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
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
Stony Brook, NY, USA
College Park, MD, USA

Hyeong Soo Chang
Jiaqiao Hu
Michael Fu
Steve Marcus
Contents

1 Markov Decision Processes ........................................... 1
  1.1 Optimality Equations ........................................... 3
  1.2 Policy Iteration and Value Iteration ......................... 5
  1.3 Rolling-Horizon Control .................................... 7
  1.4 Survey of Previous Work on Computational Methods .......... 8
  1.5 Simulation .................................................. 10
  1.6 Preview of Coming Attractions ............................. 13
  1.7 Notes ...................................................... 14

2 Multi-stage Adaptive Sampling Algorithms ...................... 19
  2.1 Upper Confidence Bound Sampling .......................... 21
    2.1.1 Regret Analysis in Multi-armed Bandits ............... 21
    2.1.2 Algorithm Description ................................ 22
    2.1.3 Alternative Estimators ................................ 25
    2.1.4 Convergence Analysis ................................ 25
    2.1.5 Numerical Example ................................... 33
  2.2 Pursuit Learning Automata Sampling ........................ 37
    2.2.1 Algorithm Description ................................ 42
    2.2.2 Convergence Analysis ................................ 44
    2.2.3 Application to POMDPs ................................ 52
    2.2.4 Numerical Example ................................... 54
  2.3 Notes ...................................................... 57

3 Population-Based Evolutionary Approaches ...................... 61
  3.1 Evolutionary Policy Iteration ............................... 63
    3.1.1 Policy Switching ...................................... 63
    3.1.2 Policy Mutation and Population Generation ............ 65
    3.1.3 Stopping Rule ......................................... 65
    3.1.4 Convergence Analysis ................................ 66
    3.1.5 Parallelization ....................................... 67
  3.2 Evolutionary Random Policy Search .......................... 67
3.2.1 Policy Improvement with Reward Swapping ..................... 68
3.2.2 Exploration .................................. 71
3.2.3 Convergence Analysis ................................ 73
3.3 Numerical Examples .................................. 76
3.3.1 A One-Dimensional Queueing Example ...................... 76
3.3.2 A Two-Dimensional Queueing Example ..................... 83
3.4 Extension to Simulation-Based Setting ......................... 86
3.5 Notes ............................................. 87

4 Model Reference Adaptive Search ................................. 89
4.1 The Model Reference Adaptive Search Method .................... 91
4.1.1 The MRAS0 Algorithm (Idealized Version) .................. 92
4.1.2 The MRAS1 Algorithm (Adaptive Monte Carlo Version) .... 96
4.1.3 The MRAS2 Algorithm (Stochastic Optimization) ........... 98
4.2 Convergence Analysis of MRAS ................................ 101
4.2.1 MRAS0 Convergence ................................ 101
4.2.2 MRAS1 Convergence ................................ 107
4.2.3 MRAS2 Convergence ................................ 117
4.3 Application of MRAS to MDPs via Direct Policy Learning ..... 131
4.3.1 Finite-Horizon MDPs ................................ 131
4.3.2 Infinite-Horizon MDPs ................................ 132
4.3.3 MDPs with Large State Spaces ............................ 132
4.3.4 Numerical Examples .................................. 135
4.4 Application of MRAS to Infinite-Horizon MDPs in Population-
   Based Evolutionary Approaches ............................... 141
4.4.1 Algorithm Description ................................ 142
4.4.2 Numerical Examples .................................. 143
4.5 Application of MRAS to Finite-Horizon MDPs Using Adaptive
   Sampling ............................................ 144
4.6 A Stochastic Approximation Framework ......................... 148
4.6.1 Model-Based Annealing Random Search ..................... 149
4.6.2 Application of MARS to Finite-Horizon MDPs ............... 166
4.7 Notes ............................................. 177

5 On-Line Control Methods via Simulation ............................ 179
5.1 Simulated Annealing Multiplicative Weights Algorithm ....... 183
5.1.1 Basic Algorithm Description .............................. 184
5.1.2 Convergence Analysis ................................ 185
5.1.3 Convergence of the Sampling Version of the Algorithm .... 189
5.1.4 Numerical Example .................................. 191
5.1.5 Simulated Policy Switching .............................. 194
5.2 Rollout .......................................... 195
5.2.1 Parallel Rollout .................................... 197
5.3 Hindsight Optimization ................................... 199
5.3.1 Numerical Example .................................. 200
5.4 Approximate Stochastic Annealing .............................. 204
Selected Notation and Abbreviations

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

\[ ^1 \text{Notation specific to a particular chapter is noted parenthetically. Equation numbers indicate where the quantity is defined.} \]
Selected Notation and Abbreviations

\( V_i^* \)  
- optimal reward-to-go value function from stage \( i \)

\( V_i^{N_i} \)  
- 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

\( V_{i}^{\pi} \)  
- reward-to-go value function for policy \( \pi \) from stage \( i \): (1.6)

\( V_{i}^{\pi} \)  
- value function for policy \( \pi \): (1.11)

\( V_{i}^{\hat{\pi}}(x) \)  
- expected total discounted reward over horizon length \( H \) under policy \( \pi \), 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)

\( Q_{i}^*(x,a) \)  
- infinite-horizon \( Q \)-function value: (1.14)

\( Q_{i}^{N_i}(x,a) \)  
- 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 \( A(x) \)

\( \approx \)  
- almost surely

\( \text{c.d.f.} \)  
- cumulative distribution function

\( \text{i.i.d.} \)  
- independent and identically distributed

\( \text{p.d.f.} \)  
- probability density function

\( \text{p.m.f.} \)  
- probability mass function

\( \text{s.t.} \)  
- such that (or subject to)

\( \text{w.p.} \)  
- with probability

\( \text{w.r.t.} \)  
- 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) \( \mu \) and variance \( \sigma^2 \) (covariance matrix \( \Sigma \))

\( E_f \)  
- expectation under p.d.f. \( f \) (Chap. 4)

\( E_{\theta,P_{\theta}} \)  
- expectation/probability under p.d.f./p.m.f. \( f(\cdot,\theta) \) (Chap. 4)

\( \tilde{E}_{\theta,P_{\theta}} \)  
- 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)

\( \text{NEF} \)  
- natural exponential family (Chap. 4)

\( := \)  
- equal by definition

\( \overset{d}{=} \)  
- equal in distribution

\( \iff \)  
- if and only if

\( \implies \)  
- implies (or weak convergence)

\( I\{\cdot\} \)  
- indicator function of the set \( \{\cdot\} \)
<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>X</td>
</tr>
<tr>
<td>$| \cdot |$</td>
<td>norm of a function or vector, or induced norm of a matrix</td>
</tr>
<tr>
<td>$x \lor y$</td>
<td>$\max(x, y)$</td>
</tr>
<tr>
<td>$x \land y$</td>
<td>$\min(x, y)$</td>
</tr>
<tr>
<td>$x^+$</td>
<td>$\max(x, 0)$</td>
</tr>
<tr>
<td>$x^-$</td>
<td>$\min(-x, 0)$</td>
</tr>
<tr>
<td>$\lfloor x \rfloor$</td>
<td>least integer greater than or equal to $x$</td>
</tr>
<tr>
<td>$\lceil x \rceil$</td>
<td>greatest integer less than or equal to $x$</td>
</tr>
<tr>
<td>$f(n) = O(g(n))$</td>
<td>$\limsup_{n \to \infty} \frac{f(n)}{g(n)} &lt; \infty$</td>
</tr>
<tr>
<td>$f(n) = \Theta(g(n))$</td>
<td>$f(n) = O(g(n))$ and $g(n) = O(f(n))$</td>
</tr>
</tbody>
</table>
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., \(0 \leq R(x,a) \leq R_{\text{max}}\) 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 \((\text{stage or period})\) \(t \in \{0, 1, \ldots\}\) 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 \(\Pi\) be the set of non-stationary Markovian policies \(\pi = \{\pi_t, t = 0, 1, \ldots\}\), where \(\pi_t : X \rightarrow A\) is a function such that \(\pi_t(x) \in A(x)\) for each \(x \in X\). The goal is to find a policy \(\pi\) that maximizes the expected total discounted reward given by

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

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
denoted by $V^*: X \rightarrow \mathbb{N}^+$, where the optimal value for a given state $x \in X$ is given by

$$V^*(x) = \sup_{\pi \in \Pi} V^\pi(x),$$  \hspace{1cm} (1.2)$$

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

$$V^*(x) = V^{\pi^*}(x), \quad x \in X.$$  \hspace{1cm} (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

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

and $R'(x_t, a_t, w_t) \leq R_{\text{max}}$ 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

\[
V^*_i(x) = \sup_{\pi \in \Pi} V^\pi_i(x),
\]

where the reward-to-go value for policy \(\pi\) for state \(x\) in stage \(i\) is defined by

\[
V^\pi_i(x) = E \left[ \sum_{t=i}^{H-1} \gamma^{t-i} R(x_t, \pi_t(x_t)) \left| x_i = x \right. \right],
\]

\(i = 0, \ldots, H - 1\), with \(V^*_H(x) = 0\) for all \(x \in X\). Note that \(V^\pi_i(x) = V^\pi_0(x)\) and \(V^*(x) = V^*_0(x)\), where \(V^\pi\) and \(V^*\) are the value function for \(\pi\) 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 \(i = 0, \ldots, H - 1\),

\[
V^*_i(x) = \sup_{a \in A(x)} \left\{ R(x,a) + \gamma \sum_{y \in X} P(x,a)(y)V^*_i+1(y) \right\},
\]

or, equivalently, by defining the Q-function,

\[
V^*_i(x) = \sup_{a \in A(x)} Q^*_i(x,a),
\]

\[
Q^*_i(x,a) = R(x,a) + \gamma \sum_{y \in X} P(x,a)(y)V^*_i+1(y).
\]

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 \(x_0\):

\[
V^*_0(x_0) = \sup_{\pi \in \Pi} V^\pi_0(x_0).
\]

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

\[
\Pi_s = \left\{ \pi \in \Pi \mid \pi_t = \pi_{t'} \forall t, t' \right\},
\]

since under mild regularity conditions, an optimal policy always exists in \(\Pi_s\) for the infinite-horizon problem. In a slight abuse of notation, we use \(\pi\) 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 \( x \in X, \ 0 < \gamma < 1, \pi \in \Pi_s \),

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

(1.11)

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

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

(1.12)

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

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

(1.13)

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

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

Define

\[
Q^*(x, a) = R(x, a) + \gamma \sum_{y \in X} P(x, a)(y) V^*(y), \ x \in X, \ a \in A(x).
\]

(1.14)

Then it immediately follows that

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

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

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

(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 \( \pi \) for state \( x \) in stage \( i \) over a horizon \( H \) corresponding to (1.6) is given by

\[
V^\pi_i(x) = E \left[ \sum_{t=i}^{H-1} \gamma^{t-i} R'(x_t, \pi_t(x_t), w_t) \bigg| x_i = x \right],
\]

(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 \( \pi \), and \( w_1, \ldots, w_{H-1} \) 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:

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

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:

\[
V^*(x) = \sup_{a \in A(x)} E[R'(x, a)] = \sup_{a \in A(x)} Q^*(x, a),
\]

\[
\pi^*(x) \in \arg \sup_{a \in A(x)} E[R'(x, a) + \gamma V^*(f(x, a))], \quad a \in A(x),
\]

\[
Q^*(x, a) = E[R'(x, a)] + \gamma E[V^*(f(x, a))],
\]

\[
= E[R'(x, a)] + \gamma E \left[ \sup_{a' \in A(f(x, a))} Q^*(f(x, a), a') \right].
\]

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 \( \pi^* \) 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

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

(1.20)

$$
T_\pi(\Phi)(x) = E[R'(x, \pi(x))] + \gamma E[\Phi(f(x, \pi(x)))].
$$

(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 $V_\pi$ for a given $\pi \in \Pi_s$ by solving the corresponding fixed-point functional equation over all $x \in X$:

$$
V_\pi(x) = R(x, \pi(x)) + \gamma \sum_{y \in X} P(x, \pi(x))(y)V_\pi(y),
$$

(1.22)

$$
V_\pi(x) = E[R'(x, \pi(x))] + \gamma E[V_\pi(f(x, \pi(x)))]
$$

(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 $\pi$ and obtains a new policy $\hat{\pi}$ by satisfying the condition $T(\hat{\pi})(x) = T_\hat{\pi}(\hat{\pi})(x)$, i.e., for each $x \in X$, by taking the action

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

(1.24)

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

(1.25)

The policy improvement step ensures that the value function of $\hat{\pi}$ is no worse than that of $\pi$, 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 $x \in X$,

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