar{\mathbf{V}}_k^2}\hat{E}_{\hat{\theta}_k}\big[\big(\mathbf{U}_k^i-\bar{\mathbf{U}}_k^i\big)\big(\mathbf{U}_k^j-\bar{\mathbf{U}}_k^j\big)\big|\mathcal{F}_k\big]\\&\quad-c^2k^\beta\frac{\bar{\mathbf{U}}_k^j}{\bar{\mathbf{V}}_k^3}\hat{E}_{\hat{\theta}_k}\big[\big(\mathbf{U}_k^i-\bar{\mathbf{U}}_k^i\big)(\mathbf{V}_k-\bar{\mathbf{V}}_k)\big|\mathcal{F}_k\big]\\&\quad-c^2k^\beta\frac{\bar{\mathbf{U}}_k^i}{\bar{\mathbf{V}}_k^3}\hat{E}_{\hat{\theta}_k}\big[\big(\mathbf{U}_k^j-\bar{\mathbf{U}}_k^j\big)(\mathbf{V}_k-\bar{\mathbf{V}}_k)\big|\mathcal{F}_k\big]\\&\quad+c^2k^\beta\frac{\bar{\mathbf{U}}_k^i\bar{\mathbf{U}}_k^j}{\bar{\mathbf{V}}_k^4}\hat{E}_{\hat{\theta}_k}\big[(\mathbf{V}_k-\bar{\mathbf{V}}_k)^2\big|\mathcal{F}_k\big]\\&\quad+c^2k^\beta\mathcal{R}_k\\&=[\mathrm{i}]-[\mathrm{ii}]-[\mathrm{iii}]+[\mathrm{iv}]+c^2k^\beta\mathcal{R}_k,\end{aligned}$$

where $\mathcal{R}_k$ represents a remainder term. We now analyze terms [i]–[iv].

$$\begin{aligned}[\mathrm{i}]&=c^2k^\beta\frac{1}{\bar{\mathbf{V}}_k^2}\big(\hat{E}_{\hat{\theta}_k}\big[\mathbf{U}_k^i\mathbf{U}_k^j\big|\mathcal{F}_k\big]-\bar{\mathbf{U}}_k^i\bar{\mathbf{U}}_k^j\big)\\&=\frac{c^2k^\beta}{\bar{\mathbf{V}}_k^2}\left[\frac{1}{N_k^2}\hat{E}_{\hat{\theta}_k}\left[\sum_{\mathbf{x}\in\Lambda_k}e^{\frac{J(\mathbf{x})}{T_{k+1}}}\Upsilon_i(\mathbf{x})/\hat{f}(\mathbf{x},\hat{\theta}_k)\cdot\sum_{\mathbf{x}\in\Lambda_k}e^{\frac{J(\mathbf{x})}{T_{k+1}}}\Upsilon_j(\mathbf{x})/\hat{f}(\mathbf{x},\hat{\theta}_k)\middle|\mathcal{F}_k\right]\right.\\&\qquad\left.-\bar{\mathbf{U}}_k^i\bar{\mathbf{U}}_k^j\right]\\&=c^2k^\beta\frac{1}{\bar{\mathbf{V}}_k^2}\frac{1}{N_k}\big(\hat{E}_{\hat{\theta}_k}\big[e^{\frac{2J(\mathbf{X})}{T_{k+1}}}\Upsilon_i(\mathbf{X})\Upsilon_j(\mathbf{X})/\hat{f}^2(\mathbf{X},\hat{\theta}_k)\big|\mathcal{F}_k\big]-\bar{\mathbf{U}}_k^i\bar{\mathbf{U}}_k^j\big)\end{aligned}$$

$$= \frac{c^2 k^\beta}{N_k}\left(\hat{E}_{\hat{\theta}_k}\left[\frac{1}{\bar{\mathbf{V}}_k^2} e^{\frac{2J(\mathbf{X})}{T_{k+1}}} \Upsilon_i(\mathbf{X})\Upsilon_j(\mathbf{X})/\hat{f}^2(\mathbf{X},\hat{\theta}_k)\middle|\mathcal{F}_k\right] - \frac{\bar{\mathbf{U}}_k^i \bar{\mathbf{U}}_k^j}{\bar{\mathbf{V}}_k^2}\right)$$

$$= \frac{c^2 k^\beta}{N_k}\left(E_{g_k}\left[\Upsilon_i(\mathbf{X})\Upsilon_j(\mathbf{X})\frac{g_k(\mathbf{X})}{\hat{f}(\mathbf{X},\hat{\theta}_k)}\middle|\mathcal{F}_k\right] - E_{g_k}\left[\Upsilon_i(\mathbf{X})\right]E_{g_k}\left[\Upsilon_j(\mathbf{X})\right]\right).$$

By following a similar argument as above, it can be shown that

$$[\text{ii}] = \frac{c^2 k^\beta}{N_k}\left(E_{g_k}\left[\Upsilon_j(\mathbf{X})\right]E_{g_k}\left[\Upsilon_i(\mathbf{X})\frac{g_k(\mathbf{X})}{\hat{f}(\mathbf{X},\hat{\theta}_k)}\middle|\mathcal{F}_k\right] - E_{g_k}\left[\Upsilon_i(\mathbf{X})\right]E_{g_k}\left[\Upsilon_j(\mathbf{X})\right]\right),$$

$$[\text{iii}] = \frac{c^2 k^\beta}{N_k}\left(E_{g_k}\left[\Upsilon_i(\mathbf{X})\right]E_{g_k}\left[\Upsilon_j(\mathbf{X})\frac{g_k(\mathbf{X})}{\hat{f}(\mathbf{X},\hat{\theta}_k)}\middle|\mathcal{F}_k\right] - E_{g_k}\left[\Upsilon_i(\mathbf{X})\right]E_{g_k}\left[\Upsilon_j(\mathbf{X})\right]\right),$$

$$[\text{iv}] = \frac{c^2 k^\beta}{N_k}\left(E_{g_k}\left[\Upsilon_i(\mathbf{X})\right]E_{g_k}\left[\Upsilon_j(\mathbf{X})\right]E_{g_k}\left[\frac{g_k(\mathbf{X})}{\hat{f}(\mathbf{X},\hat{\theta}_k)}\middle|\mathcal{F}_k\right]\right.$$
$$\left. - E_{g_k}\left[\Upsilon_i(\mathbf{X})\right]E_{g_k}\left[\Upsilon_j(\mathbf{X})\right]\right).$$

Note that $|e^{\frac{J(\mathbf{x})}{T_{k+1}}}\Upsilon_i(\mathbf{x})/\hat{f}(\mathbf{x},\hat{\theta}_k)| \le e^{\frac{J^*}{T^*}}|\Upsilon_i(\mathbf{x})|/\lambda_k f_*$. Thus by Assumption C4, Hoeffding's inequality implies that

$$\hat{E}_{\hat{\theta}_k}\left[I\{\Omega_k^c\}\middle|\mathcal{F}_k\right] \le \hat{P}_{\hat{\theta}_k}\left(|\mathbf{U}_k^i - \bar{\mathbf{U}}_k^i| \ge \epsilon|\mathcal{F}_k\right) + \hat{P}_{\hat{\theta}_k}\left(|\mathbf{V}_k - \bar{\mathbf{V}}_k| \ge \epsilon\middle|\mathcal{F}_k\right)$$
$$= O\left(e^{-CN_k\lambda_k^2}\right)$$

for some ϵ-dependent constant $C > 0$. This result, when combined with the conditions $N_k = \Theta(k^\beta)$, $\lambda_k = \Omega(k^{-\gamma})$, and $\gamma < \frac{\beta}{2}$ (Assumption D1), indicates that all terms containing $I\{\Omega_k^c\}$ in the remainder $\mathcal{R}_k$ are on the order of $o(k^{-\beta})$. Moreover, a straightforward calculation also shows that all terms involving $I\{\Omega_k\}$ in $\mathcal{R}_k$ are higher order terms of N_k^{-1}. Consequently, we have $ck^\beta\mathcal{R}_k = o(1)$ by taking $N_k = \Theta(k^\beta)$.

Since $T_k \to T^* > 0$, it is easy to show that $\lim_{k\to\infty} g_k(\mathbf{x}) = g^*(\mathbf{x})\ \forall \mathbf{x} \in \mathcal{X}$, where $g^*(\mathbf{x}) = \frac{e^{J(\mathbf{x})/T^*}}{\int_{\mathcal{X}} e^{J(\mathbf{x})/T^*}\nu(d\mathbf{x})}$. Thus, by the point-wise convergence of $\{f(\cdot,\hat{\theta}_k)\}$ (see the discussion after Assumption D2), the dominated convergence theorem implies that the (i,j)th entry of Σ^k as $k \to \infty$ is

$$\Sigma_{i,j} = \Psi\left(E_{g^*}\left[\Upsilon_i(\mathbf{X})\Upsilon_j(\mathbf{X})\frac{g^*(\mathbf{X})}{\hat{f}(\mathbf{X},m^{-1}(\Upsilon^*))}\right]\right.$$
$$- E_{g^*}\left[\Upsilon_j(\mathbf{X})\right]E_{g^*}\left[\Upsilon_i(\mathbf{X})\frac{g^*(\mathbf{X})}{\hat{f}(\mathbf{X},m^{-1}(\Upsilon^*))}\right]$$
$$\left. + E_{g^*}\left[\Upsilon_i(\mathbf{X})\right]E_{g^*}\left[\Upsilon_j(\mathbf{X})\right]E_{g^*}\left[\frac{g^*(\mathbf{X})}{\hat{f}(\mathbf{X},m^{-1}(\Upsilon^*))}\right]\right.$$

$$
\begin{aligned}
&- E_{g^*}\big[\Upsilon_i(\mathbf{X})\big]E_{g^*}\left[\Upsilon_j(\mathbf{X})\frac{g^*(\mathbf{X})}{\hat{f}(\mathbf{X}, m^{-1}(\Upsilon^*))}\right]\Big) \\
&= \Psi E_{g^*}\left[\big(\Upsilon_i(\mathbf{X}) - E_{g^*}\big[\Upsilon_i(\mathbf{X})\big]\big)\big(\Upsilon_j(\mathbf{X}) - E_{g^*}\big[\Upsilon_j(\mathbf{X})\big]\big)\frac{g^*(\mathbf{X})}{\hat{f}(\mathbf{X}, m^{-1}(\Upsilon^*))}\right] \\
&= \Psi \hat{E}_{m^{-1}(\Upsilon^*)}\Bigg[\big(\Upsilon_i(\mathbf{X}) - E_{g^*}\big[\Upsilon_i(\mathbf{X})\big]\big) \\
&\quad \times \big(\Upsilon_j(\mathbf{X}) - E_{g^*}\big[\Upsilon_j(\mathbf{X})\big]\big)\left(\frac{g^*(\mathbf{X})}{\hat{f}(\mathbf{X}, m^{-1}(\Upsilon^*))}\right)^2\Bigg]
\end{aligned}
$$

for some constant $\Psi > 0$. Therefore, the limiting matrix Σ is given by

$$
\Sigma = \Psi \widehat{\mathrm{Cov}}_{m^{-1}(\Upsilon^*)}\left[\big(\Upsilon(\mathbf{X}) - E_{g^*}\big[\Upsilon(\mathbf{X})\big]\big)\frac{g^*(\mathbf{X})}{\hat{f}(\mathbf{X}, m^{-1}(\Upsilon^*))}\right],
$$

where $\widehat{\mathrm{Cov}}_{m^{-1}(\Upsilon^*)}(\cdot)$ is the covariance under

$$
\hat{f}\big(\mathbf{x}, m^{-1}\big(\Upsilon^*\big)\big) = \big(1 - \lambda^*\big) f\big(\mathbf{x}, m^{-1}\big(\Upsilon^*\big)\big) + \lambda^* f(\mathbf{x}, \hat{\theta}_0).
$$

We now show the second claim in the lemma. By Hölder's inequality, we have

$$
\begin{aligned}
&\lim_{k\to\infty} E\big[I\big\{\|R_k\|^2 \geq rk^\alpha\big\}\|R_k\|^2\big] \\
&\quad \leq \limsup_{k\to\infty}\big[P\big(\|R_k\|^2 \geq rk^\alpha\big)\big]^{1/2}\big[E\big[\|R_k\|^4\big]\big]^{1/2}. \qquad (4.61)
\end{aligned}
$$

By the definition of R_k, it follows that

$$
\begin{aligned}
P\big(\|R_k\|^2 \geq rk^\alpha\big) &= P\left(\left\|\frac{\mathbf{U}_k}{\mathbf{V}_k} - \hat{E}_{\hat{\theta}_k}\left[\frac{\mathbf{U}_k}{\mathbf{V}_k}\middle|\mathcal{F}_k\right]\right\| \geq \frac{\sqrt{r}}{c}k^{\frac{\alpha-\beta}{2}}\right) \\
&\leq \frac{E[\|\frac{\mathbf{U}_k}{\mathbf{V}_k} - \hat{E}_{\hat{\theta}_k}[\frac{\mathbf{U}_k}{\mathbf{V}_k}|\mathcal{F}_k]\|^2]}{\frac{r}{c^2}k^{\alpha-\beta}} \quad \text{by Chebyshev's inequality} \\
&= \frac{E[\hat{E}_{\hat{\theta}_k}[\|\frac{\mathbf{U}_k}{\mathbf{V}_k} - \hat{E}_{\hat{\theta}_k}[\frac{\mathbf{U}_k}{\mathbf{V}_k}|\mathcal{F}_k]\|^2|\mathcal{F}_k]]}{\frac{r}{c^2}k^{\alpha-\beta}} \\
&= \frac{E[\hat{E}_{\hat{\theta}_k}[c^2k^\beta\|\frac{\mathbf{U}_k}{\mathbf{V}_k} - \hat{E}_{\hat{\theta}_k}[\frac{\mathbf{U}_k}{\mathbf{V}_k}|\mathcal{F}_k]\|^2|\mathcal{F}_k]]}{rk^\alpha} \\
&= \frac{E[\mathrm{tr}(\Sigma^k)]}{rk^\alpha} \\
&= O\big(k^{-\alpha}\big)
\end{aligned}
$$

by taking $N_k = \Theta(k^\beta)$ and using an argument similar to the proof of the previous part of the theorem, where $\mathrm{tr}(\Sigma^k)$ is the trace of Σ^k. On the other hand, it is

straightforward to show that

$$\begin{aligned} E\left[\|R_k\|^4\right] &= c^4 k^{2\beta} E\left[\left\| \frac{\mathbf{U}_k}{\mathbf{V}_k} - \hat{E}_{\hat{\theta}_k}\left[\frac{\mathbf{U}_k}{\mathbf{V}_k}\middle|\mathcal{F}_k\right]\right\|^4\right] \\ &= c^4 k^{2\beta} E\left[\hat{E}_{\hat{\theta}_k}\left[\left\| \frac{\mathbf{U}_k}{\mathbf{V}_k} - \hat{E}_{\hat{\theta}_k}\left[\frac{\mathbf{U}_k}{\mathbf{V}_k}\middle|\mathcal{F}_k\right]\right\|^4\middle|\mathcal{F}_k\right]\right] \\ &= c^4 k^{2\beta} O\left(N_k^{-2}\right) \\ &= O(1). \end{aligned}$$

Consequently, the right-hand-size of (4.61) is bounded above by $O(k^{-\alpha/2})$, which approaches zero as $k \to \infty$. □

We next show that the term W_k vanishes to zero w.p.1. To do so, we break W_k into two parts and write $W_k = ck^{(\alpha+\beta)/2} W_{1,k} + ck^{(\alpha+\beta)/2} W_{2,k}$, where

$$W_{1,k} = \frac{\hat{E}_{\hat{\theta}_k}[\mathbf{U}_k|\mathcal{F}_k]}{\hat{E}_{\hat{\theta}_k}[\mathbf{V}_k|\mathcal{F}_k]} - \Upsilon^* \quad \text{and} \quad W_{2,k} = \hat{E}_{\hat{\theta}_k}\left[\frac{\mathbf{U}_k}{\mathbf{V}_k}\middle|\mathcal{F}_k\right] - \frac{\hat{E}_{\hat{\theta}_k}[\mathbf{U}_k|\mathcal{F}_k]}{\hat{E}_{\hat{\theta}_k}[\mathbf{V}_k|\mathcal{F}_k]}.$$

The convergence of W_k is a direct consequence of the following lemmas, which are strengthened versions of Lemma 4.22 and Lemma 4.23.

Lemma 4.26 *If Assumptions* C2, C3, *and* D1 *hold, then*

$$k^{\frac{\alpha+\beta}{2}} W_{1,k} \to 0 \quad \text{as } k \to \infty.$$

Proof Note that since $T_{k'} > T^* \ \forall k'$ by D1, for any $k > 0$, we can find a monotonically non-increasing subsequence $\{T_{k_i},\ i = 0, 1, \ldots\}$ such that $T_{k_0} = T_{k+1}$ and $\lim_{i\to\infty} T_{k_i} = T^*$. We have, for any integer $N > 0$,

$$\begin{aligned} &\int_{\mathcal{X}} \left|g_{k_N}(\mathbf{x}) - g_{k+1}(\mathbf{x})\right| \nu(d\mathbf{x}) \\ &\leq \int_{\mathcal{X}} \sum_{i=0}^{N-1} \left|g_{k_{i+1}}(\mathbf{x}) - g_{k_i}(\mathbf{x})\right| \nu(d\mathbf{x}) \\ &= \sum_{i=0}^{N-1} \int_{\mathcal{X}} \left|g_{k_{i+1}}(\mathbf{x}) - g_{k_i}(\mathbf{x})\right| \nu(d\mathbf{x}) \\ &\leq \sum_{i=0}^{N-1} \sqrt{2\mathcal{D}(g_{k_{i+1}}, g_{k_i})} \quad \text{by Pinsker's inequality (e.g., [176]).} \end{aligned}$$

On the other hand, we have from the definition of KL-divergence,

$$
\begin{aligned}
&\mathcal{D}(g_{k_{i+1}}, g_{k_i}) \\
&\quad = E_{g_{k_{i+1}}}\left[\frac{g_{k_{i+1}}(\mathbf{X})}{g_{k_i}(\mathbf{X})}\right] \\
&\quad = \left(\frac{1}{T_{k_{i+1}}} - \frac{1}{T_{k_i}}\right) E_{g_{k_{i+1}}}\left[J(\mathbf{X})\right] - \ln E_{g_{k_i}}\left[e^{(\frac{1}{T_{k_{i+1}}} - \frac{1}{T_{k_i}})J(\mathbf{X})}\right] \\
&\quad \le \left(\frac{1}{T_{k_{i+1}}} - \frac{1}{T_{k_i}}\right)\left[E_{g_{k_{i+1}}}\left[J(\mathbf{X})\right] - E_{g_{k_i}}\left[J(\mathbf{X})\right]\right] \quad \text{by Jensen's inequality} \\
&\quad \le \left(\frac{1}{T_{k_{i+1}}} - \frac{1}{T_{k_i}}\right) J^* \int_{\mathcal{X}} \left|g_{k_{i+1}}(\mathbf{x}) - g_{k_i}(\mathbf{x})\right| \nu(d\mathbf{x}) \\
&\quad \le \left(\frac{1}{T_{k_{i+1}}} - \frac{1}{T_{k_i}}\right) J^* \sqrt{2\mathcal{D}(g_{k_{i+1}}, g_{k_i})} \quad \text{by Pinsker's inequality.}
\end{aligned}
$$

This implies $\sqrt{2\mathcal{D}(g_{k_{i+1}}, g_{k_i})} \le 2J^*(\frac{1}{T_{k_{i+1}}} - \frac{1}{T_{k_i}})$. Therefore,

$$
\begin{aligned}
\int_{\mathcal{X}} \left|g_{k_N}(\mathbf{x}) - g_{k+1}(\mathbf{x})\right| \nu(d\mathbf{x}) &\le \sum_{i=0}^{N-1} 2J^*\left(\frac{1}{T_{k_{i+1}}} - \frac{1}{T_{k_i}}\right) \\
&= 2J^*\left(\frac{1}{T_{k_N}} - \frac{1}{T_{k+1}}\right). \qquad (4.62)
\end{aligned}
$$

We now use (4.62) to bound $\|k^{\frac{\alpha+\beta}{2}} W_{1,k}\|$:

$$
\begin{aligned}
&\left\|k^{\frac{\alpha+\beta}{2}} W_{1,k}\right\| \\
&\quad = \left\|k^{\frac{\alpha+\beta}{2}}\left(E_{g_{k+1}}\left[\Upsilon(\mathbf{X})\right] - E_{g^*}\left[\Upsilon(\mathbf{X})\right]\right)\right\| \\
&\quad = k^{\frac{\alpha+\beta}{2}} \left\|E_{g_{k+1}}\left[\Upsilon(\mathbf{X})\right] - \lim_{N\to\infty} E_{g_{k_N}}\left[\Upsilon(\mathbf{X})\right]\right\| \quad \text{by Lemma 4.22} \\
&\quad \le k^{\frac{\alpha+\beta}{2}} \lim_{N\to\infty} \int_{\mathcal{X}} \left\|\Upsilon(\mathbf{x})\right\| \left|g_{k_N}(\mathbf{x}) - g_{k+1}(\mathbf{x})\right| \nu(d\mathbf{x}) \\
&\quad \le Ck^{\frac{\alpha+\beta}{2}} \lim_{N\to\infty} \int_{\mathcal{X}} \left|g_{k_N}(\mathbf{x}) - g_{k+1}(\mathbf{x})\right| \nu(d\mathbf{x}) \quad \text{where } \left\|\Upsilon(\mathbf{X})\right\| \le C \\
&\quad \le 2J^* Ck^{\frac{\alpha+\beta}{2}}\left(\frac{1}{T^*} - \frac{1}{T_{k+1}}\right),
\end{aligned}
$$

which approaches zero as $k \to \infty$ by Assumption D1. □

Lemma 4.27 *Assume Assumptions C1–C4, D1, and D2 hold, and $\beta > \alpha$, then*

$$
k^{\frac{\alpha+\beta}{2}} W_{2,k} \to 0 \quad \text{as } k \to \infty \text{ w.p.1},
$$

where the limit is component-wise.

Proof By taking conditional expectations on both sides of (4.60), we can bound $\|W_{2,k}\|$ by terms that are similar to those in the proof of Lemma 4.25. Next, invoking the a.s. convergence of the sequence $\{\hat{\theta}_k\}$, an argument similar to the proof of Lemma 4.25 implies that all bounding terms of $\|W_{2,k}\|$ are on the order of $O(N_k^{-1})$, independent of T_k. Therefore, $k^{\frac{\alpha+\beta}{2}} W_{2,k}$ approaches zero as $k \to \infty$ by taking $N_k = \Theta(k^\beta)$ with $\beta > \alpha$. □

We have the following asymptotic convergence rate result for MARS$_1$. Its proof follows from Lemmas 4.25, 4.26, and 4.27 above, and then by applying Theorem 2.2 in [58].

Theorem 4.28 *Let $\alpha_k = c/k^\alpha$ and $N_k = \Theta(k^\beta)$ for constants $c > 0$, $\alpha \in (\frac{1}{2}, 1)$, and $\beta > \alpha$. Assume Assumptions C1–C4, D1, and D2 hold. Then*

$$k^{\frac{\alpha+\beta}{2}}\left(m(\hat{\theta}_k) - \Upsilon^*\right) \xrightarrow{\text{dist}} \mathcal{N}(0, \Sigma) \quad \text{as } k \to \infty,$$

where $\Sigma = \kappa \widehat{\text{Cov}}_{m^{-1}(\Upsilon^*)}[(\Upsilon(\mathbf{X}) - E_{g^*}[\Upsilon(\mathbf{X})])g^*(\mathbf{X})/\hat{f}(\mathbf{X}, m^{-1}(\Upsilon^*))]$ for some constant $\kappa > 0$.

It is interesting to note that in contrast to the optimal asymptotic rate of $O(1/\sqrt{k})$ for general stochastic approximation algorithms, Theorem 4.28 states that the asymptotic rate of convergence for MARS$_1$ is at least $O(1/\sqrt{k})$ (i.e., when the values of α and β are chosen close to 1/2). Moreover, this rate can be made arbitrarily fast by using a sample size sequence $\{N_k\}$ that increases sufficiently fast as $k \to \infty$. However, increasing sample sizes too fast may have a negative impact on the algorithm's practical performance, as the normality result is expressed in terms of the number of algorithm iterations, not the sample size. Therefore, there is a trade-off between the need for large values of β to increase the algorithm's (asymptotic) convergence speed and the desirability of using small values of β to reduce the per iteration computational cost. Also note that the result of Theorem 4.28 depends on the asymptotic approximation quality of the sequence of surrogate distributions. To illustrate this, we proceed with a heuristic argument and assume that the second order moment of $k^{\frac{\alpha+\beta}{2}}(m(\hat{\theta}_k) - \Gamma^*)$ coincides with that of the limiting normal distribution. Then, for k sufficiently large, we have from Theorem 4.28 that the amplified mean squared error of MARS$_1$

$$\begin{aligned}
&k^{\alpha+\beta} E\left[\left\|m(\hat{\theta}_k) - \Upsilon^*\right\|^2\right] \\
&\quad \approx \kappa \sum_{i=1}^{d} \hat{E}_{m^{-1}(\Upsilon^*)}\left[\left(\Upsilon_i(\mathbf{X}) - E_{g^*}\left[\Upsilon_i(\mathbf{X})\right]\right)^2 \left(g^*(\mathbf{X})/\hat{f}\left(\mathbf{X}, m^{-1}\left(\Upsilon^*\right)\right)\right)^2\right] \\
&\quad = \kappa \sum_{i=1}^{d} E_{g^*}\left[\left(\Upsilon_i(\mathbf{X}) - E_{g^*}\left[\Upsilon_i(\mathbf{X})\right]\right)^2 g^*(\mathbf{X})/\hat{f}\left(\mathbf{x}, m^{-1}\left(\Upsilon^*\right)\right)\right],
\end{aligned}$$

which is (approximately) proportional to the ratio $g^*(\mathbf{x})/\hat{f}(\mathbf{x}, m^{-1}(\Gamma^*))$ in a small neighborhood of the global optimum. This indicates that if $\hat{f}(\cdot, m^{-1}(\Gamma^*))$ can closely approximate g^* in the vicinity of $\mathbf{x}^*$, then a better (asymptotic) mean squared error can be attained.

4.6.2 Application of MARS to Finite-Horizon MDPs

We modify and extend the MARS algorithm to a stochastic setting and present a simulation-based algorithm called approximate stochastic annealing (ASA) for solving finite horizon MDPs. At each iteration, the ASA algorithm estimates the optimal policy by sampling from a probability distribution function over the policy space. The distribution is then modified using a Boltzmann selection scheme based on the simulated/estimated value functions of the sampled policies. This idea of working with a probability distribution over the policy space is in the same spirit as that of the MRAS approach presented in Sect. 4.3. However, central to MRAS (and many other model-based algorithms such as EDAs and CE) is a selection step to concentrate the search on promising regions of the policy space, which frequently requires a quantile estimation of the (unknown) distribution of the value function estimates. MARS eliminates the need for such a selection step by implicitly sampling from a sequence of convergent Boltzmann distributions that assigns more weight to high quality policies as the sampling process proceeds. ASA also shares similarities with the learning automata approach for stochastic optimization [146] and the PLA sampling algorithm of Chap. 2, which iteratively update a probability distribution over the feasible region in a direction that pursues the current estimated optimal solution. From this perspective, ASA can be viewed as a generalized pursuit scheme that pursues a population of policies by assigning each policy a weight proportional to its current reward estimate in updating distribution functions.

We recall the discrete-time finite H-horizon MDP defined in Chap. 1 with system dynamics $x_{t+1} = f(x_t, a_t, w_t)$ for $t = 0, 1, \ldots, H-1$, where x_t represents the state of the system at time t taking values from a finite state space X, a_t is the control at time t chosen from a finite action set A, $\{w_t \in W \ \forall t\}$ is a sequence of independent (not necessarily identically distributed) random vectors representing the stochastic uncertainty of the system, and f is the next-state transition function describing the system dynamics. Let $R' : X \times A \times W \to \Re^+ \cup \{0\}$ be a non-negative one-stage reward function associated with the system. We assume that the reward function R' satisfies $R_{\max} := \sup_{x \in X, a \in A, w \in W} R'(x, a, w) < \infty$. Define Π as the set of non-stationary non-randomized Markovian policies $\pi = \{\pi_t, \ t = 0, \ldots, H-1\}$, where each $\pi_t : X \to A$ is a function that specifies the action to be applied at time t for each $x \in X$. For an initial state $x_0 = x$, the expected total reward (value function) associated with a policy π is given by

$$V^{\pi}(x) := E_w \left[\sum_{t=0}^{H-1} R'\big(x_t, \pi_t(x_t), w_t\big) \middle| x_0 = x \right], \tag{4.63}$$

where E_w is understood with respect to the joint distribution of $w_0, \ldots, w_{H-1}$. The objective is to identify an optimal policy $\pi^* \in \Pi$ that maximizes the expected total reward for a given state x, i.e.,

$$V^{\pi^*}(x) = \sup_{\pi \in \Pi} V^{\pi}(x). \tag{4.64}$$

Throughout this section, we assume that the optimal policy π^* is unique.

In ASA, candidate policies are constructed at each iteration k by using a $|X|$-by-$|A|$-by-H stochastic matrix q_k, whose (i, j, t)th entry $q_k(i, j, t)$ specifies the probability that the ith state x_i takes action a_j at time t. Note that every such stochastic matrix q_k induces a probability mass function over the policy space Π:

$$\phi(q_k, \pi) := \prod_{t=0}^{H-1} \prod_{i=1}^{|X|} \prod_{j=1}^{|A|} \left[q_k(i, j, t)\right]^{I\{\pi \in \Pi_{i,j}(t)\}} \quad \forall \pi \in \Pi,$$

where $\Pi_{i,j}(t) := \{\pi : \pi_t(x_i) = a_j\}$ denotes the set of non-randomized policies that assign action a_j to state x_i at time t.

The proposed ASA algorithm is presented in Fig. 4.17. There are two allocation rules in ASA. The first is the sample size sequence $\{N_k\}$, which specifies the number of policies to be generated at each iteration. Note that not all N_k policies are sampled from $\phi(q_k, \pi)$; instead, they are drawn from the mixture of $\phi(q_k, \pi)$ and the initial distribution $\phi(q_0, \pi)$, where the mixing intensity of the two distributions is determined by a parameter β_k. Another allocation rule is $\{M_k\}$, which allocates M_k simulation replications to each policy generated at every step. In particular, each time a given policy π is simulated, a sample path (simulation realization) $\omega_i := \{x_0, \pi_0(x_0), w_0, \ldots, x_{H-1}, \pi_{H-1}(x_{H-1}), w_{H-1}\}$ is observed, and an estimate of the value function (4.63) based on ω_i can be obtained as follows:

$$V_i^{\pi} := \sum_{t=0}^{H-1} R\big(x_t, \pi_t(x_t), w_t\big).$$

In (4.66), the current probability matrix is updated using a Boltzmann selection scheme, where each π in the current population of sampled policies Λ_k is weighted by an empirical Boltzmann mass function

$$\bar{g}_{k+1}(\pi) = \frac{e^{\bar{V}_k^{\pi}/T_{k+1}} \hat{\phi}^{-1}(q_k, \pi)}{\sum_{\pi' \in \Lambda_k} e^{\bar{V}_k^{\pi'}/T_{k+1}} \hat{\phi}^{-1}(q_k, \pi')} \quad \forall \pi \in \Lambda_k,$$

which becomes more concentrated on promising policies in Λ_k as the parameter T_k decreases to zero. Note that the division by $\hat{\phi}$ in $\bar{g}_{k+1}$ is used to compensate for policies that are unlikely to be chosen, which makes $\bar{g}_k$ a good approximation of the true Boltzmann mass function

$$g_{k+1}(\pi) = \frac{e^{V^{\pi}/T_{k+1}}}{\sum_{\pi' \in \Pi} e^{V^{\pi'}/T_{k+1}}} \quad \forall \pi \in \Pi. \tag{4.65}$$

Approximate Stochastic Annealing

Input: annealing schedule $\{T_k\}$, parameter sequences $\{\alpha_k\}$ and $\{\beta_k\}$ satisfying $0 \leq \alpha_k, \beta_k \leq 1\ \forall k$, a sample size sequence $\{N_k\}$, and a simulation allocation sequence $\{M_k\}$. Set $q_0(i, j, t) = 1/|A|\ \forall i, j, t$.

Initialization: Set iteration count $k = 0$.

Loop until Stopping Rule is satisfied:

- Sample N_k policies $\Lambda_k := \{\pi^1, \pi^2, \ldots, \pi^{N_k}\}$ from $\hat{\phi}(q_k, \pi) = (1 - \beta_k)\phi(q_k, \pi) + \beta_k \phi(q_0, \pi)$ as follows: for each $i = 1, \ldots, N_k$,
  - with probability $1 - \beta_k$, construct a policy π^i using the stochastic matrix q_k;
  - with probability β_k, construct π^i using q_0.
- Perform M_k independent simulation replication runs for each $\pi \in \Lambda_k$ and let $\bar{V}_k^{\pi} = M_k^{-1} \sum_{l=1}^{M_k} V_j^{\pi}$.
- Update matrix q_k by

$$q_{k+1}(i, j, t) = \alpha_k \frac{\sum_{\pi \in \Lambda_k} e^{\bar{V}_k^{\pi}/T_{k+1}} \hat{\phi}^{-1}(q_k, \pi) I\{\pi \in \Pi_{i,j}(t)\}}{\sum_{\pi \in \Lambda_k} e^{\bar{V}_k^{\pi}/T_{k+1}} \hat{\phi}^{-1}(q_k, \pi)} + (1 - \alpha_k) q_k(i, j, t). \tag{4.66}$$

- $k \leftarrow k + 1$.

Output: q_k.

Fig. 4.17 Description of ASA algorithm

In addition, the smoothing parameter α_k used in (4.66) ensures that the difference between q_{k+1} and q_k is incremental, so that the new distribution $\phi(q_{k+1}, \pi)$ does not deviate significantly from the current distribution $\phi(q_k, \pi)$.

Equation (4.66) can be written in the form of a standard stochastic approximation recursion as follows:

$$q_{k+1}(i, j, t) - q_k(i, j, t) = \alpha_k \left[\sum_{\pi \in \Lambda_k} \bar{g}_{k+1}(\pi) I\left\{\pi \in \Pi_{i,j}(t)\right\} - q_k(i, j, t) \right]. \tag{4.67}$$

Note that the above equation shares certain similarities with the learning automata approach [146]. Since $\bar{g}_{k+1}$ gives more weight to policies with better performance, it is easy to see from (4.67) that the stochastic matrix q_{k+1} is updated in the averaged direction of (estimated) promising policies in Λ_k. In other words, the algorithm pursues at each iteration all promising policies in the current population Λ_k, generalizing the learning automata algorithm of [146], which pursues only the current estimated best policy.

4.6.2.1 Convergence Analysis

We define $\mathcal{F}_k$ as the increasing σ-fields generated by the set of all sampled policies and random generation of simulation sample paths up to iteration $k-1$. The conditional probability and expectation $P(\cdot|\mathcal{F}_k)$ and $E[\cdot|\mathcal{F}_k]$ are to be understood with respect to $\hat{\phi}(q_k, \pi)$ and the joint distribution of stochastic generation of simulation sample paths $w^k = \{w_0^k, \ldots, w_{H-1}^k\}$ at the kth iteration. To simplify exposition, the following shorthand notations are used throughout this section:

$$
\begin{aligned}
Y_k &= \sum_{\Pi} e^{V^\pi/T_{k+1}} I\{\pi \in \Pi_{i,j}(t)\}, \\
Z_k &= \sum_{\Pi} e^{V^\pi/T_{k+1}}, \\
\hat{Y}_k &= N_k^{-1} \sum_{\Lambda_k} e^{V^\pi/T_{k+1}} \hat{\phi}^{-1}(q_k, \pi) I\{\pi \in \Pi_{i,j}(t)\}, \\
\hat{Z}_k &= N_k^{-1} \sum_{\Lambda_k} e^{V^\pi/T_{k+1}} \hat{\phi}^{-1}(q_k, \pi), \\
\bar{Y}_k &= N_k^{-1} \sum_{\Lambda_k} e^{\bar{V}_k^\pi/T_{k+1}} \hat{\phi}^{-1}(q_k, \pi) I\{\pi \in \Pi_{i,j}(t)\}, \\
\bar{Z}_k &= N_k^{-1} \sum_{\Lambda_k} e^{\bar{V}_k^\pi/T_{k+1}} \hat{\phi}^{-1}(q_k, \pi).
\end{aligned}
$$

To analyze ASA, we rewrite (4.67) in the following recursive form:

$$
\eta_{k+1} = \eta_k - \xi_k, \tag{4.68}
$$

where $\eta_k := q_k(i, j, t) - I\{\pi^* \in \Pi_{i,j}(t)\}$ and

$$
\begin{aligned}
\xi_k &= \alpha_k \left[\eta_k + I\{\pi^* \in \Pi_{i,j}(t)\} - \frac{\bar{Y}_k}{\bar{Z}_k} \right] \\
&= \alpha_k \left[\eta_k + I\{\pi^* \in \Pi_{i,j}(t)\} - \frac{\hat{Y}_k}{\hat{Z}_k} + \frac{\hat{Y}_k}{\hat{Z}_k} - \frac{\bar{Y}_k}{\bar{Z}_k} \right].
\end{aligned}
$$

We first show that the sequence of idealized Boltzmann distribution $\{g_k\}$ converges to a limiting distribution that concentrates only on the optimal policy π^*.

Lemma 4.29 *If $T_k \to 0$ as $k \to \infty$, then $E_{g_k}[I\{\pi \in \Pi_{i,j}(t)\}] \to I\{\pi^* \in \Pi_{i,j}(t)\}$ as $k \to \infty$.*

Proof Note that

$$
\left| E_{g_k}\left[I\{\pi \in \Pi_{i,j}(t)\} \right] - I\{\pi^* \in \Pi_{i,j}(t)\} \right|
$$

$$
\begin{aligned}
&\leq E_{g_k}\left[\left|I\left\{\pi \in \Pi_{i,j}(t)\right\} - I\left\{\pi^* \in \Pi_{i,j}(t)\right\}\right|\right] \\
&= \sum_{\pi \neq \pi^*} \left|I\left\{\pi \in \Pi_{i,j}(t)\right\} - I\left\{\pi^* \in \Pi_{i,j}(t)\right\}\right| \frac{e^{V^\pi/T_k}}{\sum_{\pi' \in \Pi} e^{V^{\pi'}/T_k}} \\
&\leq \frac{\sum_{\pi \neq \pi^*} e^{(V^\pi - V^{\pi^*})/T_k}}{1 + \sum_{\pi' \neq \pi^*} e^{(V^{\pi'} - V^{\pi^*})/T_k}} \\
&\leq \sum_{\pi \neq \pi^*} e^{(V^\pi - V^{\pi^*})/T_k},
\end{aligned}
$$

which approaches zero as $T_k \to 0$, since $|\Pi|$ is finite and $V^{\pi^*} > V^\pi \ \forall \pi \neq \pi^*$. □

Note that Lemma 4.29 implies the existence of a small $T^* > 0$ such that the Boltzmann distribution g^* parameterized by T^* can be arbitrarily close to the degenerated optimal distribution.

The next lemma states that $\frac{\hat{Y}_k}{\hat{Z}_k}$ is an asymptotically unbiased estimator of $\frac{Y_k}{Z_k}$.

Lemma 4.30 *If $N_k \beta_k \to \infty$ as $k \to \infty$, then*

$$
E\left[\frac{\hat{Y}_k}{\hat{Z}_k}\middle|\mathcal{F}_k\right] \to \frac{Y_k}{Z_k} \quad \text{as } k \to \infty \ w.p.1.
$$

Proof

$$
\begin{aligned}
\frac{\hat{Y}_k}{\hat{Z}_k} - \frac{Y_k}{Z_k} &= \frac{\hat{Y}_k}{\hat{Z}_k} - \frac{\hat{Y}_k}{Z_k} + \frac{\hat{Y}_k}{Z_k} - \frac{Y_k}{Z_k} \\
&= \frac{\hat{Y}_k}{\hat{Z}_k}\frac{Z_k - \hat{Z}_k}{Z_k} + \frac{1}{Z_k}(\hat{Y}_k - Y_k).
\end{aligned}
$$

Taking conditional expectations at both sides yields

$$
\begin{aligned}
\left|E\left[\frac{\hat{Y}_k}{\hat{Z}_k}\middle|\mathcal{F}_k\right] - \frac{Y_k}{Z_k}\right| &= \left|E\left[\frac{\hat{Y}_k}{\hat{Z}_k}\frac{Z_k - \hat{Z}_k}{Z_k}\middle|\mathcal{F}_k\right]\right| \\
&\leq \frac{1}{|Z_k|}E\left[|Z_k - \hat{Z}_k|\middle|\mathcal{F}_k\right] \quad \text{since } \left|\frac{\hat{Y}_k}{\hat{Z}_k}\right| \leq 1 \\
&\leq \frac{1}{|Z_k|}E\left[(Z_k - \hat{Z}_k)^2\middle|\mathcal{F}_k\right]^{1/2} \quad \text{by Hölder's inequality} \\
&\leq \frac{1}{|Z_k|}\frac{1}{\sqrt{N_k}}E\left[e^{2V^\pi/T_{k+1}}\hat{\phi}^{-2}(q_k, \pi)|\mathcal{F}_k\right]^{1/2}
\end{aligned}
$$

$$= \frac{1}{\sqrt{N_k}} \left(\sum_{\Pi} \hat{\phi}^{-1}(q_k, \pi) \frac{e^{V^\pi / T_{k+1}}}{Z_k} \frac{e^{V^\pi / T_{k+1}}}{Z_k} \right)^{1/2}$$

$$\leq \frac{1}{\sqrt{N_k}} \left(\sum_{\Pi} \beta_k^{-1} \phi^{-1}(q_0, \pi) g_{k+1}(\pi) \right)^{1/2}$$

$$\leq \frac{|A|^{|X|H/2}}{\sqrt{N_k \beta_k}}.$$

Thus, the desired result follows by taking $N_k \beta_k \to \infty$. □

Moreover, under some appropriate conditions on $\{T_k\}$ and $\{M_k\}$, the conditional expectation of the error term $\frac{\hat{Y}_k}{\hat{Z}_k} - \frac{\bar{Y}_k}{\bar{Z}_k}$ caused by simulation noise vanishes asymptotically.

Lemma 4.31 *If the simulation allocation rule and the annealing schedule satisfy* $M_k T_{k+1}^2 - \frac{1}{T_{k+1}} \to \infty$ *as* $k \to \infty$, *then*

$$E\left[\left| \frac{\hat{Y}_k}{\hat{Z}_k} - \frac{\bar{Y}_k}{\bar{Z}_k} \right| \middle| \mathcal{F}_k \right] \to 0 \quad \text{as } k \to \infty \text{ w.p.1.}$$

Proof Let $\hat{g}_{k+1}(\pi) = \frac{e^{V^\pi / T_{k+1}} \hat{\phi}^{-1}(q_k, \pi)}{\sum_{\pi' \in \Lambda_k} e^{V^{\pi'} / T_{k+1}} \hat{\phi}^{-1}(q_k, \pi')}$ $\forall \pi \in \Lambda_k$. We have

$$\left| \frac{\hat{Y}_k}{\hat{Z}_k} - \frac{\bar{Y}_k}{\bar{Z}_k} \right| = \left| \frac{1}{\hat{Z}_k} (\hat{Y}_k - \bar{Y}_k) + \frac{\bar{Y}_k}{\bar{Z}_k} \frac{\bar{Z}_k - \hat{Z}_k}{\hat{Z}_k} \right|$$

$$\leq \frac{1}{|\hat{Z}_k|} \left(|\hat{Y}_k - \bar{Y}_k| + |\bar{Z}_k - \hat{Z}_k| \right) \quad \text{since } \left| \frac{\bar{Y}_k}{\bar{Z}_k} \right| \leq 1$$

$$\leq \frac{1}{|\hat{Z}_k|} \frac{2}{N_k} \sum_{\Lambda_k} \left| e^{V^\pi / T_{k+1}} - e^{\bar{V}_k^\pi / T_{k+1}} \right| \hat{\phi}^{-1}(q_k, \pi)$$

$$= \frac{1}{|\hat{Z}_k|} \frac{2}{N_k} \sum_{\Lambda_k} e^{V^\pi / T_{k+1}} \left| 1 - e^{\frac{\bar{V}_k^\pi - V^\pi}{T_{k+1}}} \right| \hat{\phi}^{-1}(q_k, \pi)$$

$$= 2 \sum_{\Lambda_k} \hat{g}_{k+1}(\pi) \left| 1 - e^{\frac{\bar{V}_k^\pi - V^\pi}{T_{k+1}}} \right|.$$

Therefore,

$$E\left[\left| \frac{\hat{Y}_k}{\hat{Z}_k} - \frac{\bar{Y}_k}{\bar{Z}_k} \right| \middle| \mathcal{F}_k \right] \leq 2E\left[\sum_{\Lambda_k} \hat{g}_{k+1}(\pi) \left| 1 - e^{\frac{\bar{V}_k^\pi - V^\pi}{T_{k+1}}} \right| \middle| \mathcal{F}_k \right]$$

$$= 2E\left[E\left[\sum_{\Lambda_k}\hat{g}_{k+1}(\pi)\left|1-e^{(\bar{V}_k^\pi - V^\pi)/T_{k+1}}\right|\middle|\Lambda_k\right]\middle|\mathcal{F}_k\right]$$

$$= 2E\left[\sum_{\Lambda_k}\hat{g}_{k+1}(\pi)E\left[\left|1-e^{(\bar{V}_k^\pi - V^\pi)/T_{k+1}}\right|\middle|\Lambda_k\right]\middle|\mathcal{F}_k\right].$$

Note that Hölder's inequality implies that

$$\begin{aligned}E\left[\left|1-e^{(\bar{V}_k^\pi - V^\pi)/T_{k+1}}\right|\middle|\Lambda_k\right] &\le E\left[\left(1-e^{\frac{\bar{V}_k^\pi - V^\pi}{T_{k+1}}}\right)^2\middle|\Lambda_k\right]^{1/2}\\ &= \left[1-2E\left[e^{\frac{(\bar{V}_k^\pi - V^\pi)}{T_{k+1}}}\middle|\Lambda_k\right]+E\left[e^{\frac{2(\bar{V}_k^\pi - V^\pi)}{T_{k+1}}}\middle|\Lambda_k\right]\right]^{1/2}\\ &\le \left[E\left[e^{\frac{2(\bar{V}_k^\pi - V^\pi)}{T_{k+1}}}\middle|\Lambda_k\right]-1\right]^{1/2}, \end{aligned} \tag{4.69}$$

where the last step follows from Jensen's inequality, since

$$E\left[e^{(\bar{V}_k^\pi - V^\pi)/T_{k+1}}\middle|\Lambda_k\right] \ge e^{E[(\bar{V}_k^\pi - V^\pi)/T_{k+1}|\Lambda_k]} = 1.$$

Since the one-stage reward function is uniformly bounded by $R_{\max}$, it is clear that the difference $|\bar{V}_k^\pi - V^\pi|$ is bounded by $L := HR_{\max}$. For a given $\pi \in \Pi$, let $\mathcal{A}_k = \{|\bar{V}_k^\pi - V^\pi| \le T_{k+1}C\}$ and $\mathcal{A}_k^c$ be the complement of $\mathcal{A}_k$, where $C := L\sqrt{L}$. It follows that

$$\begin{aligned}E\left[e^{\frac{2(\bar{V}_k^\pi - V^\pi)}{T_{k+1}}}\middle|\Lambda_k\right] &= E\left[e^{\frac{2(\bar{V}_k^\pi - V^\pi)}{T_{k+1}}}I\{\mathcal{A}_k\}\middle|\Lambda_k\right] \quad \text{[i]}\\ &\quad + E\left[e^{\frac{2(\bar{V}_k^\pi - V^\pi)}{T_{k+1}}}I\left\{\mathcal{A}_k^c\right\}\middle|\Lambda_k\right]. \quad \text{[ii]}\end{aligned}$$

We now analyze terms [i] and [ii].

$$\begin{aligned}\text{[i]} &\le E\left[1+\frac{2(\bar{V}_k^\pi - V^\pi)}{T_{k+1}}+\frac{e^{2C}-2C-1}{C^2}\frac{(\bar{V}_k^\pi - V^\pi)^2}{T_{k+1}^2}\middle|\Lambda_k\right]\\ &= 1+\frac{e^{2C}-2C-1}{C^2T_{k+1}^2}\frac{1}{M_k}\mathrm{Var}\left(V_i^\pi\middle|\Lambda_k\right)\\ &\le 1+\frac{(e^{2C}-2C-1)L^2}{4C^2}\frac{1}{M_kT_{k+1}^2}, \end{aligned} \tag{4.70}$$

where the first inequality follows from the fact that $e^x \le 1+x+\frac{(e^b-b-1)}{b^2}x^2$ for $|x| \le b$ and the last inequality follows because $\mathrm{Var}(X) \le L^2/4$ for any bounded non-negative random variable $0 \le X \le L$.

Regarding term [ii], we have

$$\text{[ii]} \le e^{\frac{2L}{T_{k+1}}}P\left(\mathcal{A}_k^c\middle|\Lambda_k\right)$$

$$
\begin{aligned}
&= e^{\frac{2L}{T_{k+1}}} P\left(\left|\bar{V}^{\pi_k} - V^{\pi}\right| \geq T_{k+1} C \middle| \Lambda_k\right) \\
&\leq 2e^{\frac{2L}{T_{k+1}}} e^{\frac{-2M_k T_{k+1}^2 C^2}{L^2}} \quad \text{by Hoeffding's inequality} \\
&= 2e^{2L\left(\frac{1}{T_{k+1}} - M_k T_{k+1}^2\right)}. \qquad (4.71)
\end{aligned}
$$

Combining (4.70) and (4.71), it is easy to observe that (4.69) is on the order of $O(e^{L(T_{k+1}^{-1} - M_k T_{k+1}^2)})$. This further implies that

$$
E\left[\left|\frac{\hat{Y}_k}{\hat{Z}_k} - \frac{\bar{Y}_k}{\bar{Z}_k}\right| \middle| \mathcal{F}_k\right] = O\left(e^{L(T_{k+1}^{-1} - M_k T_{k+1}^2)}\right),
$$

which goes to zero as $M_k T_{k+1}^2 - T_{k+1}^{-1} \to \infty$. □

We have the following convergence theorem for ASA, which implies that the sequence of stochastic matrices q_k generated at successive iterations of the algorithm will converge to a limiting matrix that assigns unit mass to the optimal policy π^*.

Theorem 4.32 *Assume that the following conditions hold*:

(a) $\alpha_k > 0\ \forall k$, $\sum_{k=0}^{\infty} \alpha_k = \infty$, *and* $\sum_{k=0}^{\infty} \alpha_k^2 < \infty$;
(b) $T_k \to 0$ *as* $k \to \infty$;
(c) $N_k \beta_k \to \infty$ *as* $k \to \infty$;
(d) $M_k T_{k+1}^2 - T_{k+1}^{-1} \to \infty$ *as* $k \to \infty$.

Then

$$
q_k(i, j, t) \to I\left\{\pi^* \in \Pi_{i,j}(t)\right\} \quad \forall i, j, t \text{ as } k \to \infty \ w.p.1.
$$

Proof We consider recursion (4.68) and define $U_k = E[\xi_k | \mathcal{F}_k]$ and $\Delta_k = \xi_k - U_k$. To show the desired result, we establish that conditions (i)–(iv) in [56] hold.

(i) First we show that for any $\epsilon > 0$, $P(|\eta_k| > \epsilon, \eta_k U_k < 0 \text{ i.o.}) = 0$. We have

$$
\begin{aligned}
U_k = \alpha_k \Bigg[&\eta_k + I\left\{\pi^* \in \Pi_{i,j}(t)\right\} - \frac{Y_k}{Z_k} + \frac{Y_k}{Z_k} - E\left[\frac{\hat{Y}_k}{\hat{Z}_k} \middle| \mathcal{F}_k\right] \\
&+ E\left[\frac{\hat{Y}_k}{\hat{Z}_k} - \frac{\bar{Y}_k}{\bar{Z}_k} \middle| \mathcal{F}_k\right]\Bigg]. \qquad (4.72)
\end{aligned}
$$

Therefore,

$$
\begin{aligned}
\eta_k U_k = \alpha_k \Bigg[&\eta_k^2 + \eta_k \left(I\left\{\pi^* \in \Pi_{i,j}(t)\right\} - \frac{Y_k}{Z_k} \right) \\
&+ \eta_k \left(\frac{Y_k}{Z_k} - E\left[\frac{\hat{Y}_k}{\hat{Z}_k} \middle| \mathcal{F}_k\right] \right) + \eta_k \left(E\left[\frac{\hat{Y}_k}{\hat{Z}_k} - \frac{\bar{Y}_k}{\bar{Z}_k} \middle| \mathcal{F}_k\right] \right) \Bigg].
\end{aligned}
$$

Since η_k is bounded, the second term above vanishes to zero by Lemma 4.29, whereas the third and last terms both go to zero w.p.1 by Lemma 4.30 and Lemma 4.31. Thus, for almost every sample path generated by the algorithm, we have $\eta_k U_k > 0$ whenever $\eta_k > \epsilon$ for k sufficiently large, which implies $P(|\eta_k| > \epsilon, \eta_k U_k < 0 \text{ i.o.}) = 0$.

(ii) Since all terms in the square bracket of (4.72) are bounded and $\alpha_k \to 0$ by condition (a), we have $|U_k|(1 + |\eta_k|)^{-1} \to 0$ as $k \to \infty$.

(iii) By definition, we have

$$\Delta_k = \alpha_k \left[E\left[\frac{\bar{Y}_k}{\bar{Z}_k} \middle| \mathcal{F}_k \right] - \frac{\bar{Y}_k}{\bar{Z}_k} \right].$$

It follows that

$$\sum_{k=1}^{\infty} E\left[|\Delta_k|^2\right] = \sum_{k=1}^{\infty} \alpha_k^2 E\left[\left(E\left[\frac{\bar{Y}_k}{\bar{Z}_k} \middle| \mathcal{F}_k \right] - \frac{\bar{Y}_k}{\bar{Z}_k} \right)^2 \right] < \infty,$$

since $\frac{\bar{Y}_k}{\bar{Z}_k}$ is bounded and $\sum_k \alpha_k^2 < \infty$ by condition (a).

(iv) Finally, we show that

$$P\left(\liminf_{k \to \infty} |\eta_k| > 0, \sum_{k=1}^{\infty} |U_k| < \infty \right) = 0.$$

From (4.72), we have

$$|U_k| \geq \alpha_k \left[|\eta_k| - \left| I\left\{ \pi^* \in \Pi_{i,j}(t) \right\} - \frac{Y_k}{Z_k} \right| \right.$$
$$\left. - \left| \frac{Y_k}{Z_k} - E\left[\frac{\hat{Y}_k}{\hat{Z}_k} \middle| \mathcal{F}_k \right] \right| - E\left[\left| \frac{\hat{Y}_k}{\hat{Z}_k} - \frac{\bar{Y}_k}{\bar{Z}_k} \right| \middle| \mathcal{F}_k \right] \right].$$

Let $\Omega_1 = \{\liminf_k |\eta_k| > 0\}$. For every sample path $\omega \in \Omega_1$, there exists an ω-dependent constant $\delta > 0$ such that $\liminf_k |\eta_k| > \delta$. Consequently, by Lemmas 4.29, 4.30, and 4.31, for almost every $\omega \in \Omega_1$, we have $|U_k| \leq \alpha_k \delta/2$ for k sufficiently large. Therefore by condition (a), it follows that for almost all $\omega \in \Omega_1$,

$$\sum_{k=1}^{\infty} |U_k| \geq \sum_{k=K(\omega)}^{\infty} |U_k| \geq \frac{\delta}{2} \sum_{k=K(\omega)}^{\infty} \alpha_k = \infty,$$

which establishes condition (iv).

Finally, combining (i)–(iv) and applying the main theorem in [56] yield $\eta_k \to 0$ as $k \to \infty$ w.p.1. □

4.6.2.2 A Numerical Example

To illustrate the performance of ASA, we again consider the finite H-horizon inventory control problem of Sect. 2.1.5. At each time t, given the current inventory level x_t, an order in the amount of a_t is placed and received immediately, and a demand D_t is realized. Assume that the inventory has a finite capacity 20, x_t takes values from the set of non-negative integers between 0 and 20, the order amount $a_t \in \{0, 2, 4, 6, 8, 10\}$ for all $t = 0, \ldots, H-1$, and the demands D_t are i.i.d. discrete random variables with the discrete uniformly distribution $DU(0, 9)$. The inventory level evolves according to the following recursion: $x_{t+1} = (x_t + a_t - D_t)^+$, where any a_t that makes the current inventory position (i.e., $x_t + a_t$) exceed the inventory capacity is considered inadmissible. For a given initial inventory level x_0, the objective is to minimize the expected total cost over the set of all non-stationary Markovian policies, i.e.,

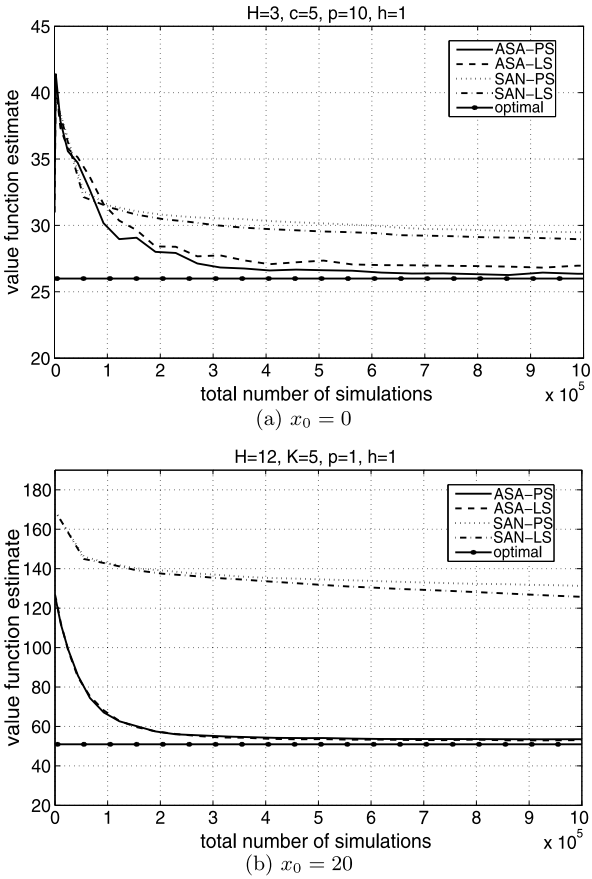
$$\min_{\pi \in \Pi} E\left[\sum_{t=0}^{H-1} \left(K I\{\pi_t(x_t) > 0\} + h\big(x_t + \pi_t(x_t) - D_t\big)^+ + p\big(D_t - x_t - \pi_t(x_t)\big)^+\right) \middle| x_0 = x\right],$$

where K is the fixed set-up cost per order, h is the per unit inventory holding cost, and p is the per unit demand penalty cost for unsatisfied demands.

The following two sets of parameters are used in our simulation experiments: (a) horizon $H = 3$, set-up cost $K = 5$, penalty cost $p = 10$, holding cost $h = 1$; (b) horizon $H = 12$, set-up cost $K = 5$, penalty cost $p = 1$, holding cost $h = 1$.

In our experiments, we have used two different annealing schedules in ASA: a polynomial schedule (PS) $T_k = 10^{-5} + 1/\sqrt{k}$, and a logarithmic schedule (LS) $T_k = 1/\ln(k+1)$. The other parameters are as follows: $\beta_k = 1/\sqrt{k}$, $N_k = \max\{20, \lfloor k^{0.501} \rfloor\}$, and $M_k = \max\{10, \lfloor 1.01 \ln^3 k \rfloor\}$, where $\lfloor a \rfloor$ is the largest integer no greater than a. As in a typical stochastic approximation algorithm, we found empirically that the performance of ASA is primarily determined by the choice of the gain sequence $\{\alpha_k\}$, but is insensitive to the choices of the above parameters. In all test cases, we have used a relatively conservative gain $\alpha_k = 1/(k + 100)^{0.501}$, where the constant 100 is used to keep initial step sizes small in early iterations of the algorithm to prevent unstable behavior, whereas a slow decay rate 0.501 is used to produce non-negligible step sizes and prevent slow improvement in later iterations. Note that the above parameter settings satisfy the relevant conditions in Theorem 4.32 for convergence.

For the purpose of comparison, we have also applied the simulated annealing (SAN) algorithm to the above test cases, where the basic idea is to interpret an MDP as a stochastic optimization problem over the policy space and then use SAN as a particular optimization technique to search for good policies. Whenever a policy is visited by the algorithm, its value function is estimated by first performing L



(a) $x_0 = 0$

(b) $x_0 = 20$

Fig. 4.18 Average performance of ASA and SAN on an inventory control problem (mean based on 50 independent replication runs), (**a**) $x_0 = 0$; (**b**) $x_0 = 20$

independent simulation replications of that policy and then averaging the simulated value function estimates over L replication runs. In our implementation of SAN, L is set to 100, the same annealing schedules PS and LS as in ASA are used, the neighborhood of a policy π is taken to be $\mathcal{N}(\pi) = \{\pi' \in \Pi : \max_x |\pi_t(x) - \pi'_t(x)| \leq 2, \forall t = 0, \ldots, H-1\}$, and the initial policy is uniformly selected from the set of admissible policies.

For each test case, we performed 50 independent replication runs of both ASA and SAN. The performances of both algorithms are shown in Fig. 4.18, which plots the averaged current best value function estimates as a function of the number of the total simulation runs consumed thus far. Simulation results indicate convergence of ASA and SAN with both annealing schedules, with the performance of ASA consistently dominating that of SAN. Since SAN combines local search, it shows a fast initial improvement, but the algorithm frequently stagnates at solutions/policies that are far from optimal, especially in test case (b) when the horizon length is large.

4.7 Notes

MRAS was introduced in [94], and the presentation in Sects. 4.1 and 4.2 is based on [94] for MRAS_0 and MRAS_1 and [96] for MRAS_2. The term *model-based* methods for optimization is from [192], where such methods are contrasted with traditional *instance-based* methods, whose search for new candidate solutions depends *directly* on previously generated solutions, instead of indirectly through a probability model. Well-known *instance-based* methods include simulated annealing (SAN) [106], genetic algorithms (GAs) [74, 169], tabu search [73], and the nested partitions (NP) method [161, 162]. Other *model-based* methods include ant colony optimization (ACO) [55], the cross-entropy (CE) method [51, 150], probability collectives [187], and the estimation of distribution algorithms (EDAs) [132]; see [69] for a general discussion of such randomized methods. There is a close relationship between MRAS and the CE method; a detailed description of their similarities and differences can be found in [69] and [94]. In addition, the natural exponential family (NEF) [131] plays a prominent role in both methods.

The construction in Eq. (4.3) has previously been used in EDAs with proportional selection schemes [191], and in randomized algorithms for solving Markov decision processes [44] (see also the SAMW algorithm of Sect. 5.1). In those approaches, the construction of the resulting distributions are carried out *explicitly* to generate new samples, whereas in MRAS, they serve as the reference distributions that are projected on the family of parameterized distributions.

There is a long history of using optimization methods to solve control problems with parameterized policies (cf. [6]). Some examples include those discussed in Sects. 1.7 and 3.5, the stochastic approximation-based methods such as [20], as well as the indirect policy search method introduced in [134], which is based on successive estimation of the probability densities that the policy induces on states at different points in time.

In [126], a framework similar to the one presented in Sect. 4.3 but based on the CE method is proposed for solving MDPs, focusing on infinite-horizon MDPs with discrete action spaces. It uses the stochastic matrix $\mathbf{P}$ to simulate random trajectories/sample paths and then iteratively updates the entries of $\mathbf{P}$ based on the performances of these trajectories. In Sect. 4.3, in contrast, marginal distributions are updated by using the performances of the randomly generated (deterministic) policies.

The stochastic approximation framework was proposed in [97]. In particular, the framework has been used to study the convergence of the CE method in [91, 97] by casting a Monte Carlo version of the recursion (4.48) in the form of a generalized Robbins–Monro algorithm, and then following the argument of the ordinary differential equation (ODE) approach [10, 116, 117]. The MARS algorithm was introduced in [92, 93]. Under the equivalent gradient interpretations, a primary difference between MARS and CE is that the gradient in (4.55) is time-varying vs. stationary in CE. Stationarity in general only guarantees local convergence, whereas the time-varying feature of MARS provides a viable way to ensure the algorithm to escape from local minima, leading to global convergence. The presentation of the ASA algorithm in Sect. 4.6.2 is based on [89].

Chapter 5
On-Line Control Methods via Simulation

Policy iteration and value iteration are examples of *off-line* computation of value functions and optimal policies, in which the policy (or the parameters in a parameterized policy) is pre-computed and stored, and is then used to determine which action should be taken when a particular state is observed in the evolution of the system. In this chapter, we consider policies in which the system (either the real system or a simulation) evolves to a particular observed state, and the action to be taken in that particular state is then computed *on-line* at the decision time, with a particular emphasis on the use of simulation.

In Chap. 1, we discussed rolling-horizon control as an approximation framework for solving infinite-horizon MDPs. The rolling-horizon control policy is based on optimal values of finite-horizon MDPs with respect to a selected finite moving horizon, and the corresponding value function converges to the optimal value of the infinite-horizon MDP as the horizon length increases, uniformly in the initial state. Unfortunately, large state space and/or action spaces make it very difficult to solve finite-horizon MDPs in practice even with a relatively small rolling horizon, particularly when this needs to be done on-line before the action is taken. This motivates us to study an approximate rolling-horizon control that uses approximately optimal values for finite-horizon MDPs with respect to the selected moving horizon in an on-line context.

Throughout this chapter, we assume that the state space X and the action space A are countable, and the admissible action sets $A(x), x \in X$, are finite. Furthermore, we use the MDP simulation model from Chap. 1.

To emphasize the dependence on the horizon length H, we introduce the following notation for the H-horizon expected total discounted reward under policy π starting from initial state x:

$$\mathcal{V}_H^{\pi}(x) = E\left[\sum_{t=0}^{H-1} \gamma^t R'\big(x_t, \pi_t(x_t), w_t\big) \middle| x_0 = x\right],$$

instead of using the reward-to-go value function V_0^{π} defined by Eq. (1.6), and let $\mathcal{V}_H^*(x) = \sup_{\pi \in \Pi} \mathcal{V}_H^{\pi}(x), x \in X$. Recall that a rolling-horizon control policy $\pi_{\text{rh}} \in$

H.S. Chang et al., *Simulation-Based Algorithms for Markov Decision Processes*, Communications and Control Engineering,
DOI 10.1007/978-1-4471-5022-0_5, © Springer-Verlag London 2013

Π_s with a horizon $H < \infty$ is a policy that satisfies

$$T_{\pi_{\rm rh}}\big(\mathcal{V}^*_{H-1}\big)(x) = T\big(\mathcal{V}^*_{H-1}\big)(x), \quad x \in X,$$

where the T operators are defined by Eqs. (1.18), (1.19), (1.20), and (1.21). In other words, the rolling-horizon control policy $\pi_{\rm rh}$ is obtained from an optimal non-stationary policy $\pi^* = \{\pi^*_t, t = 0, 1, \ldots, H-1\}$ for the H-horizon problem by defining $\pi_{\rm rh} = \pi^*_0$. That is, at state $x_t \in X$ at decision time t, the decision maker takes the optimal first-stage action $\pi^*_0(x_t)$ by looking H steps ahead instead of looking at the entire infinite horizon, and then at time $t+1$, the decision maker does the same thing at x_{t+1} by looking H steps ahead from the state x_{t+1} by moving/rolling the H-step horizon forward. This results in a stationary policy $\pi_{\rm rh}$ for the infinite horizon, because we keep moving the H-step horizon forward, but it is derived from a non-stationary policy for the finite-horizon problem.

The performance of the rolling-horizon control policy relative to an optimal policy is bounded by (cf. Theorem 1.1)

$$0 \le V^*(x) - V^{\pi_{\rm rh}}(x) \le \frac{R_{\max}}{1-\gamma} \cdot \gamma^H, \quad x \in X.$$

For an approximate rolling-horizon control defined from the estimates of the $\mathcal{V}^*_{H-1}$-function (or V^*_1) values, we now show that the infinite-horizon discounted reward obtained by following the approximate rolling-horizon control relative to the optimal value is bounded by two error terms. The first error term is from the finite-horizon approximation, where it approaches zero geometrically with a given discount factor, and the second term is due to the approximation of the optimal reward-to-go value. If the rolling horizon is sufficiently long and the approximation of the optimal finite-horizon value is sufficiently accurate, the error bound will be relatively small.

Theorem 5.1 *Given $V \in B(X)$ such that $|\mathcal{V}^*_{H-1}(x) - V(x)| \le \epsilon$ for all x in X, consider a policy $\pi \in \Pi_s$ such that $T_\pi(V) = T(V)$. Then,*

$$0 \le V^*(x) - V^\pi(x) \le \frac{R_{\max}}{1-\gamma} \cdot \gamma^H + \frac{2\gamma\epsilon}{1-\gamma}, \quad x \in X.$$

Proof The lower bound is trivially true, so we prove only the upper bound. Let $\{v_n\}$ be the sequence of value iteration functions, with $v_n = T(v_{n-1})$, $v_0(x) = 0, x \in X$. From the contraction mapping property of the T-operator, for all x in X,

$$\big|T(v_{H-1})(x) - T(V)(x)\big| \le \gamma \cdot \sup_{x \in X}\big|v_{H-1}(x) - V(x)\big| \le \gamma\epsilon, \tag{5.1}$$

and

$$\sup_{x \in X}\big|V^*(x) - v_H(x)\big| = \sup_{x \in X}\big|T\big(V^*\big)(x) - T(v_{H-1})(x)\big|$$

$$
\begin{aligned}
&\leq \gamma \sup_{x\in X}\left|V^*(x) - v_{H-1}(x)\right| \\
&= \gamma \sup_{x\in X}\left|T\left(V^*\right)(x) - T(v_{H-2})(x)\right| \\
&\leq \gamma^2 \sup_{x\in X}\left|V^*(x) - v_{H-2}(x)\right| \\
&\cdots \\
&\leq \gamma^H \sup_{x\in X}\left|V^*(x) - v_0(x)\right| \leq \frac{R_{\max}}{1-\gamma}\cdot\gamma^H. \qquad (5.2)
\end{aligned}
$$

Therefore, from (5.1) and (5.2), and since $v_H = T(v_{H-1})$, for all $x \in X$,

$$
\begin{aligned}
\left|V^*(x) - T(V)(x)\right| &\leq \left|T(v_{H-1})(x) - T(V)(x)\right| + \left|V^*(x) - T(v_{H-1})(x)\right| \\
&\leq \frac{R_{\max}}{1-\gamma}\cdot\gamma^H + \gamma\epsilon. \qquad (5.3)
\end{aligned}
$$

Below we show that $T(V)(x) - V^\pi(x) \leq \frac{\gamma\epsilon(1+\gamma)}{1-\gamma}$ for all $x \in X$. It then follows from (5.3) that, for all $x \in X$,

$$
\begin{aligned}
V^*(x) - V^\pi(x) &\leq V^*(x) - T(V)(x) + T(V)(x) - V^\pi(x) \\
&\leq \frac{R_{\max}}{1-\gamma}\cdot\gamma^H + \gamma\epsilon + \frac{\gamma\epsilon(1+\gamma)}{1-\gamma} \\
&\leq \frac{R_{\max}}{1-\gamma}\cdot\gamma^H + \frac{2\gamma\epsilon}{1-\gamma},
\end{aligned}
$$

which gives the desired result. Now, for all $x \in X$,

$$
\begin{aligned}
T(V)(x) &= E\left[R'\left(x,\pi(x),w\right) + \gamma V\left(f\left(x,\pi(x),w\right)\right)\right] \\
&\leq E\left[R'\left(x,\pi(x),w\right) + \gamma\left(v_{H-1}\left(f\left(x,\pi(x),w\right)\right) + \epsilon\right)\right] \\
&= E\left[R'\left(x,\pi(x),w\right) + \gamma v_{H-1}\left(f\left(x,\pi(x),w\right)\right)\right] + \gamma\epsilon \\
&\leq E\left[R'\left(x,\pi(x),w\right) + \gamma v_H\left(f\left(x,\pi(x),w\right)\right)\right] + \gamma\epsilon \\
&\leq E\left[R'\left(x,\pi(x),w\right) + \gamma\left(T(V)\left(f\left(x,\pi(x),w\right)\right) + \gamma\epsilon\right)\right] + \gamma\epsilon \quad \text{by (5.1)} \\
&= E\left[R'\left(x,\pi(x),w\right) + \gamma T(V)\left(f\left(x,\pi(x),w\right)\right)\right] + \gamma^2\epsilon + \gamma\epsilon \\
&= E\left[R'\left(x,\pi(x),w\right) + \gamma E\left[R'\left(f\left(x,\pi(x),w\right),\pi\left(f\left(x,\pi(x),w\right)\right),w'\right)\right.\right. \\
&\quad \left.\left. + \gamma V\left(f\left(f\left(x,\pi(x),w\right),\pi\left(f\left(x,\pi(x),w\right)\right),w'\right)\right)\right]\right] + \gamma^2\epsilon + \gamma\epsilon \\
&= E\left[R'\left(x,\pi(x),w\right)\right] + \gamma E\left[R'\left(f\left(x,\pi(x),w\right),\pi\left(f\left(x,\pi(x),w\right)\right),w'\right)\right] \\
&\quad + \gamma^2 E\left[V\left(f\left(f\left(x,\pi(x),w\right),\pi\left(f\left(x,\pi(x),w\right)\right),w'\right)\right)\right] + \gamma\epsilon(1+\gamma) \\
&\leq E\left[R'\left(x,\pi(x),w\right)\right] + \gamma E\left[R'\left(f\left(x,\pi(x),w\right),\pi\left(f\left(x,\pi(x),w\right)\right),w'\right)\right]
\end{aligned}
$$

$$+\gamma^2 E\big[T(V)\big(f\big(f\big(x,\pi(x),w\big),\pi\big(f\big(x,\pi(x),w\big)\big),w'\big)\big)\big]$$
$$+\gamma^2\epsilon(1+\gamma)+\gamma\epsilon(1+\gamma).$$

Repeating the iteration in this way, we have, for all $n=0,1,\ldots$, and $x\in X$,

$$T(V)(x)\le E\left[\sum_{t=0}^{n}\gamma^t R'\big(x_t,\pi(x_t),w_t\big)\middle| x_0=x\right]+\gamma^{n+1}E\big[T(V)(x_{n+1})\big|x_0=x\big]$$
$$+\gamma\epsilon(1+\gamma)+\cdots+\gamma^{n+1}\epsilon(1+\gamma), \tag{5.4}$$

where x_t is the random variable representing the state at time t under π. Since $T(V)$ is bounded, the second term on the right-hand side of Eq. (5.4) converges to zero as $n\to\infty$ and the first term becomes $V^\pi(x)$. Therefore it follows that $T(V)(x)-V^\pi(x)\le\frac{\gamma\epsilon(1+\gamma)}{1-\gamma}$. □

Motivated by the bound in Theorem 5.1, either the UCB or the PLA sampling algorithm presented in Chap. 2 can be used within the approximate rolling-horizon control framework. A non-stationary randomized policy is created in an on-line manner by simulation, as follows. Suppose at time $t\ge 0$, the current state is $x\in X$. Each action's expected total reward is estimated by

$$\frac{1}{N_t}\sum_{j=1}^{N_t}\big[R'\big(x,a,w_j^a\big)+\gamma\hat{V}_{t+1}^{N_{t+1}}\big(f\big(x,a,w_j^a\big)\big)\big],\quad a\in A(x), \tag{5.5}$$

where the sequence $\{w_j^a,\ j=1,\ldots,N_t\}$ contains the corresponding random numbers used to simulate the next states from x by taking a. The UCB or the PLA sampling algorithm is applied at the next states $f(x,a,w_j^a)$, $j=1,\ldots,N_t$, to obtain the value of $\hat{V}_{t+1}^{N_{t+1}}(f(x,a,w_j^a))$. Recall that $\hat{V}_{t+1}^{N_{t+1}}(f(x,a,w_j^a))$ is an estimate of the unknown value of

$$\mathcal{V}_{H-1}^*\big(f\big(x,a,w_j^a\big)\big)$$
$$=\sup_{\pi\in\Pi}E\left[\sum_{s=t+1}^{t+H-1}\gamma^{s-t-1}R'\big(x_s,\pi_s(x_s),w_s\big)\middle| x_{t+1}=f\big(x,a,w_j^a\big)\right],$$

and N_i is the total number of samples that are used per state sampled in stage $i-t$, where $i=t,t+1,\ldots,t+H-1$. The decision maker chooses the action that maximizes the expected total reward estimated by (5.5). The use of common random numbers, i.e., using the same stream $\{w_j,\ j=1,\ldots,N_t\}$ for every $a\in A(x)$ in (5.5), to compare the different actions, should reduce the variance. The resulting randomized policy yields an approximate rolling H-horizon control for the infinite-horizon problem.

However, the time complexities of the two algorithms to estimate $V_1^*(x)$ or $\mathcal{V}_{H-1}^*(x)$ for a given state $x\in X$ are exponentially dependent on the horizon size

($O(N^H)$ for a sampling budget N used per state in each stage). Therefore, even for a small state space problem, applying the UCB or the PLA algorithm in an on-line context can be quite cumbersome.

In this chapter, we present on-line simulation-based control methods for solving infinite-horizon MDPs with large state and/or action spaces within the framework of approximate rolling-horizon control. Simulation is used to estimate V_1^*. The methods are especially suited for MDP problems where we can easily design a relatively small finite subset $\Lambda \subset \Pi$ of heuristic policies. The simulation complexity of each method depends polynomially on the horizon size, because the sampling is done with respect to a restricted policy space rather than the action space.

We first present an adaptive sampling algorithm called simulated annealing multiplicative weights (SAMW), which approximates $\mathcal{V}_H^*(x), x \in X$ by $\max_{\pi\in\Lambda} \mathcal{V}_H^\pi(x)$, $x \in X$ for a finite subset $\Lambda \subset \Pi$ of heuristic policies. Therefore, SAMW can be used for solving finite-horizon MDP problems when an optimal policy in $\Lambda \subset \Pi$ is sought. In an on-line context, the SAMW algorithm can be also used for two other purposes. First, it can be used for creating an approximate version of policy switching discussed in Sect. 3.1.1 in Chap. 3 for the EPI algorithm. This is particularly useful for solving MDPs having both large state and action spaces. Because only the given heuristic policies need to be simulated, the method works *independently of the size of the state and action spaces*. Second, it can be incorporated into obtaining an approximate "parallel rollout," defined with $\max_{\pi\in\Lambda} \mathcal{V}_{H-1}^\pi(x), x \in X$.

We then provide two approximate rolling-horizon controls, (parallel) rollout and "hindsight optimization," via lower and upper bounds to V_1^*, respectively, where both can be implemented easily by simulation. We illustrate the effectiveness of both approaches by considering a simple packet scheduling problem.

Finally, we apply the model-based annealing random search (MARS) of Sect. 4.6.1 as a direct policy search procedure over the policy space and combine it with the Q-learning method to present yet another on-line simulation-based MDP algorithm.

5.1 Simulated Annealing Multiplicative Weights Algorithm

In this section, we present a simulation-based algorithm called simulated annealing multiplicative weights (SAMW) for estimating $\max_{\pi\in\Lambda} \mathcal{V}_H^\pi(x)$ for $H < \infty$, when a non-empty *finite* subset $\Lambda \subset \Pi$ and $x \in X$ are given.

Recall that $V^\pi(x_0)$ in Eq. (1.1) denotes the value function of following policy $\pi \in \Pi$ over a finite horizon H for a given initial state x_0, i.e., $V^\pi(x_0) = \mathcal{V}_H^\pi(x_0)$. Because the initial state x_0 is fixed throughout this section, we use the notation V^π, omitting the initial state x_0, for simplicity. Without loss of generality, we assume that $R_{\max} \le 1/H$ in this section. The problem we consider is that of estimating the optimal value function over a given subset $\Lambda \subset \Pi$:

$$V^* := \max_{\pi\in\Lambda} V^\pi. \tag{5.6}$$

Any policy π^* that achieves V^* is called an optimal policy in Λ.

At each iteration, the SAMW algorithm updates a probability distribution over Λ by a simple multiplicative weight rule using the estimated (from simulation) value functions for all policies in Λ, requiring $|\Lambda|$ sample paths. (The idea of updating the probability distribution over the "solution" space Λ at each iteration is similar to MRAS presented in Chap. 4. Refer to Sect. 4.1 for the relationship between MRAS and SAMW.) The time complexity of the algorithm is $O(|\Lambda|H)$. With a proper "annealing" of the control parameter associated with the algorithm—analogous to the cooling parameter in simulated annealing, the sequence of distributions generated by the multiplicative weight rule converges to a distribution concentrated only on policies that achieve V^*, motivating the moniker "simulation annealing multiplicative weights."

The algorithm is "asymptotically efficient," in the sense that a *finite-time upper bound* is obtained for the sample mean of the value of an optimal policy in Λ, and the upper bound converges to V^* with rate $O(\frac{1}{\sqrt{T}})$, where T is the number of iterations. A sampling version of the algorithm that does not enumerate all policies in Λ at each iteration, but instead samples from the sequence of generated distributions, is also shown to converge to V^*.

5.1.1 Basic Algorithm Description

Let Φ be the set of all probability distributions over Λ. For $\phi \in \Phi$ and $\pi \in \Lambda$, let $\phi(\pi)$ denote the probability for policy π. The goal is to concentrate the probability on the optimal policies π^* in Λ. The SAMW algorithm iteratively generates a sequence of distributions, where ϕ_i denotes the distribution at iteration i. Each iteration of SAMW requires H random numbers $w_0, \ldots, w_{H-1}$, i.e., i.i.d. $U(0,1)$ and independent of previous iterations. Each policy $\pi \in \Lambda$ is then *simulated* using the same sequence of random numbers for that iteration (different random number sequences can also be used for each policy, and all of the results still hold) in order to obtain a sample path estimate of the value V^π:

$$V_i^\pi := \sum_{t=0}^{H-1} \gamma^t R'\big(x_t, \pi_t(x_t), w_t\big), \tag{5.7}$$

where the subscript i denotes the iteration count, which has been omitted for notational simplicity in the quantities x_t and w_t. The estimates $\{V_i^\pi, \pi \in \Lambda\}$ are used for updating a probability distribution over Λ at each iteration i. Note that $0 \le V_i^\pi \le 1$ (a.s.) by the boundedness assumption $R_{\max} \le 1/H$.

The iterative updating to compute the new distribution ϕ^{i+1} from ϕ^i and $\{V_i^\pi\}$ uses a simple multiplicative rule:

$$\phi^{i+1}(\pi) = \phi^i(\pi)\frac{\beta_i^{V_i^\pi}}{Z^i}, \quad \forall \pi \in \Lambda, \tag{5.8}$$

where $\beta_i > 1$ is a parameter of the algorithm, the normalization factor Z^i is given by

$$Z^i = \sum_{\pi \in \Lambda} \phi^i(\pi) \beta_i^{V_i^\pi},$$

and the initial distribution ϕ^1 is the uniform distribution, i.e.,

$$\phi^1(\pi) = \frac{1}{|\Lambda|} \quad \forall \pi \in \Lambda.$$

5.1.2 Convergence Analysis

For $\phi \in \Phi$, define

$$\bar{V}_i(\phi) := \sum_{\pi \in \Lambda} V_i^\pi \phi(\pi),$$

$$\Psi_T^\pi := \frac{1}{T} \sum_{i=1}^{T} V_i^\pi,$$

where Ψ_T^π is the sample mean estimate for the value function of policy π. Again, note that (a.s.) $0 \leq \bar{V}_i(\phi) \leq 1$ for all $\phi \in \Phi$. We remark that $\bar{V}_i(\phi)$ represents an *expected* reward for each fixed (iteration) experiment i, where the expectation is w.r.t. the distribution of the policy.

The following lemma provides a finite-time upper bound for the sample mean of the value function of an optimal policy in terms of the probability distributions generated by SAMW via Eq. (5.8).

Lemma 5.2 *For $\beta_i = \beta > 1$, $i = 1, \ldots, T$, the sequence of distributions $\phi^1, \ldots, \phi^T$ generated by SAMW via Eq. (5.8) satisfies (a.s.)*

$$\Psi_T^{\pi^*} \leq \frac{\beta - 1}{\ln \beta} \cdot \frac{1}{T} \sum_{i=1}^{T} \bar{V}_i(\phi^i) + \frac{\ln |\Lambda|}{T \ln \beta},$$

for any optimal policy π^* in Λ.

Proof The idea of the proof follows that of Theorem 1 in [65]. Recall that KL divergence between two p.m.f.s p and q is given by

$$\mathcal{D}(p, q) = \sum_{\pi \in \Lambda} p(\pi) \ln\left(\frac{p(\pi)}{q(\pi)}\right), \quad p, q \in \Phi. \tag{5.9}$$

Consider any Dirac distribution $\phi^* \in \Phi$ such that, for an optimal policy π^* in Λ, $\phi^*(\pi^*) = 1$ and $\phi^*(\pi) = 0$ for all $\pi \in \Lambda - \{\pi^*\}$. We first prove that

$$V_i^{\pi^*} \leq \frac{(\beta - 1)\bar{V}_i(\phi^i) + \mathcal{D}(\phi^*, \phi^i) - \mathcal{D}(\phi^*, \phi^{i+1})}{\ln \beta}, \tag{5.10}$$

where ϕ^i and ϕ^{i+1} are generated by SAMW via Eq. (5.8) and $\beta_i > 1$.

From the definition of $\mathcal{D}$ given by Eq. (5.9),

$$\begin{aligned}
\mathcal{D}\big(\phi^*, \phi^{i+1}\big) - \mathcal{D}\big(\phi^*, \phi^i\big) &= \sum_{\pi \in \Lambda} \phi^*(\pi) \ln\left(\frac{\phi^i(\pi)}{\phi^{i+1}(\pi)}\right) = \sum_{\pi \in \Lambda} \phi^*(\pi) \ln \frac{Z^i}{\beta^{V_i^\pi}} \\
&= -\sum_{\pi \in \Lambda} \phi^*(\pi) \ln \beta^{V_i^\pi} + \ln Z^i \sum_{\pi \in \Lambda} \phi^*(\pi) \\
&= (-\ln \beta) \sum_{\pi \in \Lambda} \phi^*(\pi) V_i^\pi + \ln Z^i \\
&\leq (-\ln \beta)\bar{V}_i\big(\phi^*\big) + \ln\left[\sum_{\pi \in \Lambda} \phi^i(\pi)\big(1 + (\beta - 1)V_i^\pi\big)\right] \\
&= (-\ln \beta) V_i^{\pi^*} + \ln\big(1 + (\beta - 1)\bar{V}_i\big(\phi^i\big)\big) \\
&\leq (-\ln \beta) V_i^{\pi^*} + (\beta - 1)\bar{V}_i\big(\phi^i\big),
\end{aligned}$$

where the first inequality follows from the property $\beta^a \leq 1 + (\beta - 1)a$ for $\beta \geq 0$, $a \in [0, 1]$, and the last inequality follows from the property $\ln(1 + a) \leq a$ for $a > -1$. Solving for $V_i^{\pi^*}$ (recall $\beta > 1$) yields (5.10).

Summing inequality (5.10) over $i = 1, \ldots, T$,

$$\begin{aligned}
\sum_{i=1}^{T} V_i^{\pi^*} &\leq \frac{\beta - 1}{\ln \beta} \sum_{i=1}^{T} \bar{V}_i\big(\phi^i\big) + \frac{1}{\ln \beta}\big(\mathcal{D}\big(\phi^*, \phi^1\big) - \mathcal{D}\big(\phi^*, \phi^{T+1}\big)\big) \\
&\leq \frac{\beta - 1}{\ln \beta} \sum_{i=1}^{T} \bar{V}_i\big(\phi^i\big) + \frac{1}{\ln \beta}\mathcal{D}\big(\phi^*, \phi^1\big) \\
&\leq \frac{\beta - 1}{\ln \beta} \sum_{i=1}^{T} \bar{V}_i\big(\phi^i\big) + \frac{\ln |\Lambda|}{\ln \beta},
\end{aligned}$$

where the second inequality follows from $\mathcal{D}(\phi^*, \phi^{T+1}) \geq 0$, and the last inequality uses the uniform distribution property that

$$\phi^1(\pi) = \frac{1}{|\Lambda|} \quad \forall \pi \quad \Longrightarrow \quad \mathcal{D}\big(\phi^*, \phi^1\big) \leq \ln |\Lambda|.$$

Dividing both sides by T yields the desired result. □

If $\frac{\beta-1}{\ln\beta}$ is very close to 1 and at the same time $\frac{\ln|\Lambda|}{T\ln\beta}$ is very close to 0, then the above inequality implies that the *expected* per-iteration performance of SAMW is very close to the optimal value. However, letting $\beta \to 1$, $\frac{\ln|\Lambda|}{\ln\beta} \to \infty$. On the other hand, for fixed β and T increasing, $\frac{\ln|\Lambda|}{\ln\beta}$ becomes negligible relative to T. Thus, from the form of the bound, it is clear that the sequence β_T should be chosen as a function of T such that $\beta_T \to 1$ and $T\ln\beta_T \to \infty$ in order to achieve convergence.

Define the total variation distance for p.m.f.s p and q by

$$d_T(p,q) := \sum_{\pi\in\Lambda} \big|p(\pi) - q(\pi)\big|.$$

The following lemma states that the sequence of distributions generated by SAMW converges to a stationary distribution, with a proper tuning or annealing of the β-parameter.

Lemma 5.3 *Let* $\{\psi(T)\}$ *be a decreasing sequence such that* $\psi(T) > 1\ \forall T$ *and* $\lim_{T\to\infty}\psi(T) = 1$. *For* $\beta_i = \psi(T),\ i = 1,\ldots,T+k,\ k \geq 1$, *the sequence of distributions* $\phi^1,\ldots,\phi^T$ *generated by SAMW via Eq.* (5.8) *satisfies* (*a.s.*)

$$\lim_{T\to\infty} d_T\big(\phi^T, \phi^{T+k}\big) = 0.$$

Proof From the definition of $\mathcal{D}$ given by Eq. (5.9),

$$\begin{aligned}
\mathcal{D}\big(\phi^T,\phi^{T+1}\big) &= \sum_{\pi\in\Lambda}\phi^T(\pi)\ln\left(\frac{\phi^T(\pi)}{\phi^{T+1}(\pi)}\right) \leq \max_{\pi\in\Lambda}\ln\left(\frac{\phi^T(\pi)}{\phi^{T+1}(\pi)}\right)\\
&= \max_{\pi\in\Lambda}\ln\frac{Z^T}{\psi(T)^{V_T^\pi}}\\
&\leq \max_{\pi\in\Lambda}\ln\frac{\sum_{\pi\in\Lambda}\phi^T(\pi)\psi(T)}{\psi(T)^{V_T^\pi}}\\
&= \max_{\pi\in\Lambda}\big(1-V_T^\pi\big)\ln\psi(T)\\
&\leq \ln\psi(T),
\end{aligned}$$

since $\psi(T) > 1$ and $0 \leq V_T^\pi \leq 1$ for all π and any T.

Applying Pinsker's inequality [176],

$$d_T\big(\phi^T,\phi^{T+1}\big) \leq \sqrt{2\mathcal{D}\big(\phi^T,\phi^{T+1}\big)} \leq \sqrt{2\ln\psi(T)}.$$

Therefore,

$$d_T\big(\phi^T,\phi^{T+k}\big) \leq \sum_{j=1}^{k} d_T\big(\phi^{T+j-1},\phi^{T+j}\big) \leq \sum_{j=0}^{k-1}\sqrt{2\ln\psi(T+j)}.$$

Because $d_T(\phi^T, \phi^{T+k}) \geq 0$ for any k and $\sum_{j=0}^{k-1} \sqrt{2 \ln \psi(T+j)} \to 0$ as $T \to \infty$, $d_T(\phi^T, \phi^{T+k}) \to 0$ as $T \to \infty$. □

Theorem 5.4 *Let* $\{\psi(T)\}$ *be a decreasing sequence such that* $\psi(T) > 1 \ \forall T$, $\lim_{T\to\infty} \psi(T) = 1$, *and* $\lim_{T\to\infty} T \ln \psi(T) = \infty$. *For* $\beta_i = \psi(T), \ i = 1, \ldots, T$, *the sequence of distributions* $\phi^1, \ldots, \phi^T$ *generated by SAMW via Eq.* (5.8) *satisfies* (*a.s.*)

$$\frac{\psi(T)-1}{\ln \psi(T)} \cdot \frac{1}{T} \sum_{i=1}^{T} \bar{V}_i(\phi^i) + \frac{\ln |\Lambda|}{T \ln \psi(T)} \to V^*,$$

and $\phi_i \to \phi^* \in \Phi$, *where* $\phi^*(\pi) = 0$ *for all* π *such that* $V^\pi < V^*$.

Proof Using $x - 1 \leq x \ln x$ for all $x \geq 1$ and Lemma 5.2,

$$\begin{aligned} \Psi_T^{\pi^*} &\leq \frac{\psi(T)-1}{\ln \psi(T)} \cdot \frac{1}{T} \sum_{i=1}^{T} \bar{V}_i(\phi^i) + \frac{\ln |\Lambda|}{T \ln \psi(T)} \\ &\leq \psi(T) \cdot \frac{1}{T} \sum_{i=1}^{T} \bar{V}_i(\phi^i) + \frac{\ln |\Lambda|}{T \ln \psi(T)}. \end{aligned} \tag{5.11}$$

In the limit as $T \to \infty$, the left-hand side converges to V^* by the law of large numbers, and in the rightmost expression in (5.11), $\psi(T) \to 1$ and the second term vanishes, so it suffices to show that $\frac{1}{T} \sum_{i=1}^{T} \bar{V}_i(\phi^i)$ is bounded from above by V^* (in the limit).

From Lemma 5.3, for every $\epsilon > 0$, there exists $T' < \infty$ such that $d_T(\phi^i, \phi^{i+k}) \leq \epsilon$ for all $i > T'$ and any integer $k \geq 1$. Then, for $T > T'$, we have (a.s.)

$$\begin{aligned} &\frac{1}{T} \sum_{i=1}^{T} \bar{V}_i(\phi^i) \\ &= \frac{1}{T} \left[\sum_{i=1}^{T'} \bar{V}_i(\phi^i) + \sum_{i=T'+1}^{T} \bar{V}_i(\phi^i) \right] \\ &\leq \frac{1}{T} \sum_{i=1}^{T'} \bar{V}_i(\phi^i) + \frac{1}{T} \sum_{i=T'+1}^{T} \bar{V}_i(\phi^{T'}) + \frac{1}{T} \sum_{i=T'+1}^{T} \sum_{\pi \in \Lambda} \left| \phi^i(\pi) - \phi^{T'}(\pi) \right| V_i^\pi \\ &\leq \frac{1}{T} \sum_{i=1}^{T'} \bar{V}_i(\phi^i) + \frac{1}{T} \sum_{i=1}^{T} \bar{V}_i(\phi^{T'}) + \frac{1}{T} \sum_{i=T'+1}^{T} \sum_{\pi \in \Lambda} \left| \phi^i(\pi) - \phi^{T'}(\pi) \right| V_i^\pi \\ &\leq \frac{1}{T} \sum_{i=1}^{T'} \bar{V}_i(\phi^i) + \frac{1}{T} \sum_{i=1}^{T} \bar{V}_i(\phi^{T'}) + \frac{1}{T} \sum_{i=T'+1}^{T} |\Lambda| \epsilon, \end{aligned} \tag{5.12}$$

the last inequality following from $\max_{\pi\in\Lambda}|\phi^{i+k}(\pi)-\phi^{i}(\pi)|\leq\epsilon$ and $V_i^{\pi}\leq 1\ \forall i > T'\ \forall k\geq 1, \pi\in\Lambda$. As $T\to\infty$, the first term of the right-hand side of (5.12) vanishes, and the second term converges by the law of large numbers to $V^{\pi}, \pi\sim\phi^{T'}$, which is bounded from above by V^*. Since ϵ can be chosen arbitrarily close to zero, the desired convergence follows.

The second part of the theorem follows directly from the first part with Lemma 5.3, with a proof obtained in a straightforward manner by assuming there exists a $\pi\in\Lambda$ such that $\phi^*(\pi)\neq 0$ and $V^{\pi} < V^*$, leading to a contradiction. □

An example of a decreasing sequence $\{\psi(T)\},\ T=1,2,\ldots,$ that satisfies the condition of Theorem 5.4 is $\psi(T)=1+\sqrt{\frac{1}{T}},\ T>0$.

5.1.3 Convergence of the Sampling Version of the Algorithm

Instead of estimating the value functions for every policy in Λ according to Eq. (5.7), which requires simulating all policies in Λ, a *sampling* version of the algorithm would sample a subset of the policies in Λ at each iteration i according to ϕ^i and simulate only those policies (and estimate their corresponding value functions). In this context, Theorem 5.4 essentially establishes that the *expected* per-iteration performance of SAMW approaches the optimal value as $T\to\infty$ for an appropriately selected tuning sequence $\{\beta_i\}$. Here, we show that the actual (distribution sampled) per-iteration performance also converges to the optimal value using a particular annealing schedule of the parameter β. For simplicity, we assume that a *single* policy is sampled at each iteration (i.e., subset is a singleton).

Theorem 5.5 *Let* $T_k=\sum_{j=1}^{k}j^2$. *For* $\beta_i=1+\frac{1}{k}, T_{k-1}<i\leq T_k$, *let* $\{\phi^i\}$ *denote the sequence of distributions generated by SAMW via Eq.* (5.8), *with "resetting" of* $\phi^i(\pi)=\frac{1}{|\Lambda|}\ \forall\pi$ *at each* $i=T_k$. *Let* $\hat{\pi}(\phi^i)$ *denote the policy sampled from* ϕ^i *(at iteration* i*). Then*

$$\frac{1}{T_k}\sum_{i=1}^{T_k}V_i^{\hat{\pi}(\phi^i)}\to V^* \quad w.p.1.$$

Proof The sequence of random variables $\kappa_i=V_i^{\hat{\pi}(\phi^i)}-\bar{V}_i(\phi^i)$ forms a martingale difference sequence with $|\kappa_i|\leq 1$, since $E[\kappa_i|\kappa_1,\ldots,\kappa_{i-1}]=0$ for all i. Let $\epsilon_k=2\frac{\sqrt{\ln k}}{k}$ and $I_k=[T_{k-1}+1, T_k]$. Applying Azuma's inequality [149, p. 309], we have, for every $\epsilon_k>0$,

$$P\left(\frac{1}{k^2}\left|\sum_{i\in I_k}\left(V_i^{\hat{\pi}(\phi^i)}-\bar{V}_i(\phi^i)\right)\right|>\epsilon_k\right)\leq 2e^{-0.5k^2\epsilon_k^2}=\frac{2}{k^2}. \tag{5.13}$$

The sum of the probability bound in (5.13) over all k from 1 to ∞ is finite. Therefore, by the Borel–Cantelli lemma (a.s.) all but a finite number of I_k's ($k = 1, \ldots, \infty$) satisfy

$$\sum_{i \in I_k} \bar{V}_i(\phi^i) \leq \sum_{i \in I_k} V_i^{\hat{\pi}(\phi^i)} + k^2 \epsilon_k, \tag{5.14}$$

so those I_k that violate inequality (5.13) can be ignored (a.s.).

From Lemma 5.2 with the definition of β_i, for all $i \in I_k$,

$$\begin{aligned} k^2 \Psi_{k^2}^{\pi^*} &\leq \sum_{i \in I_k} \frac{\beta_i - 1}{\ln \beta_i} \bar{V}_i(\phi^i) + \frac{\ln |\Lambda|}{\ln \beta_i} \leq \sum_{i \in I_k} \beta_i \bar{V}_i(\phi^i) + \frac{\ln |\Lambda|}{\frac{\beta_i - 1}{\beta_i}} \\ &= \sum_{i \in I_k} \left(1 + \frac{1}{k}\right) \bar{V}_i(\phi^i) + \ln |\Lambda| (k+1) \\ &\leq \sum_{i \in I_k} \bar{V}_i(\phi^i) + k + \ln |\Lambda| (k+1), \end{aligned} \tag{5.15}$$

where the last inequality follows from $\bar{V}_i(\phi) \leq 1 \ \forall \phi \in \Phi$ and $|I_k| = k^2$.

Combining inequalities (5.14) and (5.15) and summing, we have

$$T_k \Psi_{T_k}^{\pi^*} \leq \sum_{i \in I_1 \cup \cdots \cup I_k} V_i^{\hat{\pi}(\phi^i)} + \sum_{j=1}^{k} \left[2j\sqrt{\ln j} + j\left(\ln |\Lambda| + 1\right) + \ln |\Lambda|\right],$$

so

$$\Psi_{T_k}^{\pi^*} \leq \frac{1}{T_k} \sum_{i \in I_1 \cup \cdots \cup I_k} V_i^{\hat{\pi}(\phi^i)} + \frac{1}{T_k} \sum_{j=1}^{k} \left[2j\sqrt{\ln j} + j\left(\ln |\Lambda| + 1\right) + \ln |\Lambda|\right]. \tag{5.16}$$

Because T_k is $O(k^3)$, the term on the right-hand side of (5.16) vanishes as $k \to \infty$. Therefore, for every $\epsilon > 0$, (a.s.) for all but a finite number of values of T_k,

$$\Psi_{T_k}^{\pi^*} \leq \frac{1}{T_k} \sum_{i=1}^{T_k} V_i^{\hat{\pi}(\phi^i)} + \epsilon.$$

We now argue that $\{\phi^i\}$ converges to a fixed distribution as $k \to \infty$, so that eventually the term $\frac{1}{T_k} \sum_{i=1}^{T_k} V_i^{\hat{\pi}(\phi^i)}$ is bounded from above by V^*. With similar reasoning as in the proof of Lemma 5.3, for every $\epsilon > 0$, there exists $T' \in I_k$ for some $k > 1$ such that, for all $i > T'$ with $i + j \in I_k$, $j \geq 1$, $d_T(\phi^i, \phi^{i+j}) \leq \epsilon$. Taking $T > T'$ with $T \in I_k$,

$$\begin{aligned}
&\frac{1}{T}\sum_{i=1}^{T} V_i^{\hat{\pi}(\phi^i)} \\
&= \frac{1}{T}\left[\sum_{i=1}^{T'} V_i^{\hat{\pi}(\phi^i)} + \sum_{i=T'+1}^{T} V_i^{\hat{\pi}(\phi^i)}\right] \\
&\leq \frac{1}{T}\sum_{i=1}^{T'} V_i^{\hat{\pi}(\phi^i)} + \frac{1}{T}\sum_{i=T'+1}^{T} V_i^{\hat{\pi}(\phi^{T'})} + \frac{1}{T}\sum_{i=T'+1}^{T} V_i^{\hat{\pi}(\phi^i)} - V_i^{\hat{\pi}(\phi^{T'})} \\
&\leq \frac{1}{T}\sum_{i=1}^{T'} V_i^{\hat{\pi}(\phi^i)} + \frac{1}{T}\sum_{i=1}^{T} V_i^{\hat{\pi}(\phi^{T'})} + \frac{1}{T}\sum_{i=T'+1}^{T} \left(V_i^{\hat{\pi}(\phi^i)} - V_i^{\hat{\pi}(\phi^{T'})}\right). \qquad (5.17)
\end{aligned}$$

Letting $T \to \infty$, the first term on the right-hand side of (5.17) vanishes, and the second term is bounded from above by V^*, because the second term converges to $V^{\pi}, \pi \sim \phi^{T'}$, from the law of large numbers. We know that, for all $i > T'$ in I_k, $-\epsilon + \phi^i(\pi) \leq \phi^{i+j}(\pi) \leq \phi^i(\pi) + \epsilon$ for all $\pi \in \Lambda$ and any j. Therefore, as ϵ can be chosen arbitrarily close to zero, the sequence $\{\phi^i\}$ converges to the distribution $\phi^{T'}$, making the last term vanish (once each policy is sampled from the same distribution over Λ, the simulated value would be the same for the same random numbers), providing the desired convergence result. □

5.1.4 Numerical Example

To illustrate the performance of SAMW, we consider a finite-horizon inventory control problem, the same one presented in Sect. 2.1, but with different problem parameter values. Given an inventory level, orders are placed and received, demand is realized, and the new inventory level is calculated. Let D_t, a discrete random variable, denote the demand in period t, x_t the inventory level at period t, a_t the order amount at period t, p the per period per unit demand lost penalty cost, h the per period per unit inventory holding cost, and M the inventory capacity. Thus, the inventory level evolves according to the following dynamics:

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

The goal is to minimize, over a given set of (non-stationary) policies Λ, the expected total cost over the entire horizon from a given initial inventory level x_0, i.e.,

$$\min_{\pi \in \Lambda} E\left[\sum_{t=0}^{H-1} \left[h\left(x_t + \pi_t(x_t) - D_t\right)^+ + p\left(0, D_t - x_t - \pi_t(x_t)\right)^+\right] \middle| x_0 = x\right].$$

Since there are no ordering costs, the optimal order policy follows a base-stock policy, in which ordering is placed to bring the inventory to the base-stock level

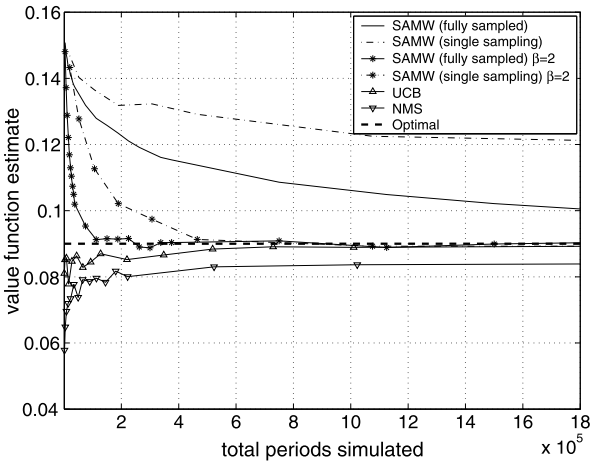


Fig. 5.1 Average performance (mean of 25 simulation replications, resulting in confidence half-widths within 5 % of estimated mean) of SAMW, the UCB sampling algorithm, and NMS on the inventory control problem ($h = 0.003,\ p = 0.012$)

S_t. Specifically, an order is placed in period t if the inventory level x_t is below the threshold S_t, with the order amount being $(S_t - x_t)^+$. Exploiting this structural property, we restrict the search of SAMW to the set of threshold policies, i.e., $\Lambda = (S_0, S_1, S_2),\ S_t \in \{0, 5, 10, 15, 20\},\ t = 0, 1, 2$, rather than considering the set of all admissible policies.

We implemented two versions of SAMW, i.e., the fully sampled version of SAMW, which constructs the optimal value function estimate by enumerating all policies in Λ and using all value function estimates, and the single sampling version of SAMW introduced in Theorem 5.5, which uses just one sampled policy in each iteration to update the optimal value function estimate; however, updating ϕ^i requires value function estimates for all policies in Λ. For numerical comparison, we also applied the UCB sampling algorithm discussed in Sect. 2.1 and a non-adaptive multi-stage sampling (NMS) algorithm [104].

The following parameter values were used in our experiments: $M = 20,\ H = 3,\ h = 0.003,\ p = 0.012,\ x_0 = 5$ and $x_t \in \{0, 5, 10, 15, 20\}$ for $t = 1, \ldots, H$, $a_t \in \{0, 5, 10, 15, 20\}$ for all $t = 0, \ldots, H - 1$, and D_t is a discrete uniformly distributed random variable taking values in $\{0, 5, 10, 15, 20\}$. The values of h and p are chosen so as to satisfy the reward bound ($R_{\max} \leq 1/H$) assumed in the SAMW convergence results. Figure 5.1 shows the performance of these algorithms as a function of the total number of periods simulated, based on 25 independent replications. The results indicate convergence of both versions of SAMW; however, the two alternative UCB and NMS algorithms seem to provide superior empirical performance over SAMW. We believe this is because the annealing schedule for β used in SAMW is too conservative for this problem, thus leading to slow convergence. To improve the empirical performance of SAMW, we also implemented both versions of the algorithm with β being held constant throughout the search, i.e., independent of T.

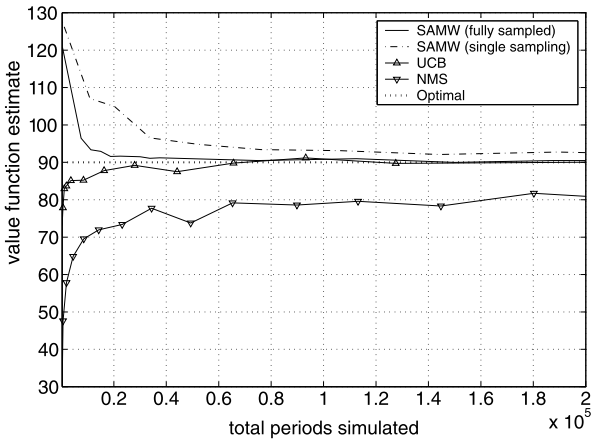


Fig. 5.2 Average performance (mean of 25 simulation replications, resulting in confidence half-widths within 5 % of estimated mean) of SAMW, the UCB sampling algorithm, and NMS on the inventory control problem ($h = 3,\ p = 12$)

The $\beta = 2$ case is included in Fig. 5.1, which shows significantly improved performance. Experimentation with the SAMW algorithm also revealed that it performed even better for cost parameter values in the inventory control problem that do not satisfy the strict reward bound. One such example is shown in Fig. 5.2, for the case $h = 3$ and $p = 12$ (all other parameter values unchanged). Furthermore, note that the value of the horizon size in our experiments is relatively small ($H = 3$) to focus on the quality of the estimate by each algorithm. As the value of H increases, both UCB and NMS algorithms suffer from large sampling complexities.

SAMW can be naturally parallelized to reduce its computational cost. Partition the given policy space Λ into $\{\Delta_j\}$ such that $\Delta_j \cap \Delta_{j'} = \emptyset$ for all $j \neq j'$ and $\bigcup_j \Delta_j = \Lambda$, and apply the algorithm in parallel for T iterations on each Δ_j. For a fixed value of $\beta > 1$, we have the following finite-time bound from Lemma 5.2:

$$V_T^* \leq \max_j \left\{ \frac{\beta - 1}{\ln \beta} \cdot \frac{1}{T} \sum_{i=1}^{T} \bar{V}_i\left(\phi_j^i\right)\left(x_j^i\right) + \frac{\ln |\Delta_j|}{T \ln \beta} \right\},$$

where ϕ_j^i is the distribution generated for Δ_j at iteration i.

The original version of SAMW recalculates an estimate of the value function for all policies in Λ at each iteration, requiring each policy to be simulated. If Λ is large, this may not be practical, and the sampling version of SAMW given by Theorem 5.5 also requires each value function estimate in order to update the ϕ^i at each iteration. One simple alternative is to use the prior value function estimates for updating ϕ^i, except for the single sampled one; thus, only one simulation per iteration would be

required. Specifically,

$$V_i^\pi := V_{i-1}^\pi \quad \text{if } \pi \text{ not sampled at iteration } i;$$
$$\text{else obtain a new estimate of } V_i^\pi \text{ via Eq. (5.7).}$$

An extension of this is to use a threshold on ϕ^i to determine which policies will be simulated. Since the sequence of the distributions generated by SAMW converges to a distribution concentrated on the optimal policies in Λ, as the number of iterations increases, the contributions from non-optimal policies get smaller and smaller, so these policies need not be resimulated (and value function estimates updated) very often. Specifically,

$$V_i^\pi := V_{i-1}^\pi \quad \text{if } \phi^i(\pi) \le \epsilon;$$
$$\text{else obtain a new estimate of } V_i^\pi \text{ via Eq. (5.7).}$$

The cooling schedule presented in Theorem 5.5 is just one way of controlling the parameter β. Characterizing properties of good schedules is critical to effective implementation, as the numerical experiments showed. The numerical experiments also demonstrated that the algorithm may work well outside the boundaries of the assumptions under which theoretical convergence is proved, specifically the bound on the one-period reward function and the value of the cooling parameter β.

5.1.5 Simulated Policy Switching

Recall that, for a given non-empty finite subset $\Lambda \subset \Pi_s$, policy switching is defined by the operation on the right-hand side of (3.1). The key advantage of policy switching is that it works independently of the size of the state and the action spaces and improves all policies in Λ. Consider the H-horizon policy switching policy such that

$$\pi_{\mathrm{ps},H}(x) \in \left\{ \arg\max_{\pi \in \Lambda} \left(\mathcal{V}_H^\pi(x) \right)(x) \right\}, \quad x \in X.$$

The decision maker can apply SAMW at each decision time t to estimate $\max_{\pi \in \Lambda} \mathcal{V}_H^\pi(x_t)$ and obtain an approximate π^* in Λ for the initial state x_t, and take $\pi^*(x_t)$ at time t and move on to the next state x_{t+1}.

An important property of *simulated* policy switching is that if we estimate $\mathcal{V}_H^\pi(x), \pi \in \Lambda$, by the sample mean of $\bar{\mathcal{V}}_H^\pi(x)$ based on simulation, then the policy switching operation defined by

$$\bar{\pi}_{\mathrm{ps},H}(x) \in \left\{ \arg\max_{\pi \in \Lambda} \left(\bar{\mathcal{V}}_H^\pi(x) \right)(x) \right\}, \quad x \in X, \tag{5.18}$$

is an *ordinal comparison*, which leads to an exponential convergence rate.

For example, at each decision time t, the decision maker generates a set of N random number sequences $\{w_0^j, \ldots, w_{H-1}^j\}$, $j = 1, \ldots, N$, to estimate $\mathcal{V}_H^\pi(x)$ for the current state x by the sample mean $\bar{\mathcal{V}}_H^\pi(x)$ of the reward sums over N sample paths,

$$\bar{\mathcal{V}}_H^\pi(x) := \frac{1}{N} \sum_{j=1}^{N} \sum_{t'=0}^{H-1} \gamma^{t'} R'\big(x_{t'}^j, \pi\big(x_{t'}^j\big), w_{t'}^j\big), \quad x_0^j = x, \ j = 1, \ldots, N,$$

where each policy $\pi \in \Lambda$ is simulated with the same random number sequence. The decision maker then takes the action prescribed by the (simulated) policy switching at t, moving to the next state at time $t+1$. This process is repeated at time $t+1$. In this case, we can immediately state the following on the probability of the correct selection:

Theorem 5.6 *Consider $\Lambda = \{\pi^1, \ldots, \pi^{|\Lambda|}\} \subset \Pi_s$ and a fixed state $x \in X$. Assume that*

$$\mathcal{V}_H^{\pi^1}(x) > \mathcal{V}_H^{\pi^2}(x) > \cdots > \mathcal{V}_H^{\pi^{|\Lambda|}}(x).$$

For $\bar{\pi}_{\mathrm{ps},H}$ defined on Λ by (5.18),

$$P\big(\bar{\pi}_{\mathrm{ps},H}(x) = \pi^1(x)\big) \geq 1 - \alpha e^{-N\beta},$$

for some constants $\alpha, \beta > 0$.

That is, the probability of the correct selection (selecting the action prescribed by policy switching at x) by $\bar{\pi}_{\mathrm{ps},H}(x)$ converges to one *exponentially* in N.

One drawback of policy switching is that it gives insufficient emphasis and freedom in the evaluation of the *initial action*, which is the only action actually selected and committed to. Parallel rollout was designed to overcome this drawback by combining rollout and policy switching. We now present the rollout method of incorporating the (estimate) value of $\max_{\pi \in \Lambda} V_{H-1}^\pi(x)$, $x \in X$, into an approximate rolling-horizon control as an estimate of V_1^*. We begin with the single-policy case ($|\Lambda| = 1$) and then generalize to the multi-policy case.

5.2 Rollout

Although obtaining an optimal policy for an MDP is often quite difficult due to the curse of dimensionality, a heuristic policy can be easily designed in many cases. The idea of the rollout approach is to improve a heuristic policy via simulation. We "roll out" or simulate the available policy over a selected finite horizon $H < \infty$ and then apply the most "promising" action to the system in an on-line manner.

We define the H-horizon rollout policy $\pi_{\mathrm{ro},H}$ with a base-policy $\pi \in \Pi_s$ to be a policy such that

$$\pi_{\mathrm{ro},H}(x) \in \arg\max_{a \in A(x)} E\big[R'(x,a,w) + \gamma \mathcal{V}^{\pi}_{H-1}\big(f(x,a,w)\big)\big], \quad x \in X.$$

To analyze the performance of the H-horizon rollout policy relative to its base-policy π, we first begin with a lemma that can be easily proven by the monotonicity property of the operator T_π and the convergence to the unique fixed point of V^π from successive applications of the operator.

Lemma 5.7 *Suppose there exists $\psi \in B(X)$ for which $T_\pi(\psi)(x) \geq \psi(x)$ for all $x \in X$; then $V^\pi(x) \geq \psi(x)$ for all $x \in X$.*

Theorem 5.8 *For any $\epsilon > 0$, if $H \geq 1 + \log_\gamma \frac{\epsilon(1-\gamma)}{R_{\max}}$, then for all $x \in X$,*

$$V^{\pi_{\mathrm{ro},H}}(x) \geq V^{\pi}(x) - \epsilon.$$

Proof Define $\psi = \mathcal{V}^{\pi}_{H-1}$. By definition of the rollout policy, for any $x \in X$,

$$\begin{aligned} T_{\pi_{\mathrm{ro},H}}(\psi)(x) &= E\big[R'\big(x, \pi_{\mathrm{ro},H}(x), w\big) + \gamma \psi\big(f\big(x, \pi_{\mathrm{ro},H}(x), w\big)\big)\big] \\ &\geq E\big[R'\big(x, \pi(x), w\big) + \gamma \psi\big(f\big(x, \pi(x), w\big)\big)\big] = \mathcal{V}^{\pi}_{H}(x) \geq \psi(x). \end{aligned}$$

Therefore, for all $x \in X$, we see that $V^{\pi_{\mathrm{ro},H}}(x) \geq \mathcal{V}^{\pi}_{H-1}(x)$ by Lemma 5.7. Now we can write for all $x \in X$, $V^\pi(x) = \mathcal{V}^{\pi}_{H-1}(x) + \gamma^{H-1} E[V^\pi(x_{H-1})|x_0 = x]$, where E is taken with respect to the probability distribution over the state x_{H-1} at time $H-1$. We know that $\sup_{x \in X} V^\pi(x) \leq \frac{R_{\max}}{1-\gamma}$. This implies that $V^{\pi_{\mathrm{ro},H}}(x) \geq V^\pi(x) - \frac{R_{\max}}{1-\gamma} \cdot \gamma^{H-1}$. Letting $\frac{R_{\max}}{1-\gamma} \cdot \gamma^{H-1} \leq \epsilon$ yields the desired result. □

Note that as $H \to \infty$, the above result gives the result of a single step of policy improvement of PI. Therefore, we can view the rollout approach as an on-line implementation of the policy improvement step of PI via simulation. Rollout can be easily implemented regardless of the size of the state space.

The following result is immediate from Theorem 5.1.

Corollary 5.9 *If $\sup_{x \in X} |V^*_1(x) - \mathcal{V}^{\pi}_{H-1}(x)| \leq \epsilon$,*

$$0 \leq V^*(x) - V^{\pi_{\mathrm{ro},H}}(x) \leq \frac{R_{\max}}{1-\gamma} \cdot \gamma^H + \frac{2\gamma\epsilon}{1-\gamma}, \quad x \in X.$$

To implement the H-horizon rollout policy by simulation using common random numbers, at each decision time t, the decision maker generates random numbers $w_0^j \sim U(0,1)$, $j = 1, \ldots, N$, where w_0^j determines the random next state y^j from the current state x by taking $a \in A(x)$, i.e., $f(x,a,w_0^j) = y^j$. A set of L

random number sequences $\{w_1^k, \ldots, w_{H-1}^k\},\ k = 1, \ldots, L$, is then generated to estimate $\mathcal{V}_{H-1}^{\pi}(y^j)$ by the sample mean $\bar{\mathcal{V}}_{H-1}^{\pi}(y^j)$ over L sample paths:

$$\bar{\mathcal{V}}_{H-1}^{\pi}\big(y^j\big) := \frac{1}{L}\sum_{k=1}^{L}\sum_{t'=1}^{H-1}\gamma^{t'-1}R'\big(x_{t'}^k, \pi\big(x_{t'}^k\big), w_{t'}^k\big), x_1^k = y^j, \quad j = 1, \ldots, N,$$

where the policy π is simulated with the same random number sequence across the next state $y^j,\ j = 1, \ldots, N$. The decision maker then takes the action prescribed by the (simulated) rollout policy at t,

$$\arg\max_{a\in A(x)} \frac{1}{N}\sum_{j=1}^{N}\big(R'\big(x, a, w_0^j\big) + \gamma\bar{\mathcal{V}}_{H-1}^{\pi}\big(y^j\big)\big),$$

moving to the next state at time $t+1$. This process is repeated at $t+1$.

5.2.1 Parallel Rollout

The rollout approach with a *single* base-policy is promising if we have a good base-policy. Note that in practice, what we are really interested in is the *ranking* of actions, not the degree of approximation. Therefore, as long as the rollout policy preserves the true ranking of actions well, the resulting policy will perform fairly well. Differential training, or the use of common random numbers, can be used for this purpose.

However, when we have multiple heuristic policies available, because we cannot predict the performance of each policy in advance, selecting a particular single base-policy to be rolled out is not an easy task. Furthermore, for some cases, each base-policy available may be good for different system paths. When this is the case, we wish to combine these base-policies *dynamically* in an on-line manner to generate a single policy that adapts automatically to different paths of the system, in addition to alleviating the difficulty of choosing a single base-policy to be rolled out.

Parallel rollout is a generalization of the rollout approach when a set $\Lambda \subset \Pi_s$ of heuristic policies can be obtained rather than a single policy. The name "parallel rollout" comes from the idea that we roll out each available policy over a selected finite horizon $H < \infty$ to estimate its infinite-horizon value *in parallel* via a simulation over H, and then apply the most "promising" action based on the estimates of the heuristic policies to the system in an on-line manner. Like rollout, parallel rollout can be easily implemented regardless of the size of the state space.

We define the H-horizon parallel rollout policy $\pi_{\mathrm{pr},H}$ with a non-empty finite subset $\Lambda \subset \Pi_s$ to be a policy such that

$$\pi_{\mathrm{pr},H}(x) \in \arg\max_{a\in A(x)} E\Big[R'(x, a, w) + \gamma\max_{\pi\in\Lambda}\mathcal{V}_{H-1}^{\pi}\big(f(x, a, w)\big)\Big],$$

where $\mathcal{V}^{\pi}_{H-1}$-function values are approximated by sample averages via simulation for each $\pi \in \Lambda$.

Theorem 5.10 *For $\pi_{\mathrm{pr},H}$ defined on a non-empty finite subset $\Lambda \subset \Pi_s$, given any $\epsilon > 0$, if $H \geq 1 + \log_\gamma \frac{\epsilon(1-\gamma)}{R_{\max}}$, then for all $x \in X$,*

$$V^{\pi_{\mathrm{pr},H}}(x) \geq \max_{\pi \in \Lambda} V^{\pi}(x) - \epsilon.$$

Proof As in the proof of Theorem 5.8, we define $\psi(x) = \max_{\pi \in \Lambda} \mathcal{V}^{\pi}_{H-1}(x)$, $x \in X$. We have, for any $\pi \in \Lambda$ and $x \in X$,

$$\begin{aligned} T_{\pi_{\mathrm{pr},H}}(\psi)(x) &= E\big[R'\big(x, \pi_{\mathrm{pr},H}(x), w\big) + \gamma \psi\big(f\big(x, \pi_{\mathrm{pr},H}(x), w\big)\big)\big] \\ &\geq E\big[R'\big(x, \pi(x), w\big) + \gamma \psi\big(f\big(x, \pi(x), w\big)\big)\big] = \mathcal{V}^{\pi}_{H}(x). \end{aligned}$$

Therefore,

$$T_{\pi_{\mathrm{pr},H}}(\psi)(x) \geq \max_{\pi \in \Lambda} \mathcal{V}^{\pi}_{H}(x) \geq \psi(x), \quad x \in X.$$

Thus, for all $x \in X$, we have $V^{\pi_{\mathrm{pr},H}}(x) \geq \max_{\pi \in \Lambda} \mathcal{V}^{\pi}_{H-1}(x)$ by Lemma 5.7. We know that $V^{\pi_{\mathrm{pr},H}}(x) \geq \max_{\pi \in \Lambda} V^{\pi}(x) - \frac{R_{\max}}{1-\gamma} \cdot \gamma^{H-1}$ (cf. Theorem 5.8). Letting $\frac{R_{\max}}{1-\gamma} \cdot \gamma^{H-1} \leq \epsilon$ yields the desired result. □

As with rollout, the following performance bound follows from Theorem 5.1.

Corollary 5.11 *If $\sup_{x \in X} |V^*_1(x) - \max_{\pi \in \Lambda} \mathcal{V}^{\pi}_{H-1}(x)| \leq \epsilon$,*

$$0 \leq V^*(x) - V^{\pi_{\mathrm{pr},H}}(x) \leq \frac{R_{\max}}{1-\gamma} \cdot \gamma^H + \frac{2\gamma\epsilon}{1-\gamma}, \quad x \in X.$$

The parallel rollout method is based on *multi-policy improvement* stated in the following theorem, which generalizes the single policy-improvement step of the PI algorithm to the case of multiple policies, where this result coincides with $H \to \infty$ in Theorem 5.10.

Theorem 5.12 *Given a non-empty finite subset $\Lambda \subset \Pi_s$, define a parallel rollout policy π_{pr} such that, for $x \in X$,*

$$\pi_{\mathrm{pr}}(x) \in \arg\max_{a \in A(x)} E\Big[R'(x, a, w) + \gamma \max_{\pi \in \Lambda} V^{\pi}\big(f(x, a, w)\big)\Big].$$

Then,

$$V^{\pi_{\mathrm{pr}}}(x) \geq \max_{\pi \in \Lambda} V^{\pi}(x), \quad x \in X.$$

The policy obtained by parallel rollout (with $H = \infty$) is essentially the same as the PIRS (policy improvement with reward swapping) step of the ERPS algorithm in Chap. 3 (cf. (3.4)), where it was used in a different context to obtain an "elite" policy. Using simulation differentiates parallel rollout from the PIRS procedure, which becomes impractical for large enough state spaces. Note that the SAMW algorithm can be incorporated into parallel rollout for estimating the value of $\max_{\pi \in \Lambda} \mathcal{V}^{\pi}_{H-1}(x), x \in X$.

5.3 Hindsight Optimization

The method of hindsight optimization is based on approximating V^*_1 or $\mathcal{V}^*_{H-1}$ by an upper bound and using the approximation for the rolling-horizon control for solving an infinite-horizon MDP problem. The bound used by hindsight optimization is obtained simply by interchanging the order of expectation and maximization in the recursive definition of Q^*_0 in Eq. (1.9), and applying Jensen's inequality. More precisely, we bound the Q^*_0-function value of a current control choice a at x, $Q^*_0(x, a)$, with

$$Q^{\mathrm{ho}}_0(x, a) = E\big[R'(x, a, w) + \gamma V^{\mathrm{ho}}_1\big(f(x, a, w)\big)\big],$$

where

$$V^{\mathrm{ho}}_1(y) = E\left[\max_{a_1,\ldots,a_{H-1}} \sum_{t=1}^{H-1} \gamma^{t-1} R'(x_t, a_t, w_t) \,\middle|\, x_1 = y\right], \quad y \in X,$$

with $a_t \in A(x_t)$ at each t. From the above equation, once a particular random number sequence $\{w_0, w_1, \ldots, w_{H-1}\}$ is selected, w_0 determines the next state y from state x by taking $a \in A(x)$, i.e., $f(x, a, w_0) = y$, and the maximization inside becomes a deterministic problem of choosing a "best" control sequence over $H - 1$ time steps (starting with the next state) to maximize the total (discounted) reward with respect to the random number sequence. The best control sequence is the sequence of controls the decision maker would select after taking the action a if it somehow knew the random numbers to come (i.e., in "hindsight"). Therefore, the inner deterministic problem is called a "hindsight optimization problem" and the approximated Q^*_0-function value is the "hindsight-optimal Q-function value." The hindsight-optimal Q-function value $Q^{\mathrm{ho}}_0(x, a), x \in X, a \in A(x)$, estimates the value of $Q^*(x, a)$ for the approximate rolling-horizon control. The controller simply takes $\arg\max_{a \in A(x)} Q^{\mathrm{ho}}_0(x, a)$ at state $x \in X$.

We remark that a hindsight optimization problem can often be solved exactly, and this solution can be used to obtain a theoretical bound for the performance when we evaluate a target policy. One drawback with hindsight optimization is that the computational complexity of obtaining the best control sequence may be large.

5.3.1 Numerical Example

We consider a simple scheduling problem for randomly arriving packets into a single server. Each packet belongs to a finite set of classes, and if the server does not serve a packet before its deadline, the packet is lost. Each class is associated with a weight that represents the importance of the class. Every packet takes one-unit time to serve. Packets arrive into the queue within each time interval, and the server makes decisions at the beginning of each time step to minimize the weighted number of the lost packets, i.e., weighted loss, over a long finite horizon.

It is assumed that at most one packet can be generated per class per unit time and each packet arriving at time t has the *same deadline* $t + d$. Each class traffic source is associated with a hidden Markov model (HMM), where the HMM's current state information is not available to the scheduler, who observes only packet arrivals.

If a heavy burst of important classes is expected, the server should stop serving relatively unimportant packets to gain smaller weighted loss (higher weighted throughput) even if the server can minimize the (unweighted) number of lost packets to achieve throughput optimality. For this type of traffic pattern, a static-priority (SP) may well be a good policy. SP serves the highest-weight class that has a packet unscheduled that is alive or pending in the buffer at each time, breaking ties by serving the earlier arriving packet. On the other hand, the server should preserve the throughput optimality by "earliest-deadline-first" (EDF) or "current-minloss" (CM) in the opposite situation or if the average load of the packet arrivals is close to one. EDF serves the class with the earliest-expiring packet unscheduled that is alive at each time, breaking ties by serving the higher class packet. CM first generates the set of packets such that serving all of the packets in the set gives the maximum weighted throughput provided that there are no future packet arrivals, and then selects a packet in the set such that CM maximizes the unweighted throughput over any finite interval.

The heuristic policies are such that each policy is near-optimal over different packet arrival patterns, where in this case, the decision maker may well wish to combine those policies to develop a single policy that somehow improves all of the heuristic policies adapting to different traffic patterns. To this end, parallel rollout can be employed with EDF, SP, and CM as candidate base-policies. However, we exclude EDF because CM is proven to be no worse than EDF for any traffic in terms of the weighted loss [72].

To apply hindsight optimization, we need to consider an additional constraint: we cannot schedule a packet before it arrives into the queue. That is, we cannot arbitrarily schedule a packet even if we have an empty slot in our scheduling plan unless the empty slot is within the packet's lifetime (from the packet's arrival time to its deadline). There exist several solution algorithms published for this off-line problem with a polynomial-time complexity in the number of packets (e.g., Peha and Tobagi's algorithm [140]). Therefore, it is straightforward to apply hindsight optimization.

We model the scheduling problem as an MDP. The state space X of this problem is infinite, given by $X = \Theta_1 \times \cdots \times \Theta_m \times \{0, 1\}^{m \times d}$. Θ_i is the set of the probability

distributions over the HMM states for Class $i \in C = \{1, \ldots, m\}$ and the last factor $\{0, 1\}^{m \times d}$ is a set of vectors, where a vector in the set represents the set of pending packets in the buffer B_t at time t, along with their laxities. If there exists a packet p whose laxity (the time remaining before its deadline) is $l_t(p)$ in the buffer at time t, then the entry of the vector indexed with $C(p)$ and $l_t(p)$ is 1, where $C(p)$ denotes the class of the packet p. The action space A is $\{1, \ldots, m\}$, where choosing an admissible action $i \in A$ corresponds to serving the pending packet in the buffer whose laxity is smallest in Class i. The buffer B_t, the set containing the packets that are in the buffer at time t, evolves as follows:

$$B_{t+1} = \left(B_t \setminus \left\{p \in B_t : l_t(p) = 1 \text{ or } p = \rho(B_t)\right\}\right) \cup \mathcal{A}_{t+1},$$

where $\rho(B_t)$ denotes the packet in B_t served by a scheduling policy ρ. B_{t+1} is thus stochastically described by the random arrivals in $\mathcal{A}_{t+1}$ during the time interval $(t, t+1)$, which are generated by the given HMMs. Therefore, the state transition function P for the MDP is defined in the obvious manner representing underlying stochastic transitions in each of the HMMs, the change in the buffer by adding new packets in $\mathcal{A}_{t+1}$ generated by the HMMs, and the expiration of unserved packets and the removal of the packet served by the action selected. The cost function is given by

$$\sum_{p: l_t(p)=1, p \in B_t - p_i} \lambda_{C(p)},$$

for $x_t \in X, i \in A(x_t)$ at time t, where $\lambda_i, i \in C$ is the weight for Class i, and p_i is the Class i packet with the smallest laxity among pending Class i packets.

For the experiment, each packet's laxity was set to be 20 ($d = 20$), seven classes (1 through 7) of packets were considered, and the weights of the seven classes were set such that Class i has a weight of $\lambda_i = \omega^{i-1}$. By decreasing the parameter ω in $[0, 1]$, we accentuate the disparity in importance between classes, making the scheduling problem more class-sensitive. We show below how performance depends on ω. Note that at the one extreme of $\omega = 0$, SP is optimal, and at the other extreme of $\omega = 1$, both EDF and CM are optimal.

Rollout with CM as the base-policy (ROCM), rollout with SP as the base-policy (ROSP), parallel rollout with CM and SP as base-policies (PARA), hindsight-optimization-based policy (HO), and CM, along with SP and EDF were tested against random traffic generated from four different sets of randomly selected HMM models for the packet arrival processes. For each set, we selected an HMM for each class from the same distribution. The HMM state space of size 3 was selected, resulting in a total of 3^7 HMM states. The HMM states were arranged in a directed cycle, and the self-transition probability for each state was chosen uniformly from the interval [0.9, 1.0], i.e., $U(0.9, 1.0)$. The packet arrival generation probability at each state was selected such that one state is "low traffic" $\sim U(0, 0.01)$, one state is "medium traffic" $\sim U(0.2, 0.5)$, and one state is "high traffic" $\sim U(0.7, 1.0)$. After randomly selecting the HMMs for each of the seven classes from the distribution, we used the stationary distribution obtained over the HMM states to normalize the

arrival generation probabilities for each class so that arrivals are (roughly) equally likely in high-cost (Classes 1 and 2), medium-cost (Classes 3 and 4), and low-cost (Classes 5–7) and to make a randomly generated traffic from the HMM set have overall arrivals at about one packet per time unit to create a difficult scheduling problem.

Each scheduling policy was run for 62,500 time periods and the "competitive ratio" achieved by each policy was measured. The competitive ratio here is the ratio between the weighted loss of a scheduling policy and the weighted loss incurred by applying Peha and Tobagi's algorithm for the same traffic. ROCM, ROSP, PARA, and HO scheduling policies used 50 sample-paths (random number sequences) and a rolling horizon of length 40 for estimating an optimal action at the current state by simulation.

Figure 5.3 shows empirical competitive ratio plots for traffic from four different sets of traffic models. We can see that PARA successfully combines the given base-policies, CM and SP, improving the performances of all of the base-policies, which shows that parallel rollout generates a single policy that adapts automatically to different trajectories of the system, in addition to alleviating the difficulty of choosing a single base-policy to be rolled out.

Figure 5.4 shows empirical competitive ratio plots for the same traffic from the same four sets of traffic models as in Fig. 5.3. For NMS, each action was sampled twice at each sampled state, i.e., sampling width of 2, and a rolling horizon of length 4 was used. To have better estimates of Q-function values, the value of following CM was used at the leaf node (state) of the sampled tree instead of zero. We can see that HO provides reasonable performance but worse than PARA. Furthermore, not surprisingly, NMS performed like SP when zero value was used at the rolling horizon of the sampling (not shown), and like ROCM or CM itself if CM was used to estimate value at the rolling horizon due to the low sampling.

Overall, all simulation-based rolling-horizon control approaches improved the performance of CM. This is more noticeable in the region of low ω values. This is because as the value of ω increases, the performance of CM gets closer to the optimal performance and there is no enough theoretical margin for improvement. The average improvement over CM by ROCM is not high, which shows that the action choice at each time is almost similar to CM itself. On the other hand, PARA outperformed or showed similar performances as all the other simulation-based approaches. At ω no bigger than 0.3, it improved CM about 30–60 %. In particular at ω very near zero, it improved CM and/or SP in the order of a magnitude. Note that, as the value of ω decreases, the complexity of the scheduling problem gets more difficult. This result is quite expected because SP is the right choice for traffic patterns for the heavy burst of important class packets, whereas CM is the right choice for the other traffic patterns, roughly speaking. Rolling out these policies at the same time makes the resulting parallel rollout approach adapt well to the right traffic pattern automatically. The hindsight-optimization-based policy HO also outperformed CM by a smaller than PARA but significant margin over a broad range of ω. Furthermore, not surprisingly, NMS performed like SP when zero value was used at the horizon of the sampling, and like ROCM or CM itself if CM was used

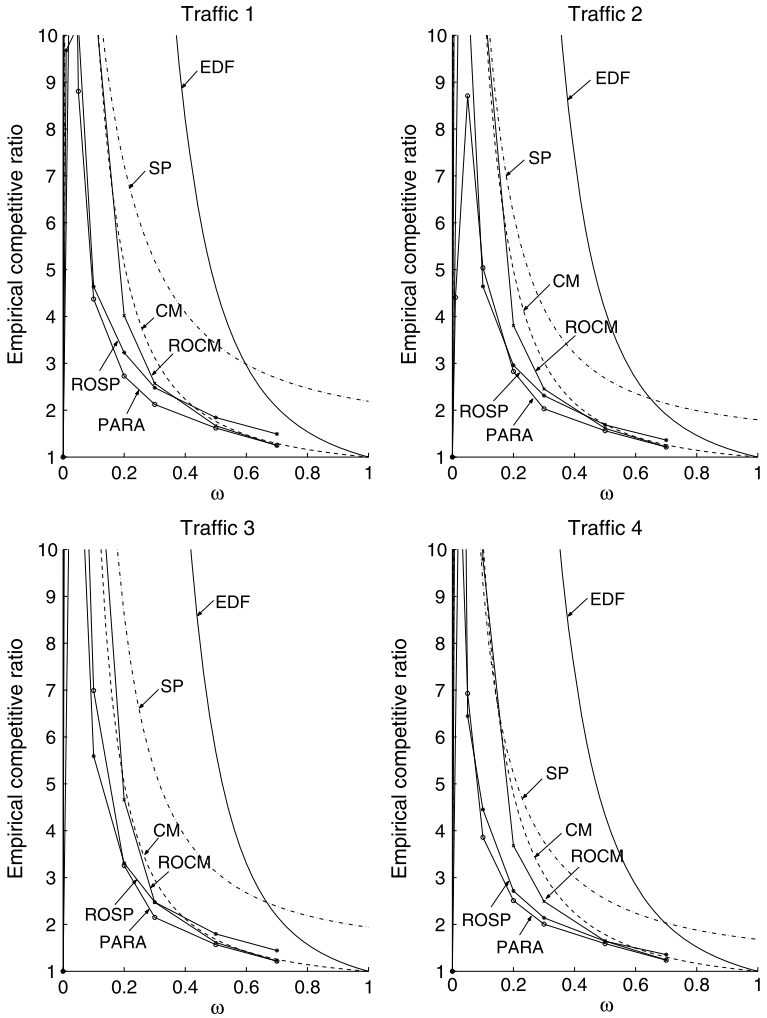


Fig. 5.3 Empirical competitive ratio plots for traffic from four different sets of traffic models, where ROCM is rollout with CM as the base-policy, ROSP is rollout with SP as the base-policy, PARA is parallel rollout with CM and SP as the base-policies, and for this competitive ratio, the off-line optimization solution algorithm in [140] was used to obtain the theoretical bound

to estimate value at the rolling horizon, due to the low rolling horizon and sampling width.

These results indicate that simulation-based on-line control via parallel rollout or hindsight optimization can be applied in practice for approximately solving an infinite-horizon MDP when a set of heuristic policies is given or an algorithm which can solve the hindsight optimization problem with a low running-time complexity can be devised.

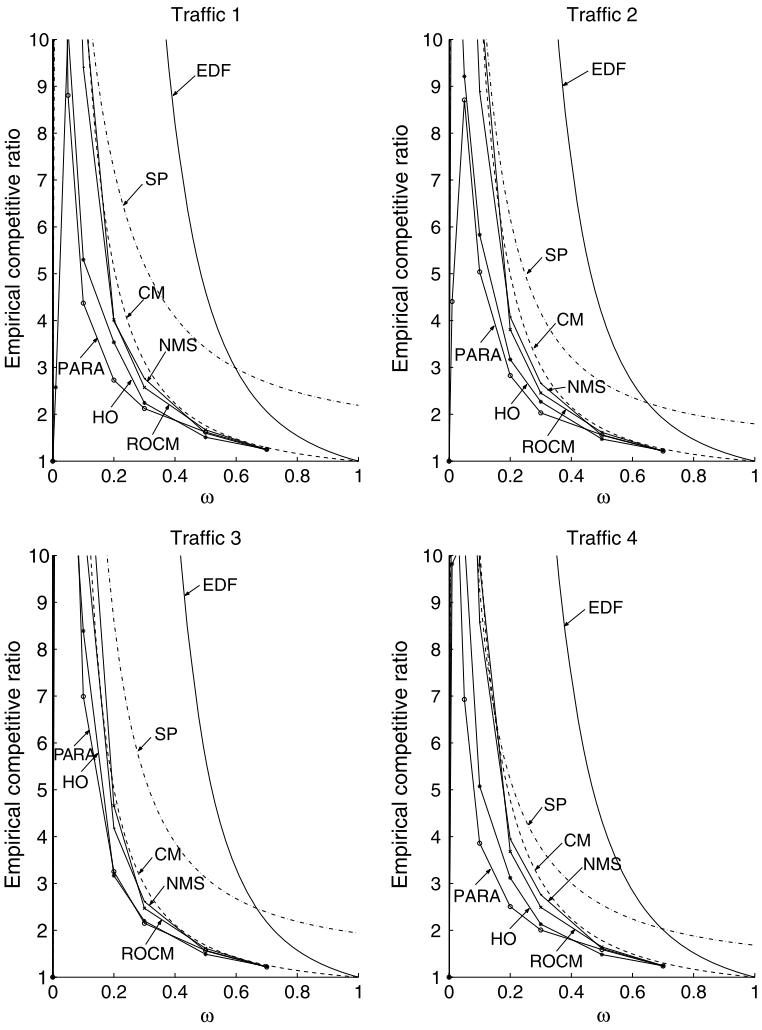


Fig. 5.4 Empirical competitive ratio plots for traffic from four different sets of traffic models, where HO refers to the hindsight-optimization-based policy, NMS used the value of applying CM at the leaf nodes, ROCM is rollout with CM as the base-policy, PARA is parallel rollout with CM and SP as the base-policies, and for this competitive ratio, the off-line optimization solution algorithm in [140] was used to obtain the theoretical bound

5.4 Approximate Stochastic Annealing

In this section, we present an on-line simulation-based algorithm called approximate stochastic annealing (ASA) for solving infinite-horizon MDPs. The algorithm combines Q-learning with the MARS algorithm of Sect. 4.6.1 to directly search the policy space. At each iteration, ASA estimates the optimal policy by sampling from

a probability distribution function over the policy space, which is then updated based on the Q-function estimates obtained via a recursion of Q-learning type. The underlying idea is to use Q-learning to evaluate the performance of the sampled policies on-line, and then use these performance estimates in a Boltzmann selection scheme to update the distribution model in a direction of performance improvement. Both ASA and the SAMW algorithm of Sect. 5.1 work with probability distributions over the policy space. However, the main difference is that SAMW focuses on estimating the optimal value function at a specified initial state and requires enumerating an entire set of policies in constructing the distribution function, whereas ASA returns the optimal value function estimates for all states in X and updates the distribution function only based on the sampled policies. We note that since ASA involves Q-learning as an integral component for policy performance evaluation, it suffers from the curse of dimensionality—an issue that we will not address.

We consider the infinite-horizon discounted reward MDP simulation model of Chap. 1 with finite state and action spaces. The objective is to identify an optimal policy $\pi^* \in \Pi_s$ that maximizes the expected total discounted reward for all initial states x, i.e.,

$$V^*(x) := V^{\pi^*}(x) = \sup_{\pi \in \Pi_s} V^{\pi}(x). \tag{5.19}$$

For simplicity and without loss of generality, we assume that every action is admissible in every state. In addition, we also assume that the optimal policy is unique and that the underlying MDP is communicating, i.e., for any two states $x, y \in X$, there exists a deterministic policy such that y is accessible from x in the Markov chain induced by the policy.

The optimal value function V^* in (5.19) satisfies Bellman's optimality equation, which can be equivalently expressed in terms of the Q-function as follows:

$$Q^*(x,a) = E\Big[R'(x,a,w) + \gamma \max_{b \in A} Q^*\big(f(x,a,w),b\big)\Big] \tag{5.20}$$

$\forall x \in X,\ a \in A$, where $Q^*(x,a) := E[R'(x,a,w) + \gamma V^*(f(x,a,w))]$. When the transition function f and/or the reward function R' are unknown, one of the most well-known methods for solving (5.20) is the Q-learning algorithm. It is an on-line model-free approach for estimating the optimal Q-function via a recursion of stochastic approximation type:

$$\begin{aligned} Q_{t+1}(x_t,a_t) = {} & \big(1-\beta_t(x_t,a_t)\big)Q_t(x_t,a_t) + \beta_t(x_t,a_t) \\ & \times \Big[R'(x_t,a_t,w_t) + \gamma \max_{b \in A} Q_t\big(f(x_t,a_t,w_t),b\big)\Big], \end{aligned} \tag{5.21}$$

where x_t, a_t, w_t are the state, action, and random simulation noise at time t, Q_t is the current estimate of the Q-function with $Q_0(x,a) = 0\ \forall x \in X,\ a \in A$, and $\beta_t(x_t,a_t)$ is the state-action dependent learning rate at time t.

In ASA, policies are constructed at each iteration t using an $|X|$-by-$|A|$ stochastic matrix q_t, whose (i,j)th entry $q_t(i,j)$ specifies the probability that state i takes

Approximate Stochastic Annealing

Input: an initial state x_0, annealing schedule $\{T_t\}_{t=0}^{\infty}$, a small constant $0 < \lambda < 1$, parameter sequences $\{\alpha_t\}_{t=0}^{\infty}$ and $\{\beta_t(i,j)\}_{t=0}^{\infty}$ satisfying $0 < \alpha_t, \beta_t(i,j) < 1\ \forall t,\ i \in X,\ j \in A$, a sample size sequence $\{N_t\}_{t=0}^{\infty}$. Set $Q_0(i,j) = 0$ and $q_0(i,j) = 1/|A|\ \forall i, j$.

Initialization: Set $t = 0$.

Loop until Stopping Rule is satisfied:

- Sample N_t policies $\Lambda_t := \{\pi^1, \pi^2, \ldots, \pi^{N_t}\}$ from $\hat{\phi}(\pi, q_t) = (1-\lambda)\phi(\pi, q_t) + \lambda\phi(\pi, q_0)$ as follows: for each $l = 1, \ldots, N_t$, with probability $1-\lambda$, construct a policy π^l using the stochastic matrix q_t; with probability λ, construct π^l using q_0.
- For each action j, denote by $C_{t,j}$ the number of times j has been sampled, i.e., $C_{t,j} = \sum_{\pi\in\Lambda_t} I\{\pi(x_t) = j\}$. Update the Q-function estimates:

$$Q_{t+1}(x_t, j) = \big(1 - \beta_t(x_t, j)\big)Q_t(x_t, j) + \frac{\beta_t(x_t, j)}{C_{t,j}} \times \sum_{k=1}^{C_{t,j}} \Big[R'\big(x_t, j, w_{t,j}^k\big) + \gamma \max_{a\in A} Q_t\big(f\big(x_t, j, w_{t,j}^k\big), a\big)\Big] \tag{5.22}$$

  whenever $C_{t,j} > 0$, and $Q_{t+1}(x_t, j) = Q_t(x_t, j)$ whenever $C_{t,j} = 0$, where $w_{t,j}^k$ is the random simulation noise the kth time when action j is taken.
- Update matrix q_t by

$$q_{t+1}(i,j) = (1-\alpha_t)q_t(i,j) + \alpha_t \frac{\sum_{\pi\in\Lambda_t} e^{\sum_{s=1}^{|X|} \frac{Q_t(s,\pi(s))}{T_t}} \hat{\phi}^{-1}(\pi, q_t) I\{\pi \in \Pi_{i,j}\}}{\sum_{\pi\in\Lambda_t} e^{\sum_{s=1}^{|X|} \frac{Q_t(s,\pi(s))}{T_t}} \hat{\phi}^{-1}(\pi, q_t)}. \tag{5.23}$$

- Sample a policy $\bar{\pi}$ from $\hat{\phi}(\pi, q_{t+1})$. Apply action $a_t = \bar{\pi}(x_t)$ at the current state x_t and observe a new state x_{t+1}. $t \leftarrow t+1$.

Output: $q_t,\ Q_t$.

Fig. 5.5 Description of ASA algorithm for infinite-horizon MDPs

action j. For a given q_t, define a probability mass function over the policy space Π_s: $\phi(\pi, q_t) := \prod_{i=1}^{|X|}\prod_{j=1}^{|A|}[q_t(i,j)]^{I\{\pi\in\Pi_{i,j}\}}\ \forall\pi \in \Pi_s$, where $\Pi_{i,j} := \{\pi : \pi(i) = j\}$ is the set of stationary policies that assign action j to state i. We now provide a detailed description of each iteration step in the algorithm presented in Fig. 5.5.

At the beginning of each iteration, policies are drawn from a mixture of $\phi(\pi, q_t)$ and the initial distribution $\phi(\pi, q_0)$, where the mixing intensity is determined by an initially specified parameter λ. To simplify exposition, we have set λ to a constant; other non-constant, state-dependent choices of λ may also be used.

Given the current state x_t, the decision maker interacts with an on-line simulation model, tries out all actions $\pi(x_t)$, $\pi \in \Lambda_t$ specified by the set of policies Λ_t chosen in the previous step, receives the random reward $R'(x_t, \pi(x_t), w_t)$, and moves to the next (simulated) state $f(x_t, \pi(x_t), w_t)$. The Q-values at the visited state-action pairs are then updated accordingly based on a slight variant of the Q-learning recursion (5.21). Note that since ASA is a population-based approach, a particular action $j \in A$ may be sampled multiple times $C_{t,j}$. Therefore in (5.22), there is an averaging over $C_{t,j}$ replications in addition to the stochastic averaging used in (5.21).

The performance of each sampled policy is evaluated as the sum of the current Q-function estimates under the policy across all states in X, i.e., $\sum_{s=1}^{|X|} Q_t(s, \pi(s))$. These performance estimates are then used in a Boltzmann selection scheme to update the probability matrix q_t. Note that given q_t and Λ_t, the second term (without α_t) in the right-hand-side of (5.23) can be written as an expectation of an indicator function $\sum_{\pi \in \Lambda_t} \bar{g}_t(\pi) I\{\pi \in \Pi_{i,j}(t)\}$, where

$$\bar{g}_t(\pi) := \frac{\exp(\sum_{s=1}^{|X|} Q_t(s, \pi(s))/T_t)\hat{\phi}^{-1}(\pi, q_t)}{\sum_{\pi' \in \Lambda_t} \exp(\sum_{s=1}^{|X|} Q_t(s, \pi'(s))/T_t)\hat{\phi}^{-1}(\pi', q_t)} \quad \forall \pi \in \Lambda_t$$

is an empirical estimate of the true Boltzmann mass function

$$g_t(\pi) = \frac{e^{\sum_{s=1}^{|X|} Q^*(s,\pi(s))/T_t}}{\sum_{\pi' \in \Pi} e^{\sum_{s=1}^{|X|} Q^*(s,\pi'(s))/T_t}} \quad \forall \pi \in \Pi_s, \tag{5.24}$$

which becomes concentrated on promising policies in Λ_t as T_t decreases to zero. Thus, each π in Λ_t is weighted by $\bar{g}_t(\pi)$ in updating q_t.

Finally, the decision maker randomly generates a policy from the updated distribution $\hat{\phi}(\pi, q_{t+1})$, adopts the action stipulated by the sampled policy, observes a new state x_{t+1}, and moves on to the next iteration.

Note that Eq. (5.23) can be written in the following form:

$$q_{t+1}(i,j) - q_t(i,j) = \alpha_t \left[\sum_{\pi \in \Lambda_t} \bar{g}_t(\pi) I\{\pi \in \Pi_{i,j}\} - q_t(i,j) \right]. \tag{5.25}$$

Since $\bar{g}_t$ puts more weight on policies with better performance, it is easy to see that the stochastic matrix q_{t+1} is updated in the averaged direction of (estimated) promising policies in Λ_t.

5.4.1 Convergence Analysis

Let $\mathcal{F}_t$ be the σ-field generated by the set of all sampled policies and random realization of stochastic uncertainty up to time $t-1$, i.e., $\mathcal{F}_t = \sigma(x_0, \Lambda_0, \mathbf{w}_0, \ldots, \Lambda_{t-1}, \mathbf{w}_{t-1}, x_t)$, where $\mathbf{w}_t := \{w_{t,j}^k, k = 1, \ldots, C_{t,j}, \ j = 1, \ldots, |A|\}$. Note that given $\mathcal{F}_t$,

both Q_t and q_t are completely determined. Thus, we use $P(\cdot|\mathcal{F}_t)$ and $E[\cdot|\mathcal{F}_t]$ to denote the conditional probability and expectation taken with respect to $\hat{\phi}(\pi, q_t)$ and the distribution of system uncertainty $\mathbf{w}_t$ at the tth iteration. To simplify the exposition, the following shorthand notations will be used:

$$
\begin{aligned}
Y_t(i,j) &= \sum_{\Pi} e^{\sum_{s=1}^{|X|} Q^*(s,\pi(s))/T_t} I\{\pi \in \Pi_{i,j}\},\\
Z_t &= \sum_{\Pi} e^{\sum_{s=1}^{|X|} Q^*(s,\pi(s))/T_t},\\
\hat{Y}_t(i,j) &= \frac{1}{N_t}\sum_{\Lambda_t} e^{\frac{\sum_{s=1}^{|X|} Q^*(s,\pi(s))}{T_t}} \hat{\phi}^{-1}(\pi, q_t) I\{\pi \in \Pi_{i,j}\},\\
\hat{Z}_t &= \frac{1}{N_t}\sum_{\Lambda_t} e^{\frac{\sum_{s=1}^{|X|} Q^*(s,\pi(s))}{T_t}} \hat{\phi}^{-1}(\pi, q_t),\\
\bar{Y}_t(i,j) &= \frac{1}{N_t}\sum_{\Lambda_t} e^{\frac{\sum_{s=1}^{|X|} Q_t(s,\pi(s))}{T_t}} \hat{\phi}^{-1}(\pi, q_t) I\{\pi \in \Pi_{i,j}\},\\
\bar{Z}_t &= \frac{1}{N_t}\sum_{\Lambda_t} e^{\frac{\sum_{s=1}^{|X|} Q_t(s,\pi(s))}{T_t}} \hat{\phi}^{-1}(\pi, q_t).
\end{aligned}
$$

To analyze ASA, we rewrite (5.25) in the following recursive form:

$$\eta_{t+1}(i,j) = \eta_t(i,j) - \xi_t(i,j),$$

where $\eta_t(i,j) := q_t(i,j) - I\{\pi^* \in \Pi_{i,j}\}$ and $\xi_t(i,j) = \alpha_t[\eta_t(i,j) + I\{\pi^* \in \Pi_{i,j}\} - \frac{\hat{Y}_t(i,j)}{\hat{Z}_t} + \frac{\hat{Y}_t(i,j)}{\hat{Z}_t} - \frac{\bar{Y}_t(i,j)}{\bar{Z}_t}]$.

The main convergence result is stated in Theorem 5.18. Its proof is based on the following intermediate results. Lemma 5.13 shows that the sequence of idealized Boltzmann distribution $\{g_t\}$ converges to a limiting distribution that concentrates only on the optimal policy π^*.

Lemma 5.13 *If $T_t \to 0$ as $t \to \infty$, then $E_{g_t}[I\{\pi \in \Pi_{i,j}\}] \to I\{\pi^* \in \Pi_{i,j}\}$ as $t \to \infty$.*

Proof Note that

$$
\begin{aligned}
&\left|E_{g_t}\left[I\{\pi \in \Pi_{i,j}\}\right] - I\left\{\pi^* \in \Pi_{i,j}\right\}\right|\\
&\quad \le E_{g_t}\left[\left|I\{\pi \in \Pi_{i,j}\} - I\left\{\pi^* \in \Pi_{i,j}\right\}\right|\right]\\
&\quad = \sum_{\pi \neq \pi^*} \left|I\{\pi \in \Pi_{i,j}\} - I\left\{\pi^* \in \Pi_{i,j}\right\}\right| \times \frac{e^{\sum_{s=1}^{|X|} Q^*(s,\pi(s))/T_t}}{\sum_{\pi' \in \Pi} e^{\sum_{s=1}^{|X|} Q^*(s,\pi'(s))/T_t}}
\end{aligned}
$$

$$\leq \sum_{\pi \neq \pi^*} \frac{e^{\sum_{s=1}^{|X|}(Q^*(s,\pi(s))-V^*(s))/T_t}}{1+\sum_{\pi' \neq \pi^*} e^{\sum_{s=1}^{|X|}(Q^*(s,\pi'(s))-V^*(s))/T_t}}$$

$$\leq \sum_{\pi \neq \pi^*} e^{\sum_{s=1}^{|X|}(Q^*(s,\pi(s))-V^*(s))/T_t},$$

which converges to zero as $T_t \to 0$, since $\sum_{s=1}^{|X|}(Q^*(s,\pi(s)) - V^*(s)) < 0$ for all $\pi \neq \pi^*$. □

The next lemma states that $\frac{\hat{Y}_t(i,j)}{\hat{Z}_t}$ is an asymptotically unbiased estimator of $\frac{Y_t(i,j)}{Z_t}$.

Lemma 5.14 *If $N_t \to \infty$ as $t \to \infty$, then $|E[\frac{\hat{Y}_t(i,j)}{\hat{Z}_t}|\mathcal{F}_t] - \frac{Y_t(i,j)}{Z_t}| \to 0$ as $t \to \infty$ w.p.*1.

Proof Note that

$$\frac{\hat{Y}_t(i,j)}{\hat{Z}_t} - \frac{Y_t(i,j)}{Z_t} = \frac{\hat{Y}_t(i,j)}{\hat{Z}_t} - \frac{\hat{Y}_t(i,j)}{Z_t} + \frac{\hat{Y}_t(i,j)}{Z_t} - \frac{Y_t(i,j)}{Z_t}$$

$$= \frac{\hat{Y}_t(i,j)}{\hat{Z}_t}\frac{Z_t - \hat{Z}_t}{Z_t} + \frac{1}{Z_t}\big(\hat{Y}_t(i,j) - Y_t(i,j)\big).$$

Taking conditional expectations on both sides yields

$$\left|E\left[\frac{\hat{Y}_t(i,j)}{\hat{Z}_t}\middle|\mathcal{F}_t\right] - \frac{Y_t(i,j)}{Z_t}\right|$$

$$= \left|E\left[\frac{\hat{Y}_t(i,j)}{\hat{Z}_t}\frac{Z_t - \hat{Z}_t}{Z_t}\middle|\mathcal{F}_t\right]\right|$$

$$\leq \frac{1}{|Z_t|}E\big[|Z_t - \hat{Z}_t|\big|\mathcal{F}_t\big] \quad \text{since } \left|\frac{\hat{Y}_t(i,j)}{\hat{Z}_t}\right| \leq 1$$

$$\leq \frac{1}{|Z_t|}E\big[(Z_t - \hat{Z}_t)^2\big|\mathcal{F}_t\big]^{1/2} \quad \text{by Hölder's inequality}$$

$$\leq \frac{1}{|Z_t|}\frac{1}{\sqrt{N_t}}E\big[e^{2\sum_{s=1}^{|X|}Q^*(s,\pi(s))/T_t}\hat{\phi}^{-2}(\pi,q_t)\big|\mathcal{F}_t\big]^{1/2}$$

$$= \frac{1}{\sqrt{N_t}}\left(\sum_{\Pi}\hat{\phi}^{-1}(\pi,q_t)\left(\frac{e^{\sum_{s=1}^{|X|}Q^*(s,\pi(s))/T_t}}{Z_t}\right)^2\right)^{1/2}$$

$$\leq \frac{1}{\sqrt{N_t}}\left(\sum_{\Pi}\lambda^{-1}\phi^{-1}(\pi,q_0)g_t(\pi)\right)^{1/2}$$
$$\leq \frac{(|A|^{|X|}/\lambda)^{1/2}}{\sqrt{N_t}}.$$

Thus, the desired result follows by taking $N_t \to \infty$. □

We also need the following result.

Lemma 5.15 *At the tth iteration of algorithm ASA, the probability that action $j \in A$ will be sampled at most $\frac{\lambda N_t}{|A|} - 2\sqrt{N_t \ln t}$ times is upper bounded by $\frac{1}{t^2}$, i.e., $P(C_{t,j} \leq \frac{\lambda N_t}{|A|} - 2\sqrt{N_t \ln t}|\mathcal{F}_t) \leq \frac{1}{t^2}$.*

Proof Let $j \in A$. For a given number N_t of samples at the tth iteration of ASA, define the indicator function $I_k = 1$ if an action other than j is obtained at the kth sample; $I_k = 0$ otherwise. It follows that $\mu_k := E[I_k|\mathcal{F}_t] \leq 1 - \frac{\lambda}{|A|}$ and $|I_k - \mu_k| \leq 1$. Let m be a positive integer such that $m \leq \frac{\lambda N_t}{|A|}$ for t sufficiently large, we have, from Hoeffding's inequality,

$$P(C_{t,j} \leq m|\mathcal{F}_t) = P\left(\sum_{k=1}^{N_t} I_k > N_t - m\middle|\mathcal{F}_t\right)$$
$$\leq P\left(\frac{1}{N_t}\sum_{k=1}^{N_t}(I_k - \mu_k) > \frac{1}{N_t}\left(\frac{\lambda N_t}{|A|} - m\right)\middle|\mathcal{F}_t\right)$$
$$\leq \exp\left(-\frac{2(\lambda N_t/|A| - m)^2}{4N_t}\right).$$

Finally, setting $\exp(-\frac{(\lambda N_t/|A|-m)^2}{2N_t}) = 1/t^2$ and solving for m yield $m = \lambda N_t/|A| - 2\sqrt{N_t \ln t}$ as required. □

Moreover, under some appropriate conditions on $\{T_t\}$ and $\{\beta_t(i,j)\}$, the Q-function estimates generated by (5.22) converge to Q^* with rate at least $o(T_t)$.

Lemma 5.16 *Assume the following conditions hold*:

(a) $\frac{\lambda N_t}{|A|} - 2\sqrt{N_t \ln t} \to \infty$;
(b) $T_t \geq \frac{K}{\min\{(\lambda N_t/|A| - 2\sqrt{N_t \ln t})^{1/2}, t\}}$ $\forall t$ *for some constant* $K > 0$;
(c) $0 \leq \beta_t(i,j) \leq 1$, $\sum_{t=0}^{\infty}\beta_t(i,j) = \infty$, $\sum_{t=0}^{\infty}\beta_t^2(i,j) < \infty$ *w.p.*1, *where* $\beta_t(i,j) = 0$ *unless* $i = x_t$ *and* $j \in \{\pi(x_t) : \pi \in \Lambda_t\}$;
(d) $\frac{1}{\beta_t(i,j)}(\frac{1}{T_{t+1}} - \frac{1}{T_t}) \to 0$ *w.p.*1 *as* $t \to \infty$ *for* $i = x_t$ *and* $j \in \{\pi(x_t) : \pi \in \Lambda_t\}$.

Then $\lim_{t\to\infty} \frac{Q_t(i,j) - Q^*(i,j)}{T_t} = 0$ $\forall i \in X$ $j \in A$ *w.p.*1.

Proof First we argue that under algorithm ASA, all state-action pairs will be visited infinitely often (i.o.) as $t \to \infty$ w.p.1. Fix a state $i \in X$, let $N_t(i)$ be the number of visits to state i by time t and $t_k(i)$ be the time of the kth visit to state i. For any $j \in A$, note that $\sum_{k=1}^{N_t(i)} P(a_{t'} = j | t_k(i) = t') \geq \sum_{k=1}^{N_t(i)} \lambda/|A| = \infty$ whenever $\lim_{t\to\infty} N_t(i) = \infty$ w.p.1. This, when combined with the earlier communication assumption on the MDP and Lemma 4 in [165], indicates that all state-action pairs will be visited i.o. w.p.1.

Define $\zeta_t(x_t, j) = (Q_t(x_t, j) - Q^*(x_t, j))/T_t$; dividing both sides of (5.22) by T_t, we have

$$\begin{aligned}
\zeta_{t+1}(x_t, j) &= \big(1 - \beta_t(x_t, j)\big)\zeta_t(x_t, j)\left(\frac{1}{T_{t+1}} - \frac{1}{T_t}\right)\big(Q_{t+1}(x_t, j) - Q^*(x_t, j)\big) \\
&\quad + \frac{\beta_t(x_t, j)}{T_t}\frac{1}{C_{t,j}}\sum_{k=1}^{C_{i,j}}\Big[R\big(x_t, j, w_{t,j}^k\big) \\
&\quad + \gamma \max_{a\in A} Q_t\big(f\big(x_t, j, w_{t,j}^k\big), a\big) - Q^*(x_t, j)\Big] \\
&= \big(1 - \beta_t(x_t, j)\big)\zeta_t(x_t, j) + \beta_t(x_t, j)\Bigg\{\zeta_t(x_t, j) \\
&\quad + \frac{1}{T_t C_{t,j}}\sum_{k=1}^{C_{t,j}}\Big[R\big(x_t, j, w_{t,j}^k\big) \\
&\quad + \gamma \max_{a\in A} Q_t\big(f\big(x_t, j, w_{t,j}^k\big), a\big) - Q_t(x_t, j)\Big] \\
&\quad + \frac{1}{\beta_t(x_t, j)}\left(\frac{1}{T_{t+1}} - \frac{1}{T_t}\right)\big(Q_{t+1}(x_t, j) - Q^*(x_t, j)\big)\Bigg\} \\
&= \big(1 - \beta_t(x_t, j)\big)\zeta_t(x_t, j) \\
&\quad + \beta_t(x_t, j)\big[H_t(x_t, j) + M_t(x_t, j) + G_t(x_t, j)\big],
\end{aligned}$$

where

$$\begin{aligned}
H_t(x_t, j) &:= \zeta_t(x_t, j) + \frac{1}{T_t C_{t,j}}\sum_{k=1}^{C_{t,j}}\Big[R\big(x_t, j, w_{t,j}^k\big) \\
&\quad + \gamma \max_{a\in A} Q^*\big(f\big(x_t, j, w_{t,j}^k\big), a\big) - Q_t(x_t, j)\Big], \\
M_t(x_t, j) &:= \frac{\gamma}{T_t C_{t,j}}\sum_{k=1}^{C_{t,j}}\Big[\max_{a\in A} Q_t\big(f\big(x_t, j, w_{t,j}^k\big), a\big) \\
&\quad - \max_{a\in A} Q^*\big(f\big(x_t, j, w_{t,j}^k\big), a\big)\Big],
\end{aligned}$$

$$G_t(x_t, j) := \frac{1}{\beta_t(x_t, j)}\left(\frac{1}{T_{t+1}} - \frac{1}{T_t}\right)\big(Q_{t+1}(x_t, j) - Q^*(x_t, j)\big).$$

We have by (5.20), $E[H_t(x_t, j)|\mathcal{F}_t] = E[E[H_t(x_t, j)|\Lambda_t]|\mathcal{F}_t] = E[\zeta_t(x_t, j) + \frac{1}{T_t}(Q^*(x_t, j) - Q_t(x_t, j))|\mathcal{F}_t] = 0$. Furthermore, it is easy to show that

$$E\big[H_t^2(x_t, j)\big|\mathcal{F}_t\big] = E\Bigg[\frac{1}{T_t^2 C_{t,j}}\operatorname{Var}\Big(R\big(x_t, j, w_{t,j}^k\big) + \gamma \max_{a\in A} Q^*\big(f\big(x_t, j, w_{t,j}^k\big), a\big)\big|\Lambda_t\Big)\Bigg|\mathcal{F}_t\Bigg].$$

Let $\mathcal{B}_t = \{C_{t,j} \le \frac{\lambda N_t}{|A|} - 2\sqrt{N_t \ln t}\}$. Since R is uniformly bounded, it immediately follows that, for some constant $K_1 > 0$,

$$\begin{aligned}
E\big[H_t^2(x_t, j)\big|\mathcal{F}_t\big] &\le K_1 E\left[\frac{1}{T_t^2 C_{t,j}}\middle|\mathcal{F}_t\right] \\
&= K_1 E\left[\frac{1}{T_t^2 C_{t,j}} I\{\mathcal{B}_t\}\middle|\mathcal{F}_t\right] + K_1 E\left[\frac{1}{T_t^2 C_{t,j}} I\big\{\mathcal{B}_t^c\big\}\middle|\mathcal{F}_t\right] \\
&\le K_1 \frac{1}{T_t^2} P(\mathcal{B}_t|\mathcal{F}_t) + K_1 \frac{1}{T_t^2(\frac{\lambda N_t}{|A|} - 2\sqrt{N_t \ln t})} \\
&\le K_1\left(\frac{1}{t^2 T_t^2} + \frac{1}{T_t^2(\frac{\lambda N_t}{|A|} - 2\sqrt{N_t \ln t})}\right) \quad \text{by Lemma 5.15} \\
&\le \frac{2K_1}{K^2} \quad \text{by condition (b).}
\end{aligned}$$

On the other hand, we have

$$\begin{aligned}
\big|M_t(x_t, j)\big| &\le \frac{\gamma}{T_t C_{t,j}} \sum_{k=1}^{C_{t,j}} \Big|\max_{a\in A} Q_t\big(f\big(x_t, j, w_{t,j}^k\big), a\big) - \max_{a\in A} Q^*\big(f\big(x_t, j, w_{t,j}^k\big), a\big)\Big| \\
&\le \frac{\gamma}{T_t C_{t,j}} \sum_{k=1}^{C_{t,j}} \max_{a\in A}\big|Q_t\big(f\big(x_t, j, w_{t,j}^k\big), a\big) - Q^*\big(f\big(x_t, j, w_{t,j}^k\big), a\big)\big| \\
&\le \gamma \max_{x\in X, a\in A}\big|\zeta_t(x, a)\big|.
\end{aligned}$$

Next, by using an argument similar to the proof of the convergence of Q-learning, it is easy to see the iterates Q_t generated by recursion (5.22) remain bounded dur-

ing the updating process. Therefore, we have $|G_t(x_t, \pi(x_t))| \to 0$ w.p.1 by condition (d). Finally, since all state-action pairs are visited infinitely often w.p.1, a direct application of Proposition 4.5 in [17] yields $\zeta_t(i, j) \to 0\ \forall i, j$ w.p.1, which shows the desired result. □

Lemma 5.16 gives rise to the following result, indicating that the conditional expectation of the error term $\frac{\hat{Y}_t(i,j)}{\hat{Z}_t} - \frac{\bar{Y}_t(i,j)}{\bar{Z}_t}$ caused by simulation noise vanishes to zero asymptotically.

Lemma 5.17 *Assume all conditions in Lemma 5.16 hold. Then* $E[|\frac{\hat{Y}_t(i,j)}{\hat{Z}_t} - \frac{\bar{Y}_t(i,j)}{\bar{Z}_t}||\mathcal{F}_t] \to 0$ *w.p.*1.

Proof Let

$$\hat{g}_t(\pi) = \frac{e^{\sum_{s=1}^{|X|} \frac{Q^*(s,\pi(s))}{T_t}} \hat{\phi}^{-1}(\pi, q_t)}{\sum_{\pi' \in \Lambda_t} e^{\sum_{s=1}^{|X|} \frac{Q^*(s,\pi'(s))}{T_t}} \hat{\phi}^{-1}(\pi', q_t)} \quad \forall \pi \in \Lambda_t.$$

We have

$$\begin{aligned}
&\left|\frac{\hat{Y}_t(i,j)}{\hat{Z}_t} - \frac{\bar{Y}_t(i,j)}{\bar{Z}_t}\right| \\
&= \left|\frac{\hat{Y}_t(i,j) - \bar{Y}_t(i,j)}{\hat{Z}_t} + \frac{\bar{Y}_t(i,j)}{\bar{Z}_t}\frac{\bar{Z}_t - \hat{Z}_t}{\hat{Z}_t}\right| \\
&\le \frac{1}{|\hat{Z}_t|}\left(\left|\hat{Y}_t(i,j) - \bar{Y}_t(i,j)\right| + |\bar{Z}_t - \hat{Z}_t|\right) \\
&\le \frac{1}{|\hat{Z}_t|}\frac{2}{N_t}\sum_{\Lambda_t}\left|e^{\frac{1}{T_t}\sum_{s=1}^{|X|} Q^*(s,\pi(s))} - e^{\frac{1}{T_t}\sum_{s=1}^{|X|} Q_t(s,\pi(s))}\right|\hat{\phi}^{-1}(\pi, q_t) \\
&= 2\sum_{\Lambda_t} \frac{e^{\frac{1}{T_t}\sum_{s=1}^{|X|} Q^*(s,\pi(s))}\hat{\phi}^{-1}(\pi, q_t)}{N_t|\hat{Z}_t|} \\
&\quad \times \left|1 - e^{\frac{1}{T_t}\sum_{s=1}^{|X|}(Q_t(s,\pi(s)) - Q^*(s,\pi(s)))}\right| \\
&= 2\sum_{\Lambda_t} \hat{g}_t(\pi)\left|1 - e^{\frac{1}{T_t}\sum_{s=1}^{|X|}(Q_t(s,\pi(s)) - Q^*(s,\pi(s)))}\right|.
\end{aligned}$$

Therefore,

$$E\left[\left|\frac{\hat{Y}_t(i,j)}{\hat{Z}_t} - \frac{\bar{Y}_t(i,j)}{\bar{Z}_t}\right|\middle|\mathcal{F}_t\right] \le 2\max_{a \in A}\left|1 - \exp\left(\sum_{s=1}^{|X|}(Q_t(s,a) - Q^*(s,a))/T_t\right)\right|,$$

which approaches zero as $t \to \infty$ w.p.1 by Lemma 5.16. □

We have the following main convergence theorem for ASA.

Theorem 5.18 *Assume all conditions in Lemma 5.16 are satisfied. If in addition,* $T_t \to 0$ *as* $t \to \infty$ *and* $\alpha_t > 0$, $\sum_{t=0}^{\infty} \alpha_t = \infty$, *and* $\sum_{t=0}^{\infty} \alpha_t^2 < \infty$, *then* $q_t(i, j) \to I\{\pi^* \in \Pi_{i,j}\}\ \forall ij$ *as* $t \to \infty$ *w.p.*1.

Proof We consider the recursion $\eta_{t+1}(i, j) = \eta_t(i, j) - \xi_t(i, j)$ and define $U_t(i, j) = E[\xi_t(i, j)|\mathcal{F}_t]$ and $\Delta_t(i, j) = \xi_t(i, j) - U_t(i, j)$. To show the desired result, we establish that conditions (i)–(iv) in [56] hold.

(i) First we show that, for any $\epsilon > 0$, $P(|\eta_t(i, j)| > \epsilon, \eta_t(i, j)U_t(i, j) < 0 \text{ i.o.}) = 0$. We have

$$U_t(i, j) = \alpha_t \Bigg[\eta_t(i, j) + I\{\pi^* \in \Pi_{i,j}\} - \frac{Y_t(i, j)}{Z_t} + \frac{Y_t(i, j)}{Z_t} - E\left[\frac{\hat{Y}_t(i, j)}{\hat{Z}_t} \middle| \mathcal{F}_t \right] + E\left[\frac{\hat{Y}_t(i, j)}{\hat{Z}_t} - \frac{\bar{Y}_t(i, j)}{\bar{Z}_t} \middle| \mathcal{F}_t \right] \Bigg]. \tag{5.26}$$

Therefore,

$$\eta_t(i, j)U_t(i, j) = \alpha_t \Bigg[\eta_t^2(i, j) + \eta_t(i, j) \left(I\{\pi^* \in \Pi_{i,j}\} - \frac{Y_t(i, j)}{Z_t} \right) + \eta_t(i, j) \left(\frac{Y_t(i, j)}{Z_t} - E\left[\frac{\hat{Y}_t(i, j)}{\hat{Z}_t} \middle| \mathcal{F}_t \right] \right) + \eta_t(i, j) \left(E\left[\frac{\hat{Y}_t(i, j)}{\hat{Z}_t} - \frac{\bar{Y}_t(i, j)}{\bar{Z}_t} \middle| \mathcal{F}_t \right] \right) \Bigg].$$

Since $\eta_t(i, j)$ is bounded, the second term above vanishes to zero by Lemma 5.13, whereas the third and last terms both go to zero w.p.1 by Lemma 5.14 and Lemma 5.17. Thus, for almost every sample path generated by the algorithm, we have $\eta_k(i, j)U_t(i, j) > 0$ whenever $\eta_t(i, j) > \epsilon$ for t sufficiently large, which implies $P(|\eta_t(i, j)| > \epsilon, \eta_t(i, j)U_t(i, j) < 0 \text{ i.o.}) = 0$.

(ii) Since all terms in the square bracket of (5.26) are bounded and $\alpha_t \to 0$ by assumption, we have $|U_t(i, j)|(1 + |\eta_t(i, j)|)^{-1} \to 0$ as $t \to \infty$.

(iii) We have $\Delta_t(i, j) = \alpha_t[E[\frac{\bar{Y}_t(i,j)}{\bar{Z}_t}|\mathcal{F}_t] - \frac{\bar{Y}_t(i,j)}{\bar{Z}_t}]$. Since $\frac{\bar{Y}_t(i,j)}{\bar{Z}_t}$ is bounded and $\sum_t \alpha_t^2 < \infty$, it follows that $\sum_{t=1}^{\infty} E[|\Delta_t(i, j)|^2] = \sum_{t=1}^{\infty} \alpha_t^2 E[(E[\frac{\bar{Y}_t(i,j)}{\bar{Z}_t}|\mathcal{F}_t] - \frac{\bar{Y}_t(i,j)}{\bar{Z}_t})^2] < \infty$.

(iv) Finally, we show that $P(\liminf_{t\to\infty} |\eta_t(i, j)| > 0, \sum_{t=1}^{\infty} |U_t(i, j)| < \infty) = 0$. From (5.26), we have $|U_t(i, j)| \geq \alpha_t[|\eta_t(i, j)| - |I\{\pi^* \in \Pi_{i,j}\} - \frac{Y_t(i,j)}{Z_t}| - |\frac{Y_t(i,j)}{Z_t} - E[\frac{\hat{Y}_t(i,j)}{\hat{Z}_t}|\mathcal{F}_t]| - E[|\frac{\hat{Y}_t(i,j)}{\hat{Z}_t} - \frac{\bar{Y}_t(i,j)}{\bar{Z}_t}||\mathcal{F}_t]]$. Let $\Omega_1 = \{\liminf_t |\eta_t(i, j)| > 0\}$. For every sample path $\omega \in \Omega_1$, there exists a constant $\delta(\omega) > 0$ such that $\liminf_t |\eta_t(i, j)| > \delta(\omega)$. Consequently, by Lemmas 5.13, 5.14, and 5.17, for almost every $\omega \in \Omega_1$, we can find a $K(\omega) > 0$ such that $|U_t(i, j)| \geq \alpha_t \delta(\omega)/2$

for $t \geq K(\omega)$. Since $\sum_{t=0}^{\infty} \alpha_t = \infty$, it follows that, for almost all $\omega \in \Omega_1$, $\sum_{t=0}^{\infty} |U_t(i,j)| \geq \sum_{t=K(\omega)}^{\infty} |U_t(i,j)| \geq \frac{\delta(\omega)}{2} \sum_{t=K(\omega)}^{\infty} \alpha_t = \infty$, which shows condition (iv).

Finally, combining (i)–(iv) and applying the main theorem in [56] yield $\eta_t \to 0$ w.p.1. □

5.4.2 Numerical Example

We consider the infinite-horizon version ($H = \infty$) of the inventory control problem of Sect. 4.6.2 with the following two sets of parameters: (1) initial state $x_0 = 5$, setup cost $K = 5$, penalty cost $p = 10$, holding cost $h = 1$; (2) $x_0 = 5$, $K = 5$, $p = 1$, and $h = 1$. For a given initial inventory level x_0, the objective is to minimize the expected total discounted cost over the set of all stationary Markovian policies, i.e., $\min_{\pi \in \Pi_s} E[\sum_{t=0}^{\infty} \gamma^t (K I\{\pi(x_t) > 0\} + h(x_t + \pi(x_t) - D_t)^+ + p(D_t - x_t - \pi(x_t))^+) | x_0 = x]$, where the discount factor $\gamma = 0.95$.

In our experiments, a logarithmic annealing schedule $T_t = 10/\ln(1+t)$ is used. The Q-learning rate is taken to be $\beta_t(i,j) = 5/(100 + N_t(i,j))^{0.501}$, where recall that $N_t(i,j) = \sum_{l=1}^{t} I\{i = x_l, j \in \{\pi(x_l) : \pi \in \Lambda_l\}\}$. The gain in (5.23) is taken to be $\alpha_t = 0.1/(t+100)^{0.501}$ with a large stability constant 100 and a slow decay rate of 0.501. In addition, the numerator constant is set to a small number, 0.1. This is because the q matrix is updated in early iterations based on unreliable estimates of the Q-values. Therefore, it is intuitive that the initial gains in (5.23) should be kept relatively small to make the algorithm less sensitive to the misinformation induced by (5.22). The other parameters are as follows: $\lambda_t = 0.01$ and sample size $N_t = \lfloor \ln^2(1+t) \rfloor$, where $\lfloor c \rfloor$ is the largest integer no greater than c. Note that the above setting satisfies the relevant conditions in Theorem 2.3 for convergence.

We have also applied the SARSA algorithm [152] and a population-based version of the Q-learning algorithm to the above test cases. Both SARSA and Q-learning use the same learning rate β_t as in ASA. The underlying learning policy in SARSA is taken to be a Boltzmann distribution over the current admissible set of actions, i.e., $\exp(Q_t(x_t,a)/T_t(x_t))/\sum_{b \in A} \exp(Q_t(x_t,b)/T_t(x_t))$. Note that the crucial differences between ASA and SARSA are that ASA is population-based and searches the entire policy space, while SARSA works with a distribution over the action set of the current sampled state at each time step. For the purpose of a fair comparison, the Q-learning algorithm is implemented in a way to allow it to use the same number of simulation samples per iteration as ASA does. In particular, at each iteration of the algorithm, a population of N_t actions are sampled from a uniform distribution over the current admissible actions, and the entries of the Q-table are then updated according to (5.22). Furthermore, an ε-greedy policy is used at the end of each iteration step of Q-learning to determine the action that leads to the next state, i.e., with probability $1 - \varepsilon_t(x_t)$, selects an action that minimizes the current Q-function estimates; with probability $\varepsilon_t(x_t)$, uniformly generates an action from

the set of admissible actions. The values of $T_t(x_t)$ and $\varepsilon_t(x_t)$ are chosen based on parameter settings discussed in [165].

Figure 5.6 shows the sample convergence behavior of all three comparison algorithms, where the two sub-figures at the top of each case plot the current value function estimates as a function of the number of time step t (i.e., the number of algorithm iterations), and the other two show the value function estimates versus the total number of periods simulated thus far (i.e., computational efforts). It is clear from the figure that the proposed ASA outperforms both SARSA and Q-learning in terms of the number of algorithm iterations. Thus, at an additional computational expense of using an on-line simulation model to assist the decision making process, ASA allows the decision maker to identify the (near) optimal inventory replenishment policy within the shortest time. In addition, note that since both ASA and the version of Q-learning implemented are population-based, they show a more stable behavior than SARSA, as the latter tends to overshoot the optimal values in both test cases. However, this instability behavior of SARSA can be alleviated by using smaller learning rates, which could potentially lead to slower convergence.

5.5 Notes

The upper bound of the result in Theorem 5.1 can be improved if the MDP satisfies an ergodicity condition [40, 84].

The SAMW algorithm is based on the "weighted majority algorithm" of [122], specifically exploiting the work of the "multiplicative weights" algorithm studied by Freund and Schapire [65] in a different context: non-cooperative repeated two-player bimatrix zero-sum games. A result related to Theorem 5.5 is proved in [65] in the context of solving two-player zero-sum bimatrix repeated game, and the proof of Theorem 5.5 is based on the proof there.

The idea of simulating a given (heuristic) policy to obtain an (approximately) improved policy originated from Tesauro's work in backgammon [172], and Bertsekas and Castanon [15] extended the idea to solve finite-horizon MDPs with total reward criterion. Successful applications of the rollout idea include [15] for stochastic scheduling problems; [158] for a vehicle routing problem; [138] and [109] for network routing problems; [157] for pricing of bandwidth provisioning over a single link; [175] for dynamic resource allocation of a geographical positioning system (GPS) server with two traffic classes when the leaky bucket scheme is employed as a traffic policing mechanism; [79] for sensor scheduling for target tracking, using a POMDP model and particle filtering for information-state estimation; [41] for a buffer management problem, where rollout of a fixed threshold policy (Droptail) worked well in numerical experiments; [21] and [113] for various queueing models, where they obtained explicit expressions for the value function of a fixed threshold policy, which plays the role of a heuristic base-policy, and showed numerically that the rollout of the policy behaves almost optimally; see also [112], where the deviation matrix for $M/M/1$ and $M/M/1/N$ queues is derived and used for computing

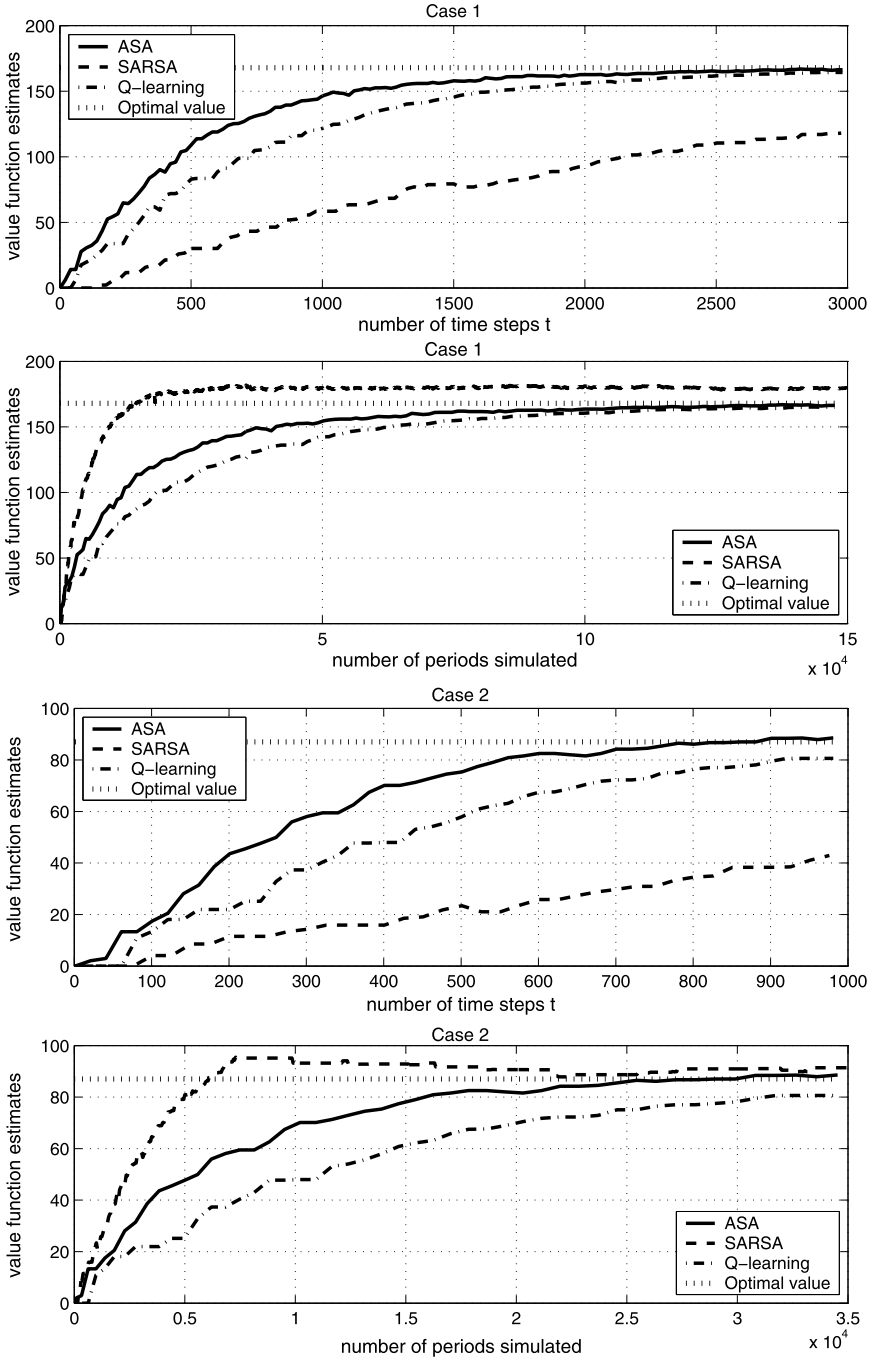


Fig. 5.6 Performance comparison of ASA, SARSA, and Q-learning on an inventory control problem

the bias vector for a particular choice of cost function and a certain base-policy, from which the rollout policy of the base-policy is generated. Further references on rollout applications can be found in [13].

Combining rollout and policy switching for parallel rollout was first proposed in [41], where it was shown that the property of multi-policy improvement in parallel rollout in Theorem 5.12 also holds for finite-horizon MDPs with total reward criterion. Differential training is presented in [11]. Multi-policy improvement can be used for designing "multi-policy iteration" [30] as a variant of PI. Iwamoto [98] established a formal transformation via an "invariant imbedding" to construct a controlled Markov chain that can be solved in a backward manner, as in backward induction for finite-horizon MDPs, for a given controlled Markov chain with a *non-additive* forward recursive objective function. Based on this transformation, Chang [32] extended the methods of parallel rollout and policy switching for forward recursive objective functions and showed that a similar policy-improvement property holds as in MDPs. Chang further studied the multi-policy improvement with a constrained setting in the MDP model of Chap. 1 (see, also [35] for average MDPs) where a policy needs to satisfy some performance constraints in order to be feasible [34] and with a uncertain transition-probability setting within the model called "controlled Markov set-chain" where the transition probabilities in the original MDP model vary in some given domain at each decision time and this variation is unobservable or unknown to the controller [38].

The performance analysis for (parallel) rollout with average reward criterion is presented in [39] in the context of rolling-horizon control under an ergodicity condition. The ordinal comparison analysis of policies, motivated from policy switching, in Markov reward processes with average reward criterion is presented in [31], analyzing the convergence rate of "ϵ-ordinal comparison" of stationary policies under an ergodicity condition. The exponential convergence rate of ordinal comparisons can be established using large deviations theory; see [50] and [67].

Although interchanging the order of expectation and maximization via Jensen's inequality to obtain an upper bound is a well-known technique in many applications (cf. [77, 78, 125]), its use in hindsight optimization for Q_0^*-function value estimates in the framework of the (sampling-based) approximate rolling-horizon control for solving MDPs was first introduced in [47].

Some papers have reported the success of hindsight optimization; e.g., [188] considered a network congestion problem with continuous action space, and [175] studied the performance of hindsight optimization along with parallel rollout for a resource allocation problem.

The scheduling problem example in Sect. 5.3.1 was excerpted from [41].

The presentation of the ASA algorithm in Sect. 5.4 is based on [90]. The structure of the algorithm is similar to that of actor-critic methods, e.g., [9, 19, 111]. However, actor-critic algorithms frequently rely on gradient methods to find improved policies, whereas ASA uses a derivative-free adaptive random search scheme to search for good policies, resulting in global convergence.

References

1. Agrawal, R.: Sample mean based index policies with $O(\log n)$ regret for the multi-armed bandit problem. Adv. Appl. Probab. **27**, 1054–1078 (1995)
2. Altman, E., Koole, G.: On submodular value functions and complex dynamic programming. Stoch. Models **14**, 1051–1072 (1998)
3. Arapostathis, A., Borkar, V.S., Fernández-Gaucherand, E., Ghosh, M.K., Marcus, S.I.: Discrete-time controlled Markov processes with average cost criterion: a survey. SIAM J. Control Optim. **31**(2), 282–344 (1993)
4. Auer, P., Cesa-Bianchi, N., Fisher, P.: Finite-time analysis of the multiarmed bandit problem. Mach. Learn. **47**, 235–256 (2002)
5. Baglietto, M., Parisini, T., Zoppoli, R.: Neural approximators and team theory for dynamic routing: a receding horizon approach. In: Proceedings of the 38th IEEE Conference on Decision and Control, pp. 3283–3288 (1999)
6. Balakrishnan, V., Tits, A.L.: Numerical optimization-based design. In: Levine, W.S. (ed.) The Control Handbook, pp. 749–758. CRC Press, Boca Raton (1996)
7. Banks, J. (ed.): Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice. Wiley, New York (1998)
8. Barash, D.: A genetic search in policy space for solving Markov decision processes. In: AAAI Spring Symposium on Search Techniques for Problem Solving Under Uncertainty and Incomplete Information. Stanford University, Stanford (1999)
9. Barto, A., Sutton, R., Anderson, C.: Neuron-like elements that can solve difficult learning control problems. IEEE Trans. Syst. Man Cybern. **13**, 835–846 (1983)
10. Benaim, M.: A dynamical system approach to stochastic approximations. SIAM J. Control Optim. **34**, 437–472 (1996)
11. Bertsekas, D.P.: Differential training of rollout policies. In: Proceedings of the 35th Allerton Conference on Communication, Control, and Computing (1997)
12. Bertsekas, D.P.: Dynamic Programming and Optimal Control, vol. 1. Athena Scientific, Belmont (2005), vol. 2 (2012)
13. Bertsekas, D.P.: Dynamic programming and suboptimal control: a survey from ASP to MPC. Eur. J. Control **11**, 310–334 (2005)
14. Bertsekas, D.P., Castanon, D.A.: Adaptive aggregation methods for infinite horizon dynamic programming. IEEE Trans. Autom. Control **34**(6), 589–598 (1989)
15. Bertsekas, D.P., Castanon, D.A.: Rollout algorithms for stochastic scheduling problems. J. Heuristics **5**, 89–108 (1999)
16. Bertsekas, D.P., Shreve, S.E.: Stochastic Control: The Discrete Time Case. Academic Press, New York (1978)
17. Bertsekas, D.P., Tsitsiklis, J.N.: Neuro-Dynamic Programming. Athena Scientific, Belmont (1996)

H.S. Chang et al., *Simulation-Based Algorithms for Markov Decision Processes*, Communications and Control Engineering,
DOI 10.1007/978-1-4471-5022-0, © Springer-Verlag London 2013

18. Bes, C., Lasserre, J.B.: An on-line procedure in discounted infinite-horizon stochastic optimal control. J. Optim. Theory Appl. **50**, 61–67 (1986)
19. Bhatnagar, S., Kumar, S.: A simultaneous perturbation stochastic approximation-based actor-critic algorithm for Markov decision processes. IEEE Trans. Autom. Control **49**, 592–598 (2004)
20. Bhatnagar, S., Fu, M.C., Marcus, S.I.: An optimal structured feedback policy for ABR flow control using two timescale SPSA. IEEE/ACM Trans. Netw. **9**, 479–491 (2001)
21. Bhulai, S., Koole, G.: On the structure of value functions for threshold policies in queueing models. Technical Report 2001-4, Department of Stochastics, Vrije Universiteit, Amsterdam (2001)
22. Blondel, V.D., Tsitsiklis, J.N.: A survey of computational complexity results in systems and control. Automatica **36**, 1249–1274 (2000)
23. Borkar, V.S.: White-noise representations in stochastic realization theory. SIAM J. Control Optim. **31**, 1093–1102 (1993)
24. Borkar, V.S.: Convex analytic methods in Markov decision processes. In: Feinberg, E.A., Shwartz, A. (eds.) Handbook of Markov Decision Processes: Methods and Applications. Kluwer, Boston (2002)
25. Bratley, P., Fox, B.L., Schrage, L.E.: A Guide to Simulation. Springer, New York (1983)
26. Browne, C., Powley, E., Whitehouse, D., Lucas, S., Cowling, P.I., Rohlfshagen, P., Tavener, S., Perez, D., Samothrakis, S., Colton, S.: A survey of Monte Carlo tree search methods. IEEE Trans. Comput. Intell. AI Games **4**(1), 1–49 (2012)
27. Burnetas, A.N., Katehakis, M.N.: Optimal adaptive policies for sequential allocation problems. Adv. Appl. Math. **17**(2), 122–142 (1996)
28. Campos-Nanez, E., Garcia, A., Li, C.: A game-theoretic approach to efficient power management in sensor networks. Oper. Res. **56**(3), 552–561 (2008)
29. Chand, S., Hsu, V.N., Sethi, S.: Forecast, solution, and rolling horizons in operations management problems: a classified bibliography. Manuf. Serv. Oper. Manag. **4**(1), 25–43 (2003)
30. Chang, H.S.: Multi-policy iteration with a distributed voting. Math. Methods Oper. Res. **60**(2), 299–310 (2004)
31. Chang, H.S.: On ordinal comparison of policies in Markov reward processes. J. Optim. Theory Appl. **122**(1), 207–217 (2004)
32. Chang, H.S.: Multi-policy improvement in stochastic optimization with forward recursive function criteria. J. Math. Anal. Appl. **305**(1), 130–139 (2005)
33. Chang, H.S.: Converging marriage in honey-bees optimization and application to stochastic dynamic programming. J. Glob. Optim. **35**(3), 423–441 (2006)
34. Chang, H.S.: A policy improvement method in constrained stochastic dynamic programming. IEEE Trans. Autom. Control **51**(9), 1523–1526 (2006)
35. Chang, H.S.: A policy improvement method for constrained average Markov decision processes. Oper. Res. Lett. **35**(4), 434–438 (2007)
36. Chang, H.S.: Finite step approximation error bounds for solving average reward controlled Markov set-chains. IEEE Trans. Autom. Control **53**(1), 350–355 (2008)
37. Chang, H.S.: Decentralized learning in finite Markov chains: revisited. IEEE Trans. Autom. Control **54**(7), 1648–1653 (2009)
38. Chang, H.S., Chong, E.K.P.: Solving controlled Markov set-chains with discounting via multi-policy improvement. IEEE Trans. Autom. Control **52**(3), 564–569 (2007)
39. Chang, H.S., Marcus, S.I.: Approximate receding horizon approach for Markov decision processes: average reward case. J. Math. Anal. Appl. **286**(2), 636–651 (2003)
40. Chang, H.S., Marcus, S.I.: Two-person zero-sum Markov games: receding horizon approach. IEEE Trans. Autom. Control **48**(11), 1951–1961 (2003)
41. Chang, H.S., Givan, R., Chong, E.K.P.: Parallel rollout for on-line solution of partially observable Markov decision processes. Discrete Event Dyn. Syst. Theory Appl. **15**(3), 309–341 (2004)
42. Chang, H.S., Fu, M.C., Hu, J., Marcus, S.I.: An adaptive sampling algorithm for solving Markov decision processes. Oper. Res. **53**(1), 126–139 (2005)

43. Chang, H.S., Lee, H.-G., Fu, M.C., Marcus, S.I.: Evolutionary policy iteration for solving Markov decision processes. IEEE Trans. Autom. Control **50**(11), 1804–1808 (2005)
44. Chang, H.S., Fu, M.C., Hu, J., Marcus, S.I.: An asymptotically efficient simulation-based algorithm for finite horizon stochastic dynamic programming. IEEE Trans. Autom. Control **52**(1), 89–94 (2007)
45. Chang, H.S., Fu, M.C., Hu, J., Marcus, S.I.: Recursive learning automata approach to Markov decision processes. IEEE Trans. Autom. Control **52**(7), 1349–1355 (2007)
46. Chin, H., Jafari, A.: Genetic algorithm methods for solving the best stationary policy of finite Markov decision processes. In: Proceedings of the 30th Southeastern Symposium on System Theory, pp. 538–543 (1998)
47. Chong, E.K.P., Givan, R., Chang, H.S.: A framework for simulation-based network control via hindsight optimization. In: Proceedings of the 39th IEEE Conference on Decision and Control, pp. 1433–1438 (2000)
48. Cooper, W.L., Henderson, S.G., Lewis, M.E.: Convergence of simulation-based policy iteration. Probab. Eng. Inf. Sci. **17**(2), 213–234 (2003)
49. Corana, A., Marchesi, M., Martini, C., Ridella, S.: Minimizing multimodal functions of continuous variables with the 'simulated annealing' algorithm. ACM Trans. Math. Softw. **13**(3), 262–280 (1987)
50. Dai, L.: Convergence properties of ordinal comparison in the simulation of discrete event dynamic systems. J. Optim. Theory Appl. **91**, 363–388 (1996)
51. De Boer, P.T., Kroese, D.P., Mannor, S., Rubinstein, R.Y.: A tutorial on the cross-entropy method. Ann. Oper. Res. **134**, 19–67 (2005)
52. de Farias, D.P., Van Roy, B.: The linear programming approach to approximate dynamic programming. Oper. Res. **51**(6), 850–865 (2003)
53. De Jong, K.A.: An analysis of the behavior of a class of genetic adaptive systems. PhD thesis, University of Michigan, Ann Arbor, MI (1975)
54. Devroye, L.: Non-uniform Random Variate Generation. Springer, New York (1986)
55. Dorigo, M., Gambardella, L.M.: Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Trans. Evol. Comput. **1**, 53–66 (1997)
56. Evans, S.N., Weber, N.C.: On the almost sure convergence of a general stochastic approximation procedure. Bull. Aust. Math. Soc. **34**, 335–342 (1986)
57. Even-Dar, E., Mannor, S., Mansour, Y.: PAC bounds for multi-armed bandit and Markov decision processes. In: Proceedings of the 15th Annual Conference on Computational Learning Theory, pp. 255–270 (2002)
58. Fabian, V.: On asymptotic normality in stochastic approximation. Ann. Math. Stat. **39**, 1327–1332 (1968)
59. Fang, H., Cao, X.: Potential-based on-line policy iteration algorithms for Markov decision processes. IEEE Trans. Autom. Control **49**, 493–505 (2004)
60. Federgruen, A., Tzur, M.: Detection of minimal forecast horizons in dynamic programs with multiple indicators of the future. Nav. Res. Logist. **43**, 169–189 (1996)
61. Feinberg, E.A., Shwartz, A. (eds.): Handbook of Markov Decision Processes: Methods and Applications. Kluwer, Boston (2002)
62. Fernández-Gaucherand, E., Arapostathis, A., Marcus, S.I.: On the average cost optimality equation and the structure of optimal policies for partially observable Markov processes. Ann. Oper. Res. **29**, 471–512 (1991)
63. Fishman, G.S.: Monte Carlo Methods: Concepts, Algorithms, and Applications. Springer, New York (1996)
64. Fishman, G.S.: A First Course in Monte Carlo. Duxbury/Thomson Brooks/Cole, Belmont (2006)
65. Freund, Y., Schapire, R.: Adaptive game playing using multiplicative weights. Games Econ. Behav. **29**, 79–103 (1999)
66. Fu, M.C., Healy, K.J.: Techniques for simulation optimization: an experimental study on an (s, S) inventory system. IIE Trans. **29**, 191–199 (1997)

67. Fu, M.C., Jin, X.: On the convergence rate of ordinal comparisons of random variables. IEEE Trans. Autom. Control **46**, 1950–1954 (2001)
68. Fu, M.C., Marcus, S.I., Wang, I.-J.: Monotone optimal policies for a transient queueing staffing problem. Oper. Res. **46**, 327–331 (2000)
69. Fu, M.C., Hu, J., Marcus, S.I.: Model-based randomized methods for global optimization. In: Proceedings of the 17th International Symposium on Mathematical Theory of Networks and Systems, Kyoto, Japan, July (2006)
70. Garcia, A., Patek, S.D., Sinha, K.: A decentralized approach to discrete optimization via simulation: application to network flow. Oper. Res. **55**(4), 717–732 (2007)
71. Givan, R., Leach, S., Dean, T.: Bounded Markov decision processes. Artif. Intell. **122**, 71–109 (2000)
72. Givan, R., Chong, E.K.P., Chang, H.S.: Scheduling multiclass packet streams to minimize weighted loss. Queueing Syst. **41**(3), 241–270 (2002)
73. Glover, F.: Tabu search: a tutorial. Interfaces **20**(4), 74–94 (1990)
74. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Boston (1989)
75. Gosavi, A.: Simulation-Based Optimization: Parametric Optimization Techniques and Reinforcement Learning. Kluwer, Dordrecht (2003)
76. Grinold, R.C.: Finite horizon approximations of infinite horizon linear programs. Math. Program. **12**, 1–17 (1997)
77. Hartley, R.: Inequalities for a class of sequential stochastic decision processes. In: Dempster, M.A.H. (ed.) Stochastic Programming, pp. 109–123. Academic Press, San Diego (1980)
78. Hausch, D.B., Ziemba, W.T.: Bounds on the value of information in uncertain decision problems. Stochastics **10**, 181–217 (1983)
79. He, Y., Chong, E.K.P.: Sensor scheduling for target tracking: a Monte Carlo sampling approach. Digit. Signal Process. **16**(5), 533–545 (2006)
80. He, Y., Fu, M.C., Marcus, S.I.: Simulation-based algorithms for average cost Markov decision processes. In: Laguna, M., González Velarde, J.L. (eds.) Computing Tools for Modeling, Optimization and Simulation, Interfaces in Computer Science and Operations Research, pp. 161–182. Kluwer, Dordrecht (2000)
81. Henderson, S.G., Nelson, B.L. (eds.): Handbooks in Operations Research and Management Science: Simulation. North-Holland/Elsevier, Amsterdam (2006)
82. Hernández-Lerma, O.: Adaptive Markov Control Processes. Springer, New York (1989)
83. Hernández-Lerma, O., Lasserre, J.B.: A forecast horizon and a stopping rule for general Markov decision processes. J. Math. Anal. Appl. **132**, 388–400 (1988)
84. Hernández-Lerma, O., Lasserre, J.B.: Error bounds for rolling horizon policies in discrete-time Markov control processes. IEEE Trans. Autom. Control **35**, 1118–1124 (1990)
85. Hernández-Lerma, O., Lasserre, J.B.: Discrete-Time Markov Control Processes: Basic Optimality Criteria. Springer, New York (1996)
86. Hoeffding, W.: Probability inequalities for sums of bounded random variables. J. Am. Stat. Assoc. **58**, 13–30 (1963)
87. Homem-de-Mello, T.: A study on the cross-entropy method for rare-event probability estimation. INFORMS J. Comput. **19**(3), 381–394 (2007)
88. Hong, L.J., Nelson, B.L.: Discrete optimization via simulation using COMPASS. Oper. Res. **54**, 115–129 (2006)
89. Hu, J., Chang, H.S.: An approximate stochastic annealing algorithm for finite horizon Markov decision processes. In: Proceedings of the 49th IEEE Conference on Decision and Control, pp. 5338–5343 (2010)
90. Hu, J., Chang, H.S.: Approximate stochastic annealing for online control of infinite horizon Markov decision processes. Automatica **48**, 2182–2188 (2012)
91. Hu, J., Hu, P.: On the performance of the cross-entropy method. In: Proceedings of the 2009 Winter Simulation Conference, pp. 459–468 (2009)

92. Hu, J., Hu, P.: An approximate annealing search algorithm to global optimization and its connections to stochastic approximation. In: Proceedings of the 2010 Winter Simulation Conference, pp. 1223–1234 (2010)
93. Hu, J., Hu, P.: Annealing adaptive search, cross-entropy, and stochastic approximation in global optimization. Nav. Res. Logist. **58**, 457–477 (2011)
94. Hu, J., Fu, M.C., Marcus, S.I.: A model reference adaptive search method for global optimization. Oper. Res. **55**, 549–568 (2007)
95. Hu, J., Fu, M.C., Ramezani, V., Marcus, S.I.: An evolutionary random policy search algorithm for solving Markov decision processes. INFORMS J. Comput. **19**, 161–174 (2007)
96. Hu, J., Fu, M.C., Marcus, S.I.: A model reference adaptive search method for stochastic global optimization. Commun. Inf. Syst. **8**, 245–276 (2008)
97. Hu, J., Hu, P., Chang, H.S.: A stochastic approximation framework for a class of randomized optimization algorithms. IEEE Trans. Autom. Control **57**, 165–178 (2012)
98. Iwamoto, S.: Stochastic optimization of forward recursive functions. J. Math. Anal. Appl. **292**, 73–83 (2004)
99. Jain, R., Varaiya, P.: Simulation-based uniform value function estimates of Markov decision processes. SIAM J. Control Optim. **45**(5), 1633–1656 (2006)
100. Johansen, L.: Lectures on Macroeconomic Planning. North-Holland, Amsterdam (1977)
101. Kaelbling, L., Littman, M., Moore, A.: Reinforcement learning: a survey. Artif. Intell. **4**, 237–285 (1996)
102. Kallenberg, L.: Finite state and action MDPs. In: Feinberg, E.A., Shwartz, A. (eds.) Handbook of Markov Decision Processes: Methods and Applications. Kluwer, Boston (2002)
103. Kalyanasundaram, S., Chong, E.K.P., Shroff, N.B.: Markov decision processes with uncertain transition rates: sensitivity and max-min control. Asian J. Control **6**(2), 253–269 (2004)
104. Kearns, M., Mansour, Y., Ng, A.Y.: A sparse sampling algorithm for near-optimal planning in large Markov decision processes. Mach. Learn. **49**, 193–208 (2001)
105. Keerthi, S.S., Gilbert, E.G.: Optimal infinite horizon feedback laws for a general class of constrained discrete time systems: stability and moving-horizon approximations. J. Optim. Theory Appl. **57**, 265–293 (1988)
106. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science **220**, 45–54 (1983)
107. Kitaev, M.Y., Rykov, V.V.: Controlled Queueing Systems. CRC Press, Boca Raton (1995)
108. Kocsis, L., Szepesvári, C.: Bandit based Monte-Carlo planning. In: Proceedings of the 17th European Conference on Machine Learning, pp. 282–293. Springer, Berlin (2006)
109. Kolarov, A., Hui, J.: On computing Markov decision theory-based cost for routing in circuit-switched broadband networks. J. Netw. Syst. Manag. **3**(4), 405–425 (1995)
110. Koller, D., Parr, R.: Policy iteration for factored MDPs. In: Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence, pp. 326–334 (2000)
111. Konda, V.R., Tsitsiklis, J.N.: Actor-critic algorithms. SIAM J. Control Optim. **42**(4), 1143–1166 (2003)
112. Koole, G.: The deviation matrix of the $M/M/1/\infty$ and $M/M/1/N$ queue, with applications to controlled queueing models. In: Proceedings of the 37th IEEE Conference on Decision and Control, pp. 56–59 (1998)
113. Koole, G., Nain, P.: On the value function of a priority queue with an application to a controlled polling model. Queueing Syst. Theory Appl. **34**, 199–214 (2000)
114. Kumar, P.R., Varaiya, P.: Stochastic Systems: Estimation, Identification, and Adaptive Control. Prentice-Hall, Englewood Cliffs (1986)
115. Kurano, M., Song, J., Hosaka, M., Huang, Y.: Controlled Markov set-chains with discounting. J. Appl. Probab. **35**, 293–302 (1998)
116. Kushner, H.J., Clark, D.S.: Stochastic Approximation Methods for Constrained and Unconstrained Systems. Springer, New York (1978)
117. Kushner, H.J., Yin, G.G.: Stochastic Approximation Algorithms and Applications. Springer, New York (1997)

118. Laarhoven, P.J.M., Aarts, E.H.L.: Simulated Annealing: Theory and Applications. Kluwer Academic, Norwell (1987)
119. Lai, T., Robbins, H.: Asymptotically efficient adaptive allocation rules. Adv. Appl. Math. **6**, 4–22 (1985)
120. Law, A.M., Kelton, W.D.: Simulation Modeling and Analysis, 3rd edn. McGraw-Hill, New York (2000)
121. Lin, A.Z.-Z., Bean, J., White, C. III: A hybrid genetic/optimization algorithm for finite horizon partially observed Markov decision processes. INFORMS J. Comput. **16**(1), 27–38 (2004)
122. Littlestone, N., Warmnuth, M.K.: The weighted majority algorithm. Inf. Comput. **108**, 212–261 (1994)
123. Littman, M., Dean, T., Kaelbling, L.: On the complexity of solving Markov decision problems. In: Proceedings of the 11th Annual Conference on Uncertainty in Artificial Intelligence, pp. 394–402 (1995)
124. MacQueen, J.: A modified dynamic programming method for Markovian decision problems. J. Math. Anal. Appl. **14**, 38–43 (1966)
125. Madansky, A.: Inequalities for stochastic linear programming problems. Manag. Sci. **6**, 197–204 (1960)
126. Mannor, S., Rubinstein, R.Y., Gat, Y.: The cross-entropy method for fast policy search. In: International Conference on Machine Learning, pp. 512–519 (2003)
127. Marbach, P., Tsitsiklis, J.N.: Simulation-based optimization of Markov reward processes. IEEE Trans. Autom. Control **46**(2), 191–209 (2001)
128. Marbach, P., Tsitsiklis, J.N.: Approximate gradient methods in policy-space optimization of Markov reward processes. In: Discrete Event Dynamic Systems: Theory and Applications, vol. 13, pp. 111–148 (2003)
129. Mayne, D.Q., Michalska, H.: Receding horizon control of nonlinear system. IEEE Trans. Autom. Control **38**, 814–824 (1990)
130. Morari, M., Lee, J.H.: Model predictive control: past, present, and future. Comput. Chem. Eng. **23**, 667–682 (1999)
131. Morris, C.N.: Natural exponential families with quadratic variance functions. Ann. Stat. **10**, 65–80 (1982)
132. Mühlenbein, H., Paaß, G.: From recombination of genes to the estimation of distributions, I: binary parameters. In: Voigt, H., Ebeling, W., Rechenberg, I., Schwefel, H. (eds.) Proceedings of the 4th International Conference on Parallel Problem Solving from Nature, pp. 178–187. Springer, Berlin (1996)
133. Narendra, K.S., Thathachar, A.L.: Learning Automata: An Introduction. Prentice-Hall, Englewood Cliffs (1989)
134. Ng, A.Y., Parr, R., Koller, D.: Policy search via density estimation. In: Solla, S.A., Leen, T.K., Müller, K.-R. (eds.) Advances in Neural Information Processing Systems, vol. 12, NIPS 1999, pp. 1022–1028. MIT Press, Cambridge (2000)
135. Niederreiter, H.: Random Number Generation and Quasi-Monte Carlo Methods. SIAM, Philadelphia (1992)
136. Nilim, A., Ghaoui, L.E.: Robust control of Markov decision processes with uncertain transition matrices. Oper. Res. **53**(5), 780–798 (2005)
137. Oommen, B.J., Lanctot, J.K.: Discrete pursuit learning automata. IEEE Trans. Syst. Man Cybern. **20**, 931–938 (1990)
138. Ott, T.J., Krishnan, K.R.: Separable routing: a scheme for state-dependent routing of circuit switched telephone traffic. Ann. Oper. Res. **35**, 43–68 (1992)
139. Patten, W.N., White, L.W.: A sliding horizon feedback control problem with feedforward and disturbance. J. Math. Syst. Estim. Control **7**, 1–33 (1997)
140. Peha, J.M., Tobagi, F.A.: Evaluating scheduling algorithms for traffic with heterogeneous performance objectives. In: Proceedings of the IEEE GLOBECOM, pp. 21–27 (1990)
141. Porteus, E.L.: Conditions for characterizing the structure of optimal strategies in infinite-horizon dynamic programs. J. Optim. Theory Appl. **36**, 419–432 (1982)

142. Powell, W.B.: Approximate Dynamic Programming: Solving the Curses of Dimensionality, 2nd edn. Wiley, New York (2010)
143. Poznyak, A.S., Najim, K.: Learning Automata and Stochastic Optimization. Springer, New York (1997)
144. Poznyak, A.S., Najim, K., Gomez-Ramirez, E.: Self-Learning Control of Finite Markov Chains. Marcel Dekker, New York (2000)
145. Puterman, M.L.: Markov Decision Processes: Discrete Stochastic Dynamic Programming. Wiley, New York (1994)
146. Rajaraman, K., Sastry, P.S.: Finite time analysis of the pursuit algorithm for learning automata. IEEE Trans. Syst. Man Cybern., Part B, Cybern. **26**(4), 590–598 (1996)
147. Romeijn, H.E., Smith, R.L.: Simulated annealing and adaptive search in global optimization. Probab. Eng. Inf. Sci. **8**, 571–590 (1994)
148. Ross, S.M.: Applied Probability Models with Optimization Applications. Dover, Mineola (1992); originally published by Holden-Day, San Francisco (1970)
149. Ross, S.M.: Stochastic Processes, 2nd edn. Wiley, New York (1996)
150. Rubinstein, R.Y., Kroese, D.P.: The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte Carlo Simulation, and Machine Learning. Springer, New York (2004)
151. Rubinstein, R.Y., Shapiro, A.: Discrete Event Systems: Sensitivity Analysis and Stochastic Optimization by the Score Function Method. Wiley, New York (1993)
152. Rummery, G.A., Niranjan, M.: On-line Q-learning using connectionist systems. Technical Report CUED/F-INFENG/TR 166, Engineering Department, Cambridge University (1994)
153. Rust, J.: Structural estimation of Markov decision processes. In: Engle, R., McFadden, D. (eds.) Handbook of Econometrics. North-Holland/Elsevier, Amsterdam (1994)
154. Rust, J.: Using randomization to break the curse of dimensionality. Econometrica **65**(3), 487–516 (1997)
155. Santharam, G., Sastry, P.S., Thathachar, M.A.L.: Continuous action set learning automata for stochastic optimization. J. Franklin Inst. **331B**(5), 607–628 (1994)
156. Satia, J.K., Lave, R.E.: Markovian decision processes with uncertain transition probabilities. Oper. Res. **21**, 728–740 (1973)
157. Savagaonkar, U., Chong, E.K.P., Givan, R.L.: Online pricing for bandwidth provisioning in multi-class networks. Comput. Netw.b **44**(6), 835–853 (2004)
158. Secomandi, N.: Comparing neuro-dynamic programming algorithms for the vehicle routing problem with stochastic demands. Comput. Oper. Res. **27**, 1201–1225 (2000)
159. Sennott, L.I.: Stochastic Dynamic Programming and the Control of Queueing Systems. Wiley, New York (1999)
160. Shanthikumar, J.G., Yao, D.D.: Stochastic monotonicity in general queueing networks. J. Appl. Probab. **26**, 413–417 (1989)
161. Shi, L., Ólafsson, S.: Nested partitions method for global optimization. Oper. Res. **48**, 390–407 (2000)
162. Shi, L., Ólafsson, S.: Nested partitions method for stochastic optimization. Methodol. Comput. Appl. Probab. **2**, 271–291 (2000)
163. Shiryaev, A.N.: Probability, 2nd edn. Springer, New York (1995)
164. Si, J., Barto, A.G., Powell, W.B., Wunsch, D.W. (eds.): Handbook of Learning and Approximate Dynamic Programming. IEEE Press, Piscataway (2004)
165. Singh, S., Jaakkola, T., Littman, M.L., Szepesvári, C.: Convergence results for single-step on-policy reinforcement-learning algorithms. Mach. Learn. **39**, 287–308 (2000)
166. Smith, J.E., McCardle, K.F.: Structural properties of stochastic dynamic programs. Oper. Res. **50**, 796–809 (2002)
167. Spall, J.C.: Introduction to Stochastic Search and Optimization. Wiley, New York (2003)
168. Spall, J.C., Cristion, J.A.: Model-free control of nonlinear stochastic systems with discrete-time measurements. IEEE Trans. Autom. Control **43**, 1198–1210 (1998)
169. Srinivas, M., Patnaik, L.M.: Genetic algorithms: a survey. IEEE Comput. **27**(6), 17–26 (1994)

170. Stidham, S., Weber, R.: A survey of Markov decision models for control of networks of queues. Queueing Syst. **13**, 291–314 (1993)
171. Sutton, R., Barto, A.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (1998)
172. Tesauro, G., Galperin, G.R.: On-line policy improvement using Monte-Carlo search. In: Mozer, M., Jordan, M.I., Petsche, T. (eds.) Advances in Neural Information Processing Systems, vol. 9, NIPS 1996, pp. 1068–1074. MIT Press, Cambridge (1997)
173. Thathachar, M.A.L., Sastry, P.S.: A class of rapidly converging algorithms for learning automata. IEEE Trans. Syst. Man Cybern. **SMC-15**, 168–175 (1985)
174. Thathachar, M.A.L., Sastry, P.S.: Varieties of learning automata: an overview. IEEE Trans. Syst. Man Cybern., Part B, Cybern. **32**(6), 711–722 (2002)
175. Tinnakornsrisuphap, P., Vanichpun, S., La, R.: Dynamic resource allocation of GPS queues under leaky buckets. In: Proceedings of IEEE GLOBECOM, pp. 3777–3781 (2003)
176. Topsoe, F.: Bounds for entropy and divergence for distributions over a two-element set. J. Inequal. Pure Appl. Math. **2**(2), 25 (2001)
177. Tsitsiklis, J.N.: Asynchronous stochastic approximation and Q-learning. Mach. Learn. **16**, 185–202 (1994)
178. van den Broek, W.A.: Moving horizon control in dynamic games. J. Econ. Dyn. Control **26**, 937–961 (2002)
179. Van Roy, B.: Neuro-dynamic programming: overview and recent trends. In: Feinberg, E.A., Shwartz, A. (eds.) Handbook of Markov Decision Processes: Methods and Applications. Kluwer, Boston (2002)
180. Watkins, C.J.C.H.: Q-learning Mach. Learn. **8**, 279–292 (1992)
181. Weber, R.: On the Gittins index for multiarmed bandits. Ann. Appl. Probab. **2**, 1024–1033 (1992)
182. Wells, C., Lusena, C., Goldsmith, J.: Genetic algorithms for approximating solutions to POMDPs. Technical Report TR-290-99, Department of Computer Science, University of Kentucky (1999)
183. Wheeler, R.M., Jr., Narendra, K.S.: Decentralized learning in finite Markov chains. IEEE Trans. Autom. Control **31**(6), 519–526 (1986)
184. White, C.C., Eldeib, H.K.: Markov decision processes with imprecise transition probabilities. Oper. Res. **43**, 739–749 (1994)
185. Williams, R.J.: Simple statistical gradient-following algorithms for connectionist reinforcement learning. Mach. Learn. **8**, 229–256 (1992)
186. Williams, J.L., Fisher, J.W. III, Willsky, A.S.: Importance sampling actor-critic algorithms. In: Proceedings of the 2006 American Control Conference, pp. 1625–1630 (2006)
187. Wolpert, D.H.: Finding bounded rational equilibria, part I: iterative focusing. In: Vincent, T. (ed.) Proceedings of the Eleventh International Symposium on Dynamic Games and Applications, ISDG '04 (2004)
188. Wu, G., Chong, E.K.P., Givan, R.L.: Burst-level congestion control using hindsight optimization. IEEE Trans. Autom. Control **47**(6), 979–991 (2002)
189. Yakowitz, S., L'Ecuyer, P., Vázquez-Abad, F.: Global stochastic optimization with low-dispersion point sets. Oper. Res. **48**, 939–950 (2000)
190. Zabinsky, Z.B.: Stochastic Adaptive Search for Global Optimization. Kluwer Academic, Norwell (2003)
191. Zhang, Q., Mühlenbein, H.: On the convergence of a class of estimation of distribution algorithm. IEEE Trans. Evol. Comput. **8**(2), 127–136 (2004)
192. Zlochin, M., Birattari, M., Meuleau, N., Dorigo, M.: Model-based search for combinatorial optimization: a critical survey. Ann. Oper. Res. **131**, 373–395 (2004)

Index

H.S. Chang et al., *Simulation-Based Algorithms for Markov Decision Processes*, Communications and Control Engineering,
DOI 10.1007/978-1-4471-5022-0, © Springer-Verlag London 2013