Reinforcement Learning and Optimal Control

by

Dimitri P. Bertsekas

Arizona State University
and
Massachusetts Institute of Technology

WWW site for book information and orders

http://www.athenasc.com

Athena Scientific, Belmont, Massachusetts

Athena Scientific
Post Office Box 805
Nashua, NH 03060
U.S.A.

Email: info@athenasc.com
WWW: http://www.athenasc.com

Cover photography: Dimitri Bertsekas

Publisher's Cataloging-in-Publication Data

Bertsekas, Dimitri P.
Reinforcement Learning and Optimal Control
Includes Bibliography and Index
1. Mathematical Optimization. 2. Dynamic Programming. I. Title.
QA402.5 .B465 2019 519.703 00-91281

ISBN-10: 1-886529-39-6, ISBN-13: 978-1-886529-39-7

Ebook version (includes minor revisions)

ABOUT THE AUTHOR

Dimitri Bertsekas studied Mechanical and Electrical Engineering at the National Technical University of Athens, Greece, and obtained his Ph.D. in system science from the Massachusetts Institute of Technology. He has held faculty positions with the Engineering-Economic Systems Department, Stanford University, and the Electrical Engineering Department of the University of Illinois, Urbana. In 1979 he joined the Electrical Engineering and Computer Science Department of the Massachusetts Institute of Technology (M.I.T.), where he retired in 2016 as McAfee Professor of Engineering. In 2019, he became Fulton Professor of Computational Decision Making in the School of Computing, Informatics, and Decision Systems Engineering at the Arizona State University, Tempe, AZ.

Professor Bertsekas' teaching and research have spanned several fields, including deterministic optimization, dynamic programming and stochastic control, large-scale and distributed computation, and data communication networks. He has authored or coauthored numerous research papers and eighteen books, several of which are currently used as textbooks in MIT and ASU classes, including "Dynamic Programming and Optimal Control," "Data Networks," "Introduction to Probability," "Convex Optimization Algorithms," "Nonlinear Programming," and the present book.

Professor Bertsekas was awarded the INFORMS 1997 Prize for Research Excellence in the Interface Between Operations Research and Computer Science, the 2001 AACC John R. Ragazzini Education Award, the 2009 INFORMS Expository Writing Award, the 2014 AACC Richard Bellman Heritage Award, the 2014 INFORMS Khachiyan Prize for Life-Time Accomplishments in Optimization, the 2015 MOS/SIAM George B. Dantzig Prize, and the 2022 IEEE Control Systems Award. In 2018, he was awarded, jointly with his coauthor John Tsitsiklis, the INFORMS John von Neumann Theory Prize, for the contributions of the research monographs "Parallel and Distributed Computation" and "Neuro-Dynamic Programming." In 2001, he was elected to the United States National Academy of Engineering for "pioneering contributions to fundamental research, practice and education of optimization/control theory, and especially its application to data communication networks."

ATHENA SCIENTIFIC

OPTIMIZATION AND COMPUTATION SERIES

1. Rollout, Policy Iteration, and Distributed Reinforcement Learning, by Dimitri P. Bertsekas, 2020, ISBN 978-1-886529-07-6, 480 pages
2. Reinforcement Learning and Optimal Control, by Dimitri P. Bertsekas, 2019, ISBN 978-1-886529-39-7, 388 pages
3. Abstract Dynamic Programming, 2nd Edition, by Dimitri P. Bertsekas, 2018, ISBN 978-1-886529-46-5, 360 pages
4. Dynamic Programming and Optimal Control, Two-Volume Set, by Dimitri P. Bertsekas, 2017, ISBN 1-886529-08-6, 1270 pages
5. Nonlinear Programming, 3rd Edition, by Dimitri P. Bertsekas, 2016, ISBN 1-886529-05-1, 880 pages
6. Convex Optimization Algorithms, by Dimitri P. Bertsekas, 2015, ISBN 978-1-886529-28-1, 576 pages
7. Convex Optimization Theory, by Dimitri P. Bertsekas, 2009, ISBN 978-1-886529-31-1, 256 pages
8. Introduction to Probability, 2nd Edition, by Dimitri P. Bertsekas and John N. Tsitsiklis, 2008, ISBN 978-1-886529-23-6, 544 pages
9. Convex Analysis and Optimization, by Dimitri P. Bertsekas, Angelia Nedić, and Asuman E. Ozdaglar, 2003, ISBN 1-886529-45-0, 560 pages
10. Network Optimization: Continuous and Discrete Models, by Dimitri P. Bertsekas, 1998, ISBN 1-886529-02-7, 608 pages
11. Network Flows and Monotropic Optimization, by R. Tyrrell Rockafellar, 1998, ISBN 1-886529-06-X, 634 pages
12. Introduction to Linear Optimization, by Dimitris Bertsimas and John N. Tsitsiklis, 1997, ISBN 1-886529-19-1, 608 pages
13. Parallel and Distributed Computation: Numerical Methods, by Dimitri P. Bertsekas and John N. Tsitsiklis, 1997, ISBN 1-886529-01-9, 718 pages
14. Neuro-Dynamic Programming, by Dimitri P. Bertsekas and John N. Tsitsiklis, 1996, ISBN 1-886529-10-8, 512 pages
15. Constrained Optimization and Lagrange Multiplier Methods, by Dimitri P. Bertsekas, 1996, ISBN 1-886529-04-3, 410 pages
16. Stochastic Optimal Control: The Discrete-Time Case, by Dimitri P. Bertsekas and Steven E. Shreve, 1996, ISBN 1-886529-03-5, 330 pages

Contents

Preface

Turning to the succor of modern computing machines, let us renounce all analytic tools.

Richard Bellman [Bel57]

From a teleological point of view the particular numerical solution of any particular set of equations is of far less importance than the understanding of the nature of the solution.

Richard Bellman [Bel57]

In this book we consider large and challenging multistage decision problems, which can be solved in principle by dynamic programming (DP for short), but their exact solution is computationally intractable. We discuss solution methods that rely on approximations to produce suboptimal policies with adequate performance. These methods are collectively known by several essentially equivalent names: *reinforcement learning*, *approximate dynamic programming*, and *neuro-dynamic programming*. We will use primarily the most popular name: reinforcement learning.

Our subject has benefited greatly from the interplay of ideas from optimal control and from artificial intelligence. One of the aims of the book is to explore the common boundary between these two fields and to form a bridge that is accessible by workers with background in either field. Another aim is to organize coherently the broad mosaic of methods that have proved successful in practice while having a solid theoretical and/or logical foundation. This may help researchers and practitioners to find their way through the maze of competing ideas that constitute the current state of the art.

There are two general approaches for DP-based suboptimal control. The first is *approximation in value space*, where we approximate in some way the optimal cost-to-go function with some other function. The major alternative to approximation in value space is *approximation in policy*

space, whereby we select the policy by using optimization over a suitably restricted class of policies, usually a parametric family of some form. In some schemes these two types of approximation may be combined, aiming to capitalize on the advantages of both. Generally, approximation in value space is tied more closely to the central DP ideas of value and policy iteration than approximation in policy space, which relies on gradient-like descent, a more broadly applicable optimization mechanism.

While we provide a substantial treatment of approximation in policy space, most of the book is focused on approximation in value space. Here, the control at each state is obtained by optimization of the cost over a limited horizon, plus an approximation of the optimal future cost. The latter cost, which we generally denote by $\tilde{J}$, is a function of the state where we may be. It may be computed by a variety of methods, possibly involving simulation and/or some given or separately derived heuristic/suboptimal policy. The use of simulation often allows for implementations that do not require a mathematical model, a major idea that has allowed the use of DP beyond its classical boundaries.

We discuss selectively four types of methods for obtaining $\tilde{J}$:

(a) *Problem approximation*: Here $\tilde{J}$ is the optimal cost function of a related simpler problem, which is solved by exact DP. Certainty equivalent control and enforced decomposition schemes are discussed in some detail.

(b) *Rollout and model predictive control*: Here $\tilde{J}$ is the cost function of some known heuristic policy. The needed cost values to implement a rollout policy are often calculated by simulation. While this method applies to stochastic problems, the reliance on simulation favors deterministic problems, including challenging combinatorial problems for which heuristics may be readily implemented. Rollout may also be combined with adaptive simulation and Monte Carlo tree search, which have proved very effective in the context of games such as backgammon, chess, Go, and others.

Model predictive control was originally developed for continuous-space optimal control problems that involve some goal state, e.g., the origin in a classical control context. It can be viewed as a specialized rollout method that is based on a suboptimal optimization for reaching a goal state.

(c) *Parametric cost approximation*: Here $\tilde{J}$ is chosen from within a parametric class of functions, including neural networks, with the parameters "optimized" or "trained" by using state-cost sample pairs and some type of incremental least squares/regression algorithm. Approximate policy iteration and its variants are covered in some detail, including several actor and critic schemes. These involve policy evaluation with simulation-based training methods, and policy improve-

ment that may rely on approximation in policy space.

(d) *Aggregation*: Here $\tilde{J}$ is the optimal cost function of some approximation to the original problem, called aggregate problem, which has fewer states. The aggregate problem can be formulated in a variety of ways, and may be solved by using exact DP techniques. Its optimal cost function is then used as $\tilde{J}$ in a limited horizon optimization scheme. Aggregation may also be used to provide local improvements to parametric approximation schemes that involve neural networks or linear feature-based architectures.

We have adopted a gradual expository approach, which proceeds along four directions:

(1) *From exact DP to approximate DP*: We first discuss exact DP algorithms, explain why they may be difficult to implement, and then use them as the basis for approximations.

(2) *From finite horizon to infinite horizon problems*: We first discuss finite horizon exact and approximate DP methodologies, which are intuitive and mathematically simple in Chapters 1-3. We then progress to infinite horizon problems in Chapters 4-6.

(3) *From deterministic to stochastic models*: We often discuss separately deterministic and stochastic problems. The reason is that deterministic problems are simpler and offer special advantages for some of our methods.

(4) *From model-based to model-free implementations*: Reinforcement learning methods offer a major potential benefit over classical DP approaches, which were practiced exclusively up to the early 90s: they can be implemented by using a simulator/computer model rather than a mathematical model. In our presentation, we first discuss model-based implementations, and then we identify schemes that can be appropriately modified to work with a simulator.

After the first chapter, each new class of methods is introduced as a more sophisticated or generalized version of a simpler method introduced earlier. Moreover, we illustrate some of the methods by means of examples, which should be helpful in providing insight into their use, but may also be skipped selectively and without loss of continuity.

The mathematical style of this book is somewhat different from the one of the author's DP books [Ber12], [Ber17], [Ber18a], and the 1996 neuro-dynamic programming (NDP) research monograph, written jointly with John Tsitsiklis [BeT96]. While we provide a rigorous, albeit short, mathematical account of the theory of finite and infinite horizon DP, and some fundamental approximation methods, we rely more on intuitive explanations and less on proof-based insights. Moreover, our mathematical requirements are quite modest: calculus, a minimal use of matrix-vector al-

gebra, and elementary probability (mathematically complicated arguments involving laws of large numbers and stochastic convergence are bypassed in favor of intuitive explanations).

Still in our use of a more intuitive but less proof-oriented expository style, we have followed a few basic principles. The most important of these is to maintain rigor in the use of natural language. The reason is that with fewer mathematical arguments and proofs, precise language is essential to maintain a logically consistent exposition. In particular, we have aimed to define terms unambiguously, and to avoid using multiple terms with essentially identical meaning. Moreover, when circumstances permitted, we have tried to provide enough explanation/intuition so that a mathematician can find the development believable and even construct the missing rigorous proofs.

We note that several of the methods that we present are often successful in practice, but have less than solid performance properties. This is a reflection of the state of the art in the field: there are no methods that are guaranteed to work for all or even most problems, but there are enough methods to try on a given problem with a reasonable chance of success in the end.† To aid in this process, we place primary emphasis on developing intuition into the inner workings of each type of method. Still, however, it is important to have a foundational understanding of the analytical principles of the field and of the mechanisms underlying the central computational methods. To quote a statement from the preface of the NDP monograph [BeT96]: "It is primarily through an understanding of the mathematical structure of the NDP methodology that we will be able to identify promising or solid algorithms from the bewildering array of speculative proposals and claims that can be found in the literature."

Another statement from a recent NY Times article [Str18], in connection with DeepMind's remarkable AlphaZero chess program, is also worth quoting: "What is frustrating about machine learning, however, is that the algorithms can't articulate what they're thinking. We don't know why they work, so we don't know if they can be trusted. AlphaZero gives every appearance of having discovered some important principles about chess, but it can't share that understanding with us. Not yet, at least. As human beings, we want more than answers. We want insight. This is going to be

† While reinforcement learning rests on the mathematical principles of DP, it also relies on multiple interacting approximations whose effects are hard to predict and quantify in practice. It may be hoped that with further theoretical and applications research, the state of the subject will improve and clarify. However, it can be said that in its current form, reinforcement learning is an exploding field, which is complicated, unclean, and somewhat confusing (something that the front cover image of the book also tries to convey). Reinforcement learning is not unique in this. One may think of other important optimization areas where a similar state of the art has prevailed for a long time.

a source of tension in our interactions with computers from now on."† To this we may add that human insight can only develop within some structure of human thought, and it appears that mathematical reasoning with algorithmic models is the most suitable structure for this purpose.

I would like to express my appreciation to the many students and colleagues that contributed directly or indirectly to the book. A special thanks is due to my principal collaborators on the subject, over the last 25 years, particularly John Tsitsiklis, Janey (Huizhen) Yu, and Mengdi Wang. Moreover, sharing insights with Ben Van Roy over the years has been important in shaping my thinking. Interactions with Ben Recht regarding policy gradient methods were also very helpful. The projects that my students worked on as part of DP courses I taught at MIT inspired many ideas that indirectly found their way into the book. I want to express my thanks to the many readers, who proofread parts of the book. In this respect I would like to single out Yuchao Li who made many helpful comments, and Thomas Stahlbuhk, who went through the entire book with great care, and offered numerous insightful suggestions.

The book took shape while teaching a course on the subject at the Arizona State University (ASU) during a two-month period starting in January 2019. Videolectures and slides from this class are available from my website

http://web.mit.edu/dimitrib/www/RLbook.html

and provide a good supplement and companion resource to the book. The hospitable and stimulating environment at ASU contributed much to my productivity during this period, and for this I am very thankful to Stephanie Gil, as well as other colleagues from ASU, including Heni Ben Amor, Esma Gel, Subbarao (Rao) Kambhampati, Angelia Nedic, Giulia Pedrielli, Jennie Si, and Petr Sulc. Moreover, Stephanie together with her

† The two 1957 Bellman quotations at the beginning of this preface also express this tension, although the first of these, while striking and widely cited, is admittedly taken a little out of context (throughout his work on practical applications, Bellman remained a mathematical analyst at heart). Bellman's fascinating autobiography [Bel84] contains a lot of information on the origins of DP (and approximate DP as well!); selected quotations from this autobiography have been compiled by his collaborator Dreyfus [Dre02]. Among others, Bellman states that "In order to make any progress, it is necessary to think of approximate techniques, and above all, of numerical algorithms. Finally, having devoted a great deal of time and effort, mostly fruitless, to the analysis of many varieties of simple models, I was prepared to face up to the challenge of using dynamic programming as an effective tool for obtaining numerical answers to numerical questions." He goes on to attribute his motivation to work on numerical DP to the emergence of the (then primitive) digital computer, which he calls "Sorcerer's Apprentice."

students, Sushmita Bhattacharya and Thomas Wheeler, collaborated with me on research and implementation of various methods, contributed many insights, and tested out several variations.

Dimitri P. Bertsekas

June 2019

NOTE ABOUT THE EBOOK EDITION

This 2021 ebook edition contains minor editorial changes to the 2019 original version, which were prompted by the publication of my companion research monograph

> Rollout, Policy Iteration, and Distributed Reinforcement Learning, Athena Scientific, 2020, ISBN 978-1-886529-07-6

This latter monograph focuses more closely on several topics related to rollout, approximate policy iteration, multiagent problems, discrete and Bayesian optimization, and distributed computation, which are either discussed in less detail or not covered at all in the present book. Moreover, the monograph's development follows somewhat more narrowly the AlphaZero and TD-Gammon paradigms, and their off-line training/on-line play algorithmic structure, which is the key to their success.

On the other hand, the present book provides a more comprehensive coverage of reinforcement learning, and includes the development of topics that are not covered at all in the 2020 book, such as approximation in policy space, aggregation, and temporal difference methods. It also contains, in Chapters 4 and 5, a proof-based development of some of the infinite horizon exact and approximate DP theory.

My website

> http://web.mit.edu/dimitrib/www/RLbook.html

contains class notes, and a series of videolectures and slides from my 2021 course, which address a selection of topics from both books. The videolectures are also available at

> https://www.youtube.com/playlist?list
> =PLmH30BG15SIp79JRJ-MVF12uvB1qPtPzn

and at

> https://space.bilibili.com/2036999141

Dimitri P. Bertsekas

July 2021

1

Exact Dynamic Programming

Contents

In this chapter, we provide some background on exact dynamic programming (DP for short), with a view towards the suboptimal solution methods that are the main subject of this book. These methods are known by several essentially equivalent names: *reinforcement learning*, *approximate dynamic programming*, and *neuro-dynamic programming*. In this book, we will use primarily the most popular name: reinforcement learning (RL for short).

We first consider finite horizon problems, which involve a finite sequence of successive decisions, and are thus conceptually and analytically simpler. We defer the discussion of the more intricate infinite horizon problems to Chapters 4-6. We also discuss separately deterministic and stochastic problems (Sections 1.1 and 1.2, respectively). The reason is that deterministic problems are simpler and lend themselves better as an entry point to the optimal control methodology. Moreover, they have some favorable characteristics, which allow the application of a broader variety of methods. For example, simulation-based methods are greatly simplified and sometimes better understood in the context of deterministic optimal control.

Finally, in Section 1.3 we provide various examples of DP formulations, illustrating some of the concepts of Sections 1.1 and 1.2. The reader with substantial background in DP may wish to just scan Section 1.3 and skip to the next chapter, where we start the development of the approximate DP methodology.

1.1 DETERMINISTIC DYNAMIC PROGRAMMING

All DP problems involve a discrete-time dynamic system that generates a sequence of states under the influence of control. In finite horizon problems the system evolves over a finite number N of time steps (also called stages). The state and control at time k are denoted by x_k and u_k, respectively. In deterministic systems, x_{k+1} is generated nonrandomly, i.e., it is determined solely by x_k and u_k.

1.1.1 Deterministic Problems

A deterministic DP problem involves a discrete-time dynamic system of the form

$$x_{k+1} = f_k(x_k, u_k), \qquad k = 0, 1, \ldots, N-1, \tag{1.1}$$

where

k is the time index,

x_k is the state of the system, an element of some space,

u_k is the control or decision variable, to be selected at time k from some given set $U_k(x_k)$ that depends on x_k,

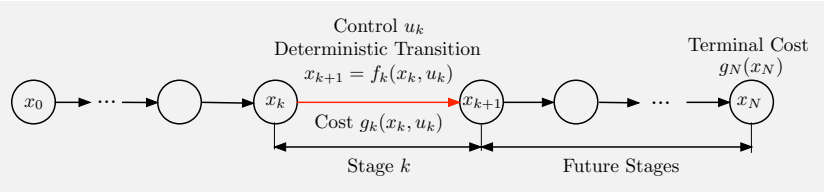

Figure 1.1.1 Illustration of a deterministic N-stage optimal control problem. Starting from state x_k, the next state under control u_k is generated nonrandomly, according to

$$x_{k+1} = f_k(x_k, u_k),$$

and a stage cost $g_k(x_k, u_k)$ is incurred.

f_k is a function of (x_k, u_k) that describes the mechanism by which the state is updated from time k to time $k+1$.

N is the horizon or number of times control is applied.

The set of all possible x_k is called the *state space* at time k. It can be any set and can depend on k; *this generality is one of the great strengths of the DP methodology*. Similarly, the set of all possible u_k is called the *control space* at time k. Again it can be any set and can depend on k.

The problem also involves a cost function that is additive in the sense that the cost incurred at time k, denoted by $g_k(x_k, u_k)$, accumulates over time. Formally, g_k is a function of (x_k, u_k) that takes real number values, and may depend on k. For a given initial state x_0, the total cost of a control sequence $\{u_0, \ldots, u_{N-1}\}$ is

$$J(x_0; u_0, \ldots, u_{N-1}) = g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k), \tag{1.2}$$

where $g_N(x_N)$ is a terminal cost incurred at the end of the process. The total cost is a well-defined number, since the control sequence $\{u_0, \ldots, u_{N-1}\}$ together with x_0 determines exactly the state sequence $\{x_1, \ldots, x_N\}$ via the system equation (1.1). We want to minimize the cost (1.2) over all sequences $\{u_0, \ldots, u_{N-1}\}$ that satisfy the control constraints, thereby obtaining the optimal value†

$$J^*(x_0) = \min_{\substack{u_k \in U_k(x_k) \\ k=0,\ldots,N-1}} J(x_0; u_0, \ldots, u_{N-1}),$$

as a function of x_0. Figure 1.1.1 illustrates the main elements of the problem.

We will next illustrate deterministic problems with some examples.

† We use throughout "min" (in place of "inf") to indicate minimal value over a feasible set of controls, even when we are not sure that the minimum is attained by some feasible control.

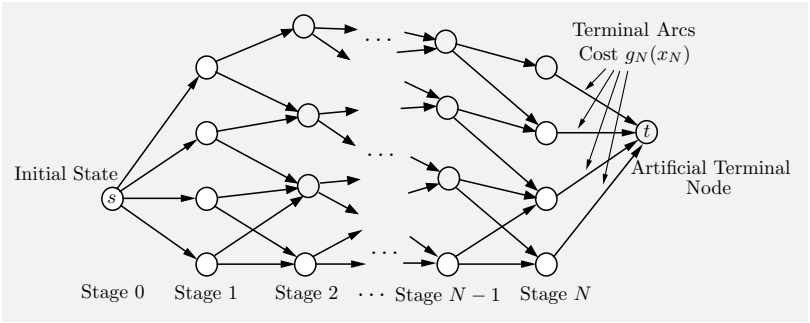

Figure 1.1.2 Transition graph for a deterministic finite-state system. Nodes correspond to states x_k. Arcs correspond to state-control pairs (x_k, u_k). An arc (x_k, u_k) has start and end nodes x_k and $x_{k+1} = f_k(x_k, u_k)$, respectively. We view the cost $g_k(x_k, u_k)$ of the transition as the length of this arc. The problem is equivalent to finding a shortest path from initial node s to terminal node t.

Discrete Optimal Control Problems

There are many situations where the state and control spaces are naturally discrete and consist of a finite number of elements. Such problems are often conveniently described with an acyclic graph specifying for each state x_k the possible transitions to next states x_{k+1}. The nodes of the graph correspond to states x_k and the arcs of the graph correspond to state-control pairs (x_k, u_k). Each arc with start node x_k corresponds to a choice of a single control $u_k \in U_k(x_k)$ and has as end node the next state $f_k(x_k, u_k)$. The cost of an arc (x_k, u_k) is defined as $g_k(x_k, u_k)$; see Fig. 1.1.2. To handle the final stage, an artificial terminal node t is added. Each state x_N at stage N is connected to the terminal node t with an arc having cost $g_N(x_N)$.

Note that control sequences $\{u_0, \ldots, u_{N-1}\}$ correspond to paths originating at the initial state (node s at stage 0) and terminating at one of the nodes corresponding to the final stage N. If we view the cost of an arc as its length, we see that *a deterministic finite-state finite-horizon problem is equivalent to finding a minimum-length (or shortest) path from the initial node s of the graph to the terminal node t.* Here, by the length of a path we mean the sum of the lengths of its arcs.†

Generally, combinatorial optimization problems can be formulated as deterministic finite-state finite-horizon optimal control problem. The following scheduling example illustrates the idea.

† It turns out also that any shortest path problem (with a possibly nonacyclic graph) can be reformulated as a finite-state deterministic optimal control problem, as we will show in Section 1.3.1. See [Ber17], Section 2.1, and [Ber91], [Ber98] for an extensive discussion of shortest path methods, which connects with our discussion here.

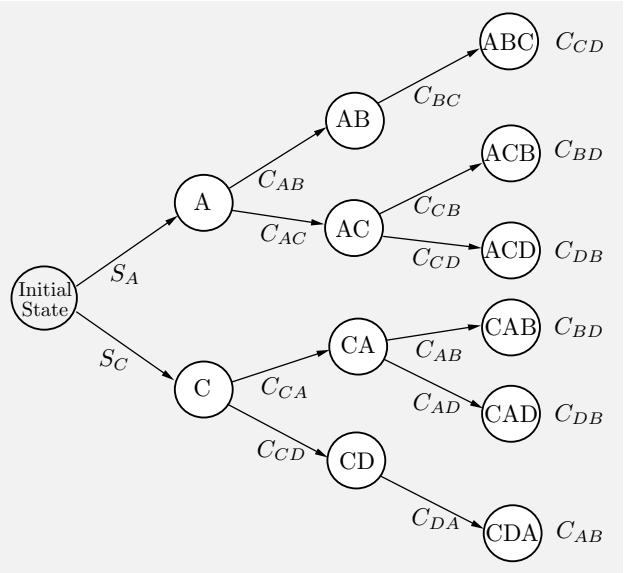

Figure 1.1.3 The transition graph of the deterministic scheduling problem of Example 1.1.1. Each arc of the graph corresponds to a decision leading from some state (the start node of the arc) to some other state (the end node of the arc). The corresponding cost is shown next to the arc. The cost of the last operation is shown as a terminal cost next to the terminal nodes of the graph.

Example 1.1.1 (A Deterministic Scheduling Problem)

Suppose that to produce a certain product, four operations must be performed on a certain machine. The operations are denoted by A, B, C, and D. We assume that operation B can be performed only after operation A has been performed, and operation D can be performed only after operation C has been performed. (Thus the sequence CDAB is allowable but the sequence CDBA is not.) The setup cost C_{mn} for passing from any operation m to any other operation n is given. There is also an initial startup cost S_A or S_C for starting with operation A or C, respectively (cf. Fig. 1.1.3). The cost of a sequence is the sum of the setup costs associated with it; for example, the operation sequence ACDB has cost

$$S_A + C_{AC} + C_{CD} + C_{DB}.$$

We can view this problem as a sequence of three decisions, namely the choice of the first three operations to be performed (the last operation is determined from the preceding three). It is appropriate to consider as state the set of operations already performed, the initial state being an artificial state corresponding to the beginning of the decision process. The possible state transitions corresponding to the possible states and decisions for this

problem are shown in Fig. 1.1.3. Here the problem is deterministic, i.e., at a given state, each choice of control leads to a uniquely determined state. For example, at state AC the decision to perform operation D leads to state ACD with certainty, and has cost C_{CD}. Thus the problem can be conveniently represented with the transition graph of Fig. 1.1.3. The optimal solution corresponds to the path that starts at the initial state and ends at some state at the terminal time and has minimum sum of arc costs plus the terminal cost.

Continuous-Spaces Optimal Control Problems

Many classical problems in control theory involve a continuous state space, such as a Euclidean space, i.e., the space of n-dimensional vectors of real variables, where n is some positive integer. The following is representative of the class of *linear-quadratic problems*, where the system equation is linear, the cost function is quadratic, and there are no control constraints. In our example, the states and controls are one-dimensional, but there are multidimensional extensions, which are very popular (see [Ber17], Section 3.1).

Example 1.1.2 (A Linear-Quadratic Problem)

A certain material is passed through a sequence of N ovens (see Fig. 1.1.4). Denote

x_0: initial temperature of the material,

x_k, $k = 1, \ldots, N$: temperature of the material at the exit of oven k,

u_{k-1}, $k = 1, \ldots, N$: heat energy applied to the material in oven k. In practice there will be some constraints on u_k, such as nonnegativity. However, for analytical tractability one may also consider the case where u_k is unconstrained, and check later if the solution satisfies some natural restrictions in the problem at hand.

We assume a system equation of the form

$$x_{k+1} = (1 - a)x_k + au_k, \qquad k = 0, 1, \ldots, N - 1,$$

where a is a known scalar from the interval $(0, 1)$. The objective is to get the final temperature x_N close to a given target T, while expending relatively little energy. We express this with a cost function of the form

$$r(x_N - T)^2 + \sum_{k=0}^{N-1} u_k^2,$$

where $r > 0$ is a given scalar that trades off the error in attaining the final temperature with the expended energy.

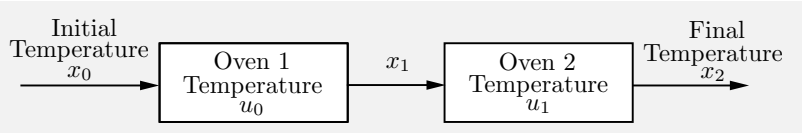

Figure 1.1.4 The linear-quadratic problem of Example 1.1.2 for $N = 2$. The temperature of the material evolves according to the system equation $x_{k+1} = (1 - a)x_k + au_k$, where a is some scalar with $0 < a < 1$.

Linear-quadratic problems with no constraints on the state or the control admit a nice analytical solution, as we will see later in Section 1.3.7. In another frequently arising optimal control problem there are linear constraints on the state and/or the control. In the preceding example it would have been natural to require that $a_k \leq x_k \leq b_k$ and/or $c_k \leq u_k \leq d_k$, where a_k, b_k, c_k, d_k are given scalars. Then the problem would be solvable not only by DP but also by quadratic programming methods. Generally deterministic optimal control problems with continuous state and control spaces (in addition to DP) admit a solution by nonlinear programming methods, such as gradient, conjugate gradient, and Newton's method, which can be suitably adapted to their special structure.

1.1.2 The Dynamic Programming Algorithm

In this section we will state the DP algorithm and formally justify it. The algorithm rests on a simple idea, the *principle of optimality*, which roughly states the following; see Fig. 1.1.5.

Principle of Optimality

Let $\{u_0^*, \ldots, u_{N-1}^*\}$ be an optimal control sequence, which together with x_0 determines the corresponding state sequence $\{x_1^*, \ldots, x_N^*\}$ via the system equation (1.1). Consider the subproblem whereby we start at x_k^* at time k and wish to minimize the "cost-to-go" from time k to time N,

$$g_k(x_k^*, u_k) + \sum_{m=k+1}^{N-1} g_m(x_m, u_m) + g_N(x_N),$$

over $\{u_k, \ldots, u_{N-1}\}$ with $u_m \in U_m(x_m)$, $m = k, \ldots, N-1$. Then the truncated optimal control sequence $\{u_k^*, \ldots, u_{N-1}^*\}$ is optimal for this subproblem.

The subproblem referred to above is called the *tail subproblem* that starts at x_k^*. Stated succinctly, the principle of optimality says that *the tail of an optimal sequence is optimal for the tail subproblem*. Its intuitive

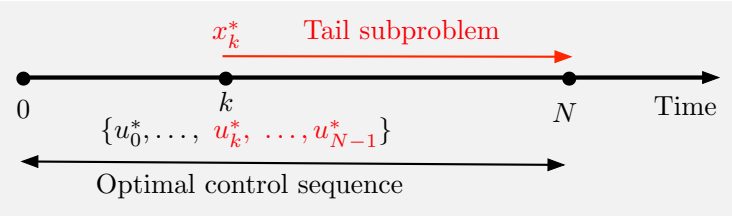

Figure 1.1.5 Illustration of the principle of optimality. The tail $\{u_k^*, \ldots, u_{N-1}^*\}$ of an optimal sequence $\{u_0^*, \ldots, u_{N-1}^*\}$ is optimal for the tail subproblem that starts at the state x_k^* of the optimal trajectory $\{x_1^*, \ldots, x_N^*\}$.

justification is simple. If the truncated control sequence $\{u_k^*, \ldots, u_{N-1}^*\}$ were not optimal as stated, we would be able to reduce the cost further by switching to an optimal sequence for the subproblem once we reach x_k^* (since the preceding choices of controls, $u_0^*, \ldots, u_{k-1}^*$, do not restrict our future choices). For an auto travel analogy, suppose that the fastest route from Los Angeles to Boston passes through Chicago. The principle of optimality translates to the obvious fact that the Chicago to Boston portion of the route is also the fastest route for a trip that starts from Chicago and ends in Boston.

The principle of optimality suggests that the optimal cost function can be constructed in piecemeal fashion going backwards: first compute the optimal cost function for the "tail subproblem" involving the last stage, then solve the "tail subproblem" involving the last two stages, and continue in this manner until the optimal cost function for the entire problem is constructed.

The DP algorithm is based on this idea: it proceeds sequentially, by *solving all the tail subproblems of a given time length, using the solution of the tail subproblems of shorter time length*. We illustrate the algorithm with the scheduling problem of Example 1.1.1. The calculations are simple but tedious, and may be skipped without loss of continuity. However, they may be worth going over by a reader that has no prior experience in the use of DP.

Example 1.1.1 (Scheduling Problem - Continued)

Let us consider the scheduling Example 1.1.1, and let us apply the principle of optimality to calculate the optimal schedule. We have to schedule optimally the four operations A, B, C, and D. There a cost for a transition between two operations, and the numerical values of the transition costs are shown in Fig. 1.1.6 next to the corresponding arcs.

According to the principle of optimality, the "tail" portion of an optimal schedule must be optimal. For example, suppose that the optimal schedule is CABD. Then, having scheduled first C and then A, it must be optimal to complete the schedule with BD rather than with DB. With this in mind, we

solve all possible tail subproblems of length two, then all tail subproblems of length three, and finally the original problem that has length four (the subproblems of length one are of course trivial because there is only one operation that is as yet unscheduled). As we will see shortly, the tail subproblems of length $k+1$ are easily solved once we have solved the tail subproblems of length k, and this is the essence of the DP technique.

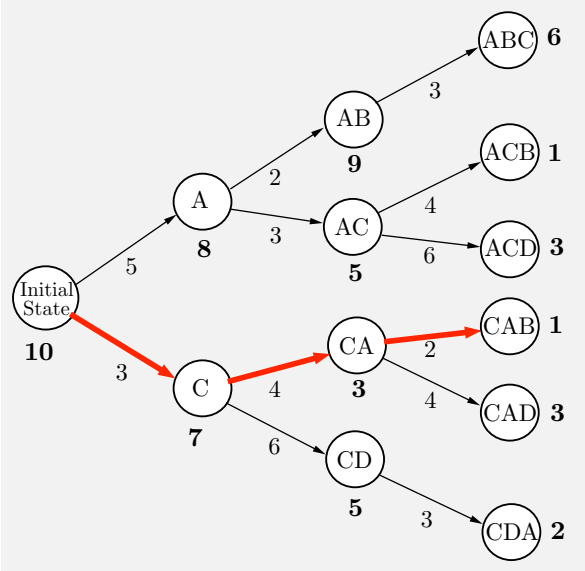

Figure 1.1.6 Transition graph of the deterministic scheduling problem, with the cost of each decision shown next to the corresponding arc. Next to each node/state we show the cost to optimally complete the schedule starting from that state. This is the optimal cost of the corresponding tail subproblem (cf. the principle of optimality). The optimal cost for the original problem is equal to 10, as shown next to the initial state. The optimal schedule corresponds to the thick-line arcs.

Tail Subproblems of Length 2: These subproblems are the ones that involve two unscheduled operations and correspond to the states AB, AC, CA, and CD (see Fig. 1.1.6).

State AB: Here it is only possible to schedule operation C as the next operation, so the optimal cost of this subproblem is 9 (the cost of scheduling C after B, which is 3, plus the cost of scheduling D after C, which is 6).

State AC: Here the possibilities are to (a) schedule operation B and then D, which has cost 5, or (b) schedule operation D and then B, which has cost 9. The first possibility is optimal, and the corresponding cost of the tail subproblem is 5, as shown next to node AC in Fig. 1.1.6.

State CA: Here the possibilities are to (a) schedule operation B and then D, which has cost 3, or (b) schedule operation D and then B, which has cost 7. The first possibility is optimal, and the corresponding cost of the tail subproblem is 3, as shown next to node CA in Fig. 1.1.6.

State CD: Here it is only possible to schedule operation A as the next operation, so the optimal cost of this subproblem is 5.

Tail Subproblems of Length 3: These subproblems can now be solved using the optimal costs of the subproblems of length 2.

State A: Here the possibilities are to (a) schedule next operation B (cost 2) and then solve optimally the corresponding subproblem of length 2 (cost 9, as computed earlier), a total cost of 11, or (b) schedule next operation C (cost 3) and then solve optimally the corresponding subproblem of length 2 (cost 5, as computed earlier), a total cost of 8. The second possibility is optimal, and the corresponding cost of the tail subproblem is 8, as shown next to node A in Fig. 1.1.6.

State C: Here the possibilities are to (a) schedule next operation A (cost 4) and then solve optimally the corresponding subproblem of length 2 (cost 3, as computed earlier), a total cost of 7, or (b) schedule next operation D (cost 6) and then solve optimally the corresponding subproblem of length 2 (cost 5, as computed earlier), a total cost of 11. The first possibility is optimal, and the corresponding cost of the tail subproblem is 7, as shown next to node A in Fig. 1.1.6.

Original Problem of Length 4: The possibilities here are (a) start with operation A (cost 5) and then solve optimally the corresponding subproblem of length 3 (cost 8, as computed earlier), a total cost of 13, or (b) start with operation C (cost 3) and then solve optimally the corresponding subproblem of length 3 (cost 7, as computed earlier), a total cost of 10. The second possibility is optimal, and the corresponding optimal cost is 10, as shown next to the initial state node in Fig. 1.1.6.

Note that having computed the optimal cost of the original problem through the solution of all the tail subproblems, we can construct the optimal schedule: we begin at the initial node and proceed forward, each time choosing the optimal operation, i.e., the one that starts the optimal schedule for the corresponding tail subproblem. In this way, by inspection of the graph and the computational results of Fig. 1.1.6, we determine that CABD is the optimal schedule.

Finding an Optimal Control Sequence by DP

We now state the DP algorithm for deterministic finite horizon problem by translating into mathematical terms the heuristic argument underlying the principle of optimality. The algorithm constructs functions

$$J_N^*(x_N), J_{N-1}^*(x_{N-1}), \ldots, J_0^*(x_0),$$

sequentially, starting from J_N^*, and proceeding backwards to J_{N-1}^*, J_{N-2}^*, etc.

DP Algorithm for Deterministic Finite Horizon Problems

Start with

$$J_N^*(x_N) = g_N(x_N), \qquad \text{for all } x_N, \tag{1.3}$$

and for $k = 0, \ldots, N-1$, let

$$J_k^*(x_k) = \min_{u_k \in U_k(x_k)} \Big[g_k(x_k, u_k) + J_{k+1}^*\big(f_k(x_k, u_k)\big) \Big], \qquad \text{for all } x_k. \tag{1.4}$$

Note that at stage k, the calculation in (1.4) must be done for all states x_k before proceeding to stage $k-1$. The key fact about the DP algorithm is that for every initial state x_0, the number $J_0^*(x_0)$ obtained at the last step, is equal to the optimal cost $J^*(x_0)$. Indeed, a more general fact can be shown, namely that for all $k = 0, 1, \ldots, N-1$, and all states x_k at time k, we have

$$J_k^*(x_k) = \min_{\substack{u_m \in U_m(x_m) \\ m=k,\ldots,N-1}} J(x_k; u_k, \ldots, u_{N-1}), \tag{1.5}$$

where

$$J(x_k; u_k, \ldots, u_{N-1}) = g_N(x_N) + \sum_{m=k}^{N-1} g_m(x_m, u_m), \tag{1.6}$$

i.e., $J_k^*(x_k)$ is the optimal cost for an $(N-k)$-stage tail subproblem that starts at state x_k and time k, and ends at time N.† Based on this fact, we

† We can prove this by induction. The assertion holds for $k = N$ in view of the initial condition $J_N^*(x_N) = g_N(x_N)$. To show that it holds for all k, we use Eqs. (1.5) and (1.6) to write

$$\begin{aligned} J_k^*(x_k) &= \min_{\substack{u_m \in U_m(x_m) \\ m=k,\ldots,N-1}} \left[g_N(x_N) + \sum_{m=k}^{N-1} g_m(x_m, u_m) \right] \\ &= \min_{u_k \in U_k(x_k)} \left[g_k(x_k, u_k) \right. \\ &\qquad \left. + \min_{\substack{u_m \in U_m(x_m) \\ m=k+1,\ldots,N-1}} \left[g_N(x_N) + \sum_{m=k+1}^{N-1} g_m(x_m, u_m) \right] \right] \\ &= \min_{u_k \in U_k(x_k)} \Big[g_k(x_k, u_k) + J_{k+1}^*\big(f_k(x_k, u_k)\big) \Big], \end{aligned}$$

call $J_k^*(x_k)$ the *optimal cost-to-go* at state x_k and time k, and refer to J_k^* as the *optimal cost-to-go function* or *optimal cost function* at time k. In maximization problems the DP algorithm (1.4) is written with maximization in place of minimization, and then J_k^* is referred to as the *optimal value function* at time k.

Note that the algorithm solves every tail subproblem, i.e., the minimization of the cost accumulated additively starting from an intermediate state up to the end of the horizon. Once the functions $J_0^*, \ldots, J_N^*$ have been obtained, we can use the following forward algorithm to construct an optimal control sequence $\{u_0^*, \ldots, u_{N-1}^*\}$ and corresponding state trajectory $\{x_1^*, \ldots, x_N^*\}$ for the given initial state x_0.

Construction of Optimal Control Sequence $\{u_0^*, \ldots, u_{N-1}^*\}$

Set

$$u_0^* \in \arg \min_{u_0 \in U_0(x_0)} \Big[g_0(x_0, u_0) + J_1^*\big(f_0(x_0, u_0)\big) \Big],$$

and

$$x_1^* = f_0(x_0, u_0^*).$$

Sequentially, going forward, for $k = 1, 2, \ldots, N-1$, set

$$u_k^* \in \arg \min_{u_k \in U_k(x_k^*)} \Big[g_k(x_k^*, u_k) + J_{k+1}^*\big(f_k(x_k^*, u_k)\big) \Big], \qquad (1.7)$$

and

$$x_{k+1}^* = f_k(x_k^*, u_k^*).$$

The same algorithm can be used to find an optimal control sequence for any tail subproblem. Figure 1.1.6 traces the calculations of the DP algorithm for the scheduling Example 1.1.1. The numbers next to the nodes, give the corresponding cost-to-go values, and the thick-line arcs give the construction of the optimal control sequence using the preceding algorithm.

1.1.3 Approximation in Value Space

The preceding forward optimal control sequence construction is possible only after we have computed $J_k^*(x_k)$ by DP for all x_k and k. Unfortunately, in practice this is often prohibitively time-consuming, because of

where for the last equality we use the induction hypothesis. A subtle mathematical point here is that, through the minimization operation, the cost-to-go functions J_k^* may take the value $-\infty$ for some x_k. Still the preceding induction argument is valid even if this is so.

the number of possible x_k and k can be very large. However, a similar forward algorithmic process can be used if the optimal cost-to-go functions J_k^* are replaced by some approximations $\tilde{J}_k$. This is the basis for *approximation in value space*, which will be central in our future discussions. It constructs a suboptimal solution $\{\tilde{u}_0, \ldots, \tilde{u}_{N-1}\}$ in place of the optimal $\{u_0^*, \ldots, u_{N-1}^*\}$, based on using $\tilde{J}_k$ in place of J_k^* in the DP procedure (1.7).

Approximation in Value Space - Use of $\tilde{J}_k$ in Place of J_k^*

Start with

$$\tilde{u}_0 \in \arg\min_{u_0 \in U_0(x_0)} \Big[g_0(x_0, u_0) + \tilde{J}_1\big(f_0(x_0, u_0)\big) \Big],$$

and set

$$\tilde{x}_1 = f_0(x_0, \tilde{u}_0).$$

Sequentially, going forward, for $k = 1, 2, \ldots, N-1$, set

$$\tilde{u}_k \in \arg\min_{u_k \in U_k(\tilde{x}_k)} \Big[g_k(\tilde{x}_k, u_k) + \tilde{J}_{k+1}\big(f_k(\tilde{x}_k, u_k)\big) \Big], \qquad (1.8)$$

and

$$\tilde{x}_{k+1} = f_k(\tilde{x}_k, \tilde{u}_k).$$

The construction of suitable approximate cost-to-go functions $\tilde{J}_k$ is a major focal point of the RL methodology. There are several possible methods, depending on the context, and they will be taken up starting with the next chapter.

Q-Factors and Q-Learning

The expression

$$\tilde{Q}_k(x_k, u_k) = g_k(x_k, u_k) + \tilde{J}_{k+1}\big(f_k(x_k, u_k)\big),$$

which appears in the right-hand side of Eq. (1.8) is known as the (approximate) *Q-factor of* (x_k, u_k).† In particular, the computation of the

† The term "Q-learning" and some of the associated algorithmic ideas were introduced in the thesis by Watkins [Wat89] (after the symbol "Q" that he used to represent Q-factors). The term "Q-factor" was used in the book [BeT96], and is adopted here as well. Watkins [Wat89] used the term "action value" (at a given state). The terms "state-action value" and "Q-value" are also common in the literature.

approximately optimal control (1.8) can be done through the Q-factor minimization

$$\tilde{u}_k \in \arg \min_{u_k \in U_k(\tilde{x}_k)} \tilde{Q}_k(\tilde{x}_k, u_k).$$

This suggests the possibility of using Q-factors in place of cost functions in approximation in value space schemes. Methods of this type use as starting point an alternative (and equivalent) form of the DP algorithm, which instead of the optimal cost-to-go functions J_k^*, generates the *optimal Q-factors*, defined for all pairs (x_k, u_k) and k by

$$Q_k^*(x_k, u_k) = g_k(x_k, u_k) + J_{k+1}^*\big(f_k(x_k, u_k)\big). \tag{1.9}$$

Thus the optimal Q-factors are simply the expressions that are minimized in the right-hand side of the DP equation (1.4). Note that this equation implies that the optimal cost function J_k^* can be recovered from the optimal Q-factor Q_k^* by means of

$$J_k^*(x_k) = \min_{u_k \in U_k(x_k)} Q_k^*(x_k, u_k).$$

Moreover, using the above relation, the DP algorithm can be written in an essentially equivalent form that involves Q-factors only

$$Q_k^*(x_k, u_k) = g_k(x_k, u_k) + \min_{u_{k+1} \in U_{k+1}(f_k(x_k, u_k))} Q_{k+1}^*\big(f_k(x_k, u_k), u_{k+1}\big).$$

We will discuss later exact and approximate forms of related algorithms in the context of a class of RL methods known as *Q-learning*.

1.2 STOCHASTIC DYNAMIC PROGRAMMING

The stochastic finite horizon optimal control problem differs from the deterministic version primarily in the nature of the discrete-time dynamic system that governs the evolution of the state x_k. This system includes a random "disturbance" w_k, which is characterized by a probability distribution $P_k(\cdot \mid x_k, u_k)$ that may depend explicitly on x_k and u_k, but not on values of prior disturbances $w_{k-1}, \ldots, w_0$. The system has the form

$$x_{k+1} = f_k(x_k, u_k, w_k), \qquad k = 0, 1, \ldots, N-1,$$

where as before x_k is an element of some state space S_k, the control u_k is an element of some control space. The cost per stage is denoted $g_k(x_k, u_k, w_k)$ and also depends on the random disturbance w_k; see Fig. 1.2.1. The control

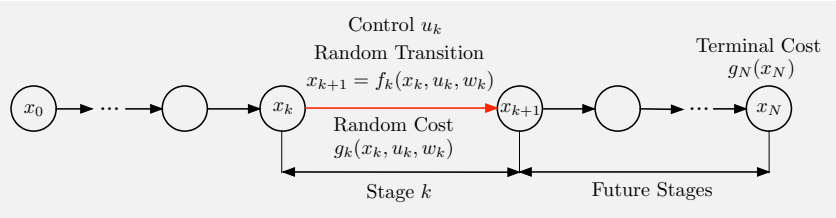

Figure 1.2.1 Illustration of an N-stage stochastic optimal control problem. Starting from state x_k, the next state under control u_k is generated randomly, according to

$$x_{k+1} = f_k(x_k, u_k, w_k),$$

where w_k is the random disturbance, and a random stage cost $g_k(x_k, u_k, w_k)$ is incurred.

u_k is constrained to take values in a given subset $U_k(x_k)$, which depends on the current state x_k.

An important difference is that we optimize not over control sequences $\{u_0, \ldots, u_{N-1}\}$, but rather over *policies* (also called *closed-loop control laws*, or *feedback policies*) that consist of a sequence of functions

$$\pi = \{\mu_0, \ldots, \mu_{N-1}\},$$

where μ_k maps states x_k into controls $u_k = \mu_k(x_k)$, and satisfies the control constraints, i.e., is such that $\mu_k(x_k) \in U_k(x_k)$ for all $x_k \in S_k$. Policies are more general objects than control sequences, and in the presence of stochastic uncertainty, they can result in improved cost, since they allow choices of controls u_k that incorporate knowledge of the state x_k. Without this knowledge, the controller cannot adapt appropriately to unexpected values of the state, and as a result the cost can be adversely affected. This is a fundamental distinction between deterministic and stochastic optimal control problems.

Another important distinction between deterministic and stochastic problems is that in the latter, the evaluation of various quantities such as cost function values involves forming expected values. Consequently, several of the methods that we will discuss for stochastic problems will involve the use of Monte Carlo simulation.

Given an initial state x_0 and a policy $\pi = \{\mu_0, \ldots, \mu_{N-1}\}$, the future states x_k and disturbances w_k are random variables with distributions defined through the system equation

$$x_{k+1} = f_k\big(x_k, \mu_k(x_k), w_k\big), \qquad k = 0, 1, \ldots, N-1.$$

Thus, for given functions g_k, $k = 0, 1, \ldots, N$, the expected cost of π starting at x_0 is

$$J_\pi(x_0) = E\left\{ g_N(x_N) + \sum_{k=0}^{N-1} g_k\big(x_k, \mu_k(x_k), w_k\big) \right\},$$

where the expected value operation $E\{\cdot\}$ is over all the random variables w_k and x_k. An optimal policy π^* is one that minimizes this cost; i.e.,

$$J_{\pi^*}(x_0) = \min_{\pi \in \Pi} J_\pi(x_0),$$

where Π is the set of all policies.

The optimal cost depends on x_0 and is denoted by $J^*(x_0)$; i.e.,

$$J^*(x_0) = \min_{\pi \in \Pi} J_\pi(x_0).$$

It is useful to view J^* as a function that assigns to each initial state x_0 the optimal cost $J^*(x_0)$, and call it the *optimal cost function* or *optimal value function*.

Finite Horizon Stochastic Dynamic Programming

The DP algorithm for the stochastic finite horizon optimal control problem has a similar form to its deterministic version, and shares several of its major characteristics:

(a) Using tail subproblems to break down the minimization over multiple stages to single stage minimizations.

(b) Generating backwards for all k and x_k the values $J_k^*(x_k)$, which give the optimal cost-to-go starting at stage k at state x_k.

(c) Obtaining an optimal policy by minimization in the DP equations.

(d) A structure that is suitable for approximation in value space, whereby we replace J_k^* by approximations $\tilde{J}_k$, and obtain a suboptimal policy by the corresponding minimization.

DP Algorithm for Stochastic Finite Horizon Problems

Start with

$$J_N^*(x_N) = g_N(x_N),$$

and for $k = 0, \ldots, N-1$, let

$$J_k^*(x_k) = \min_{u_k \in U_k(x_k)} E\Big\{ g_k(x_k, u_k, w_k) + J_{k+1}^*\big(f_k(x_k, u_k, w_k)\big) \Big\}. \quad (1.10)$$

If $u_k^* = \mu_k^*(x_k)$ minimizes the right side of this equation for each x_k and k, the policy $\pi^* = \{\mu_0^*, \ldots, \mu_{N-1}^*\}$ is optimal.

The key fact is that for every initial state x_0, the optimal cost $J^*(x_0)$ is equal to the function $J_0^*(x_0)$, obtained at the last step of the above DP

algorithm. This can be proved by induction similar to the deterministic case; we will omit the proof (see the discussion of Section 1.3 in the textbook [Ber17]).†

Simultaneously with the off-line computation of the optimal cost-to-go functions $J_0^*, \ldots, J_N^*$, we can compute and store an optimal policy $\pi^* = \{\mu_0^*, \ldots, \mu_{N-1}^*\}$ by minimization in Eq. (1.10). We can then use this policy on-line to retrieve from memory and apply the control $\mu_k^*(x_k)$ once we reach state x_k.

The alternative is to forego the storage of the policy π^* and to calculate the control $\mu_k^*(x_k)$ by executing the minimization (1.10) on-line. This method of on-line control calculation is called *one-step lookahead minimization*. Its main use is not so much in the context of exact DP, but rather in the context of approximate DP methods that involve approximation in value space. There, approximations $\tilde{J}_k$ are used in place of J_k^*, similar to the deterministic case; cf. Eqs. (1.7) and (1.8).

Approximation in Value Space - Use of $\tilde{J}_k$ in Place of J_k^*

At any state x_k encountered at stage k, compute and apply the control

$$\tilde{\mu}_k(x_k) \in \arg \min_{u_k \in U_k(x_k)} E\Big\{ g_k(x_k, u_k, w_k) + \tilde{J}_{k+1}\big(f_k(x_k, u_k, w_k)\big) \Big\}. \tag{1.11}$$

As in deterministic problems, the motivation for approximation in value space is that the DP algorithm can be very time-consuming. The one-step lookahead minimization (1.11) needs to be performed only for the N states $x_0, \ldots, x_{N-1}$ that are encountered during the on-line control of the system, and not for every state within the potentially enormous state space. Of course this simplification entails the loss of optimality, and requires the construction of suitable approximate cost-to-go functions $\tilde{J}_k$. This is a major focal point of the RL methodology, and will be discussed at length in the following chapters.

Q-Factors for Stochastic Problems

We can define optimal Q-factors for a stochastic problem, similar to the

† There are some technical/mathematical difficulties here, having to do with the expected value operation in Eq. (1.10) being well-defined and finite. These difficulties are of no concern in practice, and disappear completely when the disturbance spaces w_k can take only a finite number of values, in which case all expected values consist of sums of finitely many real number terms. For a mathematical treatment, see the relevant discussion in Chapter 1 of [Ber17] and the book [BeS78].

case of deterministic problems [cf. Eq. (1.9)], as the expressions that are minimized in the right-hand side of the stochastic DP equation (1.10). They are given by

$$Q_k^*(x_k, u_k) = E\Big\{ g_k(x_k, u_k, w_k) + J_{k+1}^*\big(f_k(x_k, u_k, w_k)\big) \Big\}.$$

The optimal cost-to-go functions J_k^* can be recovered from the optimal Q-factors Q_k^* by means of

$$J_k^*(x_k) = \min_{u_k \in U_k(x_k)} Q_k^*(x_k, u_k),$$

and the DP algorithm can be written in terms of Q-factors as

$$Q_k^*(x_k, u_k) = E\Big\{ g_k(x_k, u_k, w_k) + \min_{u_{k+1} \in U_{k+1}(f_k(x_k, u_k, w_k))} Q_{k+1}^*\big(f_k(x_k, u_k, w_k), u_{k+1}\big) \Big\}.$$

1.3 EXAMPLES, VARIATIONS, AND SIMPLIFICATIONS

In this section we provide some examples to illustrate problem formulation techniques, solution methods, and adaptations of the basic DP algorithm to various contexts. As a guide for formulating optimal control problems in a manner that is suitable for DP solution, the following two-stage process is suggested:

(a) Identify the controls/decisions u_k and the times k at which these controls are applied. Usually this step is fairly straightforward. However, in some cases there may be some choices to make. For example in deterministic problems, where the objective is to select an optimal sequence of controls $\{u_0, \ldots, u_{N-1}\}$, one may lump multiple controls to be chosen together, e.g., view the pair (u_0, u_1) as a single choice. This is usually not possible in stochastic problems, where distinct decisions are differentiated by the information/feedback available when making them.

(b) Select the states x_k. The basic guideline here is that x_k should encompass all the information that is known to the controller at time k and can be used with advantage in choosing u_k. In effect, at time k the state x_k should separate the past from the future, in the sense that anything that has happened in the past (states, controls, and disturbances from stages prior to stage k) is irrelevant to the choices of future controls as long as we know x_k. Sometimes this is described

by saying that the state should have a "Markov property" to express an analogy with states of Markov chains, where (by definition) the conditional probability distribution of future states depends on the past history of the chain only through the present state.

Note that there may be multiple possibilities for selecting the states, because information may be packaged in several different ways that are equally useful from the point of view of control. It is thus worth considering alternative ways to choose the states; for example try to use states that minimize the dimensionality of the state space. For a trivial example that illustrates the point, if a quantity x_k qualifies as state, then (x_{k-1}, x_k) also qualifies as state, since (x_{k-1}, x_k) contains all the information contained within x_k that can be useful to the controller when selecting u_k. However, using (x_{k-1}, x_k) in place of x_k, gains nothing in terms of optimal cost while complicating the DP algorithm that would have to be executed over a larger space. The concept of a *sufficient statistic*, which refers to a quantity that summarizes all the essential content of the information available to the controller, may be useful in reducing the size of the state space (see the discussion in Section 3.1.1, and in [Ber17], Section 4.3). Section 1.3.6 provides an example, and Section 3.1.1 contains further discussion.

Generally minimizing the dimension of the state makes sense but there are exceptions. A case in point is problems involving *partial* or *imperfect* state information, where we collect measurements to use for control of some quantity of interest y_k that evolves over time (for example, y_k may be the position/velocity vector of a moving vehicle). If I_k is the collection of all measurements and controls up to time k, it is correct to use I_k as state. However, a better alternative may be to use as state the conditional probability distribution $P_k(y_k \mid I_k)$, called *belief state*, which may subsume all the information that is useful for the purposes of choosing a control. On the other hand, the belief state $P_k(y_k \mid I_k)$ is an infinite-dimensional object, whereas I_k may be finite dimensional, so the best choice may be problem-dependent; see the textbooks [Ber17] and [Kri16] for further discussion of partial state information problems.

We refer to DP textbooks for extensive additional discussions of modeling and problem formulation techniques. The subsequent chapters do not rely substantially on the material of the present section, so the reader may selectively skip forward to the next chapter and return to this material later as needed.

1.3.1 Deterministic Shortest Path Problems

Consider a directed graph with a special node, called the *destination*. Let $\{1, 2, \ldots, N, t\}$ be the set of nodes of the graph, where t is the destination, and let a_{ij} be the cost of moving from node i to node j [also referred to as the *length* of the directed arc (i, j) that joins i and j]. By a path we mean a sequence of arcs such that the end node of each arc in the sequence is

the start node of the next arc. The length of a path from a given node to another node is the sum of the lengths of the arcs on the path. We want to find a shortest (i.e., minimum length) path from each node i to node t.

We make an assumption relating to cycles, i.e., paths of the form $(i, j_1), (j_1, j_2), \ldots, (j_k, i)$ that start and end at the same node. In particular, we exclude the possibility that a cycle has negative total length. Otherwise, it would be possible to decrease the length of some paths to arbitrarily small values simply by adding more and more negative-length cycles. We thus assume that *all cycles have nonnegative length*. With this assumption, it is clear that an optimal path need not take more than N moves, so we may limit the number of moves to N. To conform to the N-stage DP framework, we formulate the problem as one where *we require exactly N moves but allow degenerate moves from a node i to itself with cost $a_{ii} = 0$*. We also assume that *for every node i there exists at least one path from i to t*, so that the problem has at least one solution.

We can formulate this problem as a deterministic DP problem with N stages, where the states at any stage $0, \ldots, N-1$ are the nodes $\{1, \ldots, N\}$, the destination t is the unique state at stage N, and the controls correspond to the arcs (i, j), including the self arcs (i, i). Thus at each state i we select a control (i, j) and move to state j at cost a_{ij}.

We can write the DP algorithm for our problem, with the optimal cost-to-go functions J_k^* having the meaning

$$J_k^*(i) = \text{optimal cost of getting from } i \text{ to } t \text{ in } N - k \text{ moves},$$

for $i = 1, \ldots, N$ and $k = 0, \ldots, N-1$. The cost of the optimal path from i to t is $J_0^*(i)$. The DP algorithm takes the intuitively clear form

$$\begin{aligned}&\text{optimal cost from } i \text{ to } t \text{ in } N-k \text{ moves}\\ &\quad = \min_{\text{All arcs } (i,j)} \big[a_{ij} + (\text{optimal cost from } j \text{ to } t \text{ in } N-k-1 \text{ moves})\big],\end{aligned}$$

or

$$J_k^*(i) = \min_{\text{All arcs } (i,j)} \big[a_{ij} + J_{k+1}^*(j)\big], \qquad k = 0, 1, \ldots, N-2,$$

with

$$J_{N-1}^*(i) = a_{it}, \qquad i = 1, 2, \ldots, N.$$

This algorithm is also known as the *Bellman-Ford algorithm* for shortest paths. It is one of the most popular shortest path algorithms.

The optimal policy when at node i after k moves is to move to a node j^* that minimizes $a_{ij} + J_{k+1}^*(j)$ over all j such that (i, j) is an arc. If the optimal path obtained from the algorithm contains degenerate moves from a node to itself, this simply means that the path involves in reality less than N moves.

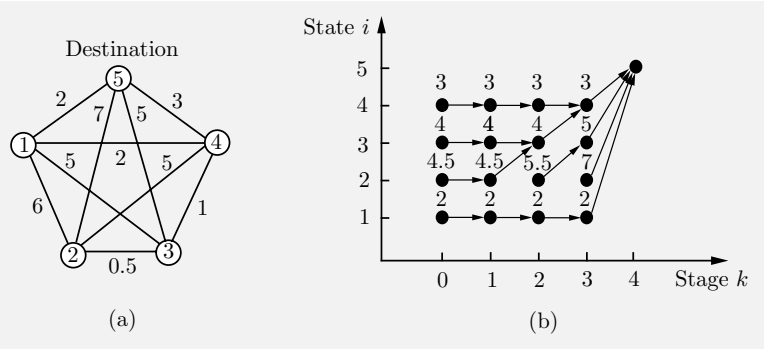

Figure 1.3.1 (a) Shortest path problem data. The destination is node 5. Arc lengths are equal in both directions and are shown along the line segments connecting nodes. (b) Costs-to-go generated by the DP algorithm. The number along stage k and state i is $J_k^*(i)$. Arrows indicate the optimal moves at each stage and node. The optimal paths that start from nodes 1,2,3,4 are $1 \to 5,\ 2 \to 3 \to 4 \to 5,\ 3 \to 4 \to 5,\ 4 \to 5$, respectively.

Note that if for some $k > 0$, we have

$$J_k^*(i) = J_{k+1}^*(i), \qquad \text{for all } i,$$

then subsequent DP iterations will not change the values of the cost-to-go $[J_{k-m}^*(i) = J_k^*(i)$ for all $m > 0$ and $i]$, so the algorithm can be terminated with $J_k^*(i)$ being the shortest distance from i to t, for all i.

To demonstrate the algorithm, consider the problem shown in Fig. 1.3.1(a) where the costs a_{ij} with $i \neq j$ are shown along the connecting line segments (we assume that $a_{ij} = a_{ji}$). Figure 1.3.1(b) shows the optimal cost-to-go $J_k^*(i)$ at each i and k together with the optimal paths.

1.3.2 Discrete Deterministic Optimization

Discrete optimization problems can be typically formulated as DP problems by breaking down each feasible solution into a sequence of decisions/controls; see e.g., the scheduling Example 1.1.1. This formulation will often lead to an intractable DP computation because of an exponential explosion of the number of states. However, it brings to bear approximate DP methods, such as rollout and others that we will discuss in future chapters. We illustrate the reformulation by an example and then generalize.

Example 1.3.1 (The Traveling Salesman Problem)

An important model for scheduling a sequence of operations is the classical traveling salesman problem. Here we are given N cities and the travel time between each pair of cities. We wish to find a minimum time travel that visits

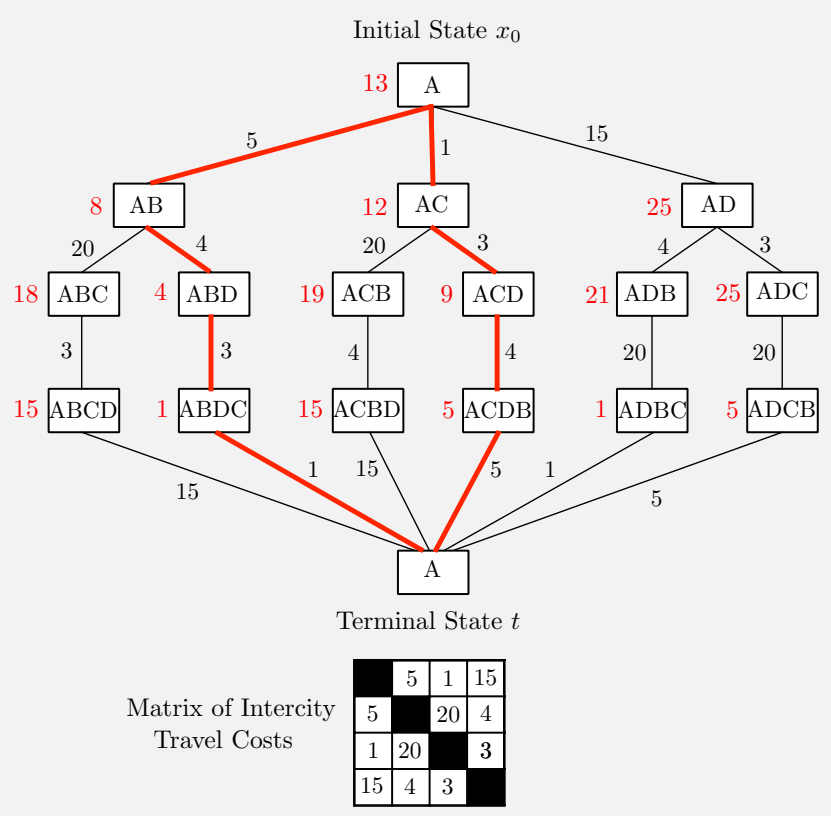

Figure 1.3.2 Example of a DP formulation of the traveling salesman problem. The travel times between the four cities A, B, C, and D are shown in the matrix at the bottom. We form a graph whose nodes are the k-city sequences and correspond to the states of the kth stage. The transition costs/travel times are shown next to the arcs. The optimal costs-to-go are generated by DP starting from the terminal state and going backwards towards the initial state, and are shown next to the nodes. There are two optimal sequences here (ABDCA and ACDBA), and they are marked with thick lines. Both optimal sequences can be obtained by forward minimization [cf. Eq. (1.7)], starting from the initial state x_0.

each of the cities exactly once and returns to the start city. To convert this problem to a DP problem, we form a graph whose nodes are the sequences of k distinct cities, where $k = 1, \ldots, N$. The k-city sequences correspond to the states of the kth stage. The initial state x_0 consists of some city, taken as the start (city A in the example of Fig. 1.3.2). A k-city node/state leads to a $(k+1)$-city node/state by adding a new city at a cost equal to the travel time between the last two of the $k+1$ cities; see Fig. 1.3.2. Each sequence of N cities is connected to an artificial terminal node t with an arc of cost equal to the travel time from the last city of the sequence to the starting city, thus

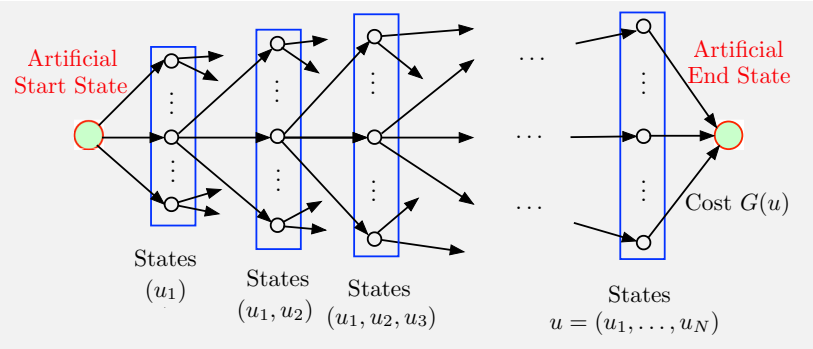

Figure 1.3.3. Formulation of a discrete optimization problem as a DP problem with $N + 1$ stages. There is a cost $G(u)$ only at the terminal stage on the arc connecting an N-solution $u = (u_1, \ldots, u_N)$ to the artificial terminal state. Alternative formulations may use fewer states by taking advantage of the problem's structure.

completing the transformation to a DP problem.

The optimal costs-to-go from each node to the terminal state can be obtained by the DP algorithm and are shown next to the nodes. Note, however, that the number of nodes grows exponentially with the number of cities N. This makes the DP solution intractable for large N. As a result, large traveling salesman and related scheduling problems are typically addressed with approximation methods, some of which are based on DP, and will be discussed as part of our subsequent development.

Let us now extend the ideas of the preceding example to the general discrete optimization problem:

$$\begin{aligned} &\text{minimize} \quad G(u) \\ &\text{subject to} \quad u \in U, \end{aligned}$$

where U is a finite set of feasible solutions and $G(u)$ is a cost function. We assume that each solution u has N components; i.e., it has the form $u = (u_1, \ldots, u_N)$, where N is a positive integer. We can then view the problem as a sequential decision problem, where the components $u_1, \ldots, u_N$ are selected one-at-a-time. A k-tuple $(u_1, \ldots, u_k)$ consisting of the first k components of a solution is called a *k-solution*. We associate k-solutions with the kth stage of the finite horizon DP problem shown in Fig. 1.3.3. In particular, for $k = 1, \ldots, N$, we view as the states of the kth stage all the k-tuples $(u_1, \ldots, u_k)$. The initial state is an artificial state denoted s. From this state we may move to any state (u_1), with u_1 belonging to the set

$$U_1 = \big\{\tilde{u}_1 \mid \text{there exists a solution of the form } (\tilde{u}_1, \tilde{u}_2, \ldots, \tilde{u}_N) \in U\big\}.$$

Thus U_1 is the set of choices of u_1 that are consistent with feasibility.

More generally, from a state $(u_1, \ldots, u_k)$, we may move to any state of the form $(u_1, \ldots, u_k, u_{k+1})$, with u_{k+1} belonging to the set

$$U_{k+1}(u_1, \ldots, u_k) = \big\{ \tilde{u}_{k+1} \mid \text{there exists a solution of the form } (u_1, \ldots, u_k, \tilde{u}_{k+1}, \ldots, \tilde{u}_N) \in U \big\}.$$

At state $(u_1, \ldots, u_k)$ we must choose u_{k+1} from the set $U_{k+1}(u_1, \ldots, u_k)$. These are the choices of u_{k+1} that are consistent with the preceding choices $u_1, \ldots, u_k$, and are also consistent with feasibility. The last stage corresponds to the N-solutions $u = (u_1, \ldots, u_N)$, which in turn lead to an artificial end state t. The cost-to-go from each u in the last stage to t is $G(u)$, the cost of the solution u; see Fig. 1.3.3. All other transitions in this DP problem formulation have cost 0.

Let $J_k^*(u_1, \ldots, u_k)$ denote the optimal cost starting from the k-solution $(u_1, \ldots, u_k)$, i.e., the optimal cost of the problem over solutions whose first k components are constrained to be equal to u_i, $i = 1, \ldots, k$, respectively. The DP algorithm is described by the equation

$$J_k^*(u_1, \ldots, u_k) = \min_{u_{k+1} \in U_{k+1}(u_1, \ldots, u_k)} J_{k+1}^*(u_1, \ldots, u_k, u_{k+1}), \qquad (1.12)$$

with the terminal condition

$$J_N^*(u_1, \ldots, u_N) = G(u_1, \ldots, u_N).$$

The algorithm (1.12) executes backwards in time: starting with the known function $J_N^* = G$, we compute J_{N-1}^*, then J_{N-2}^*, and so on up to computing J_1^*. An optimal solution $(u_1^*, \ldots, u_N^*)$ is then constructed by going forward through the algorithm

$$u_{k+1}^* \in \arg \min_{u_{k+1} \in U_{k+1}(u_1^*, \ldots, u_k^*)} J_{k+1}^*(u_1^*, \ldots, u_k^*, u_{k+1}), \quad k = 0, \ldots, N-1, \qquad (1.13)$$

first compute u_1^*, then u_2^*, and so on up to u_N^*; cf. Eq. (1.7).

Of course here the number of states typically grows exponentially with N, but we can use the DP minimization (1.13) as a starting point for the use of approximation methods. For example we may try to use approximation in value space, whereby we replace J_{k+1}^* with some suboptimal $\tilde{J}_{k+1}$ in Eq. (1.13). One possibility is to use as

$$\tilde{J}_{k+1}(u_1^*, \ldots, u_k^*, u_{k+1}),$$

the cost generated by a heuristic method that solves the problem suboptimally with the values of the first $k+1$ decision components fixed at $u_1^*, \ldots, u_k^*, u_{k+1}$. This is called a *rollout algorithm*, and it is a very simple and effective approach for approximate combinatorial optimization. It will

be discussed later in this book, in Chapter 2 for finite horizon stochastic problems, and in Chapter 5 for infinite horizon problems, where it will be related to the method of policy iteration and self-learning ideas.

Finally, let us mention that shortest path and discrete optimization problems with a sequential character can be addressed by a variety of approximate shortest path methods. These include the so called *label correcting, A^*, and branch and bound methods* for which extensive discussions can be found in the literature. The author's DP textbook [Ber17] (Chapter 2) contains a substantial account, which connects with the material of this section, as well as with a much more detailed discussion of shortest path methods in the author's network optimization textbook [Ber98].

1.3.3 Problems with a Termination State

Many DP problems of interest involve a *termination state*, i.e., a state t that is cost-free and absorbing in the sense that for all k,

$$g_k(t, u_k, w_k) = 0, \quad f_k(t, u_k, w_k) = t, \qquad \text{for all } w_k \text{ and } u_k \in U_k(t).$$

Thus the control process essentially terminates upon reaching t, even if this happens before the end of the horizon. One may reach t by choice if a special stopping decision is available, or by means of a random transition from another state.

Generally, when it is known that an optimal policy will reach the termination state within at most some given number of stages N, the DP problem can be formulated as an N-stage horizon problem.† The reason is that even if the termination state t is reached at a time $k < N$, we can extend our stay at t for an additional $N - k$ stages at no additional cost. An illustration of this was given in the deterministic shortest path problem that we discussed in Section 1.3.1.

Discrete deterministic optimization problems generally have a close connection to shortest path problems as we have seen in Section 1.3.2. In the problem discussed in that section, the termination state is reached after exactly N stages (cf. Fig. 1.3.3), but in other problems it is possible that termination can happen earlier. The following well known puzzle is an example.

Example 1.3.2 (The Four Queens Problem)

Four queens must be placed on a 4×4 portion of a chessboard so that no queen can attack another. In other words, the placement must be such that every row, column, or diagonal of the 4×4 board contains at most one queen.

† When an upper bound on the number of stages to termination is not known, the problem must be formulated as an infinite horizon problem, as will be discussed in a subsequent chapter.

Equivalently, we can view the problem as a sequence of problems; first, placing a queen in one of the first two squares in the top row, then placing another queen in the second row so that it is not attacked by the first, and similarly placing the third and fourth queens. (It is sufficient to consider only the first two squares of the top row, since the other two squares lead to symmetric positions; this is an example of a situation where we have a choice between several possible state spaces, but we select the one that is smallest.)

We can associate positions with nodes of an acyclic graph where the root node s corresponds to the position with no queens and the terminal nodes correspond to the positions where no additional queens can be placed without some queen attacking another. Let us connect each terminal position with an artificial terminal node t by means of an arc. Let us also assign to all arcs cost zero except for the artificial arcs connecting terminal positions with less than four queens with the artificial node t. These latter arcs are assigned a cost of 1 (see Fig. 1.3.4) to express the fact that they correspond to dead-end positions that cannot lead to a solution. Then, the four queens problem reduces to finding a minimal cost path from node s to node t, with an optimal sequence of queen placements corresponding to cost 0.

Note that once the states/nodes of the graph are enumerated, the problem is essentially solved. In this 4×4 problem the states are few and can be easily enumerated. However, we can think of similar problems with much larger state spaces. For example consider the problem of placing N queens on an $N \times N$ board without any queen attacking another. Even for moderate values of N, the state space for this problem can be extremely large (for $N = 8$ the number of possible placements with exactly one queen in each row is $8^8 = 16,777,216$). It can be shown that there exist solutions to the N queens problem for all $N \geq 4$ (for $N = 2$ and $N = 3$, clearly there is no solution).

There are also several variants of the N queens problem. For example finding the minimal number of queens that can be placed on an $N \times N$ board so that they either occupy or attack every square; this is known as the *queen domination problem*. The minimal number can be found in principle by DP, and it is known for some N (for example the minimal number is 5 for $N = 8$), but not for all N (see e.g., the paper by Fernau [Fer10]).

1.3.4 Forecasts

Consider a situation where at time k the controller has access to a forecast y_k that results in a reassessment of the probability distribution of the subsequent disturbance w_k and, possibly, future disturbances. For example, y_k may be an exact prediction of w_k or an exact prediction that the probability distribution of w_k is a specific one out of a finite collection of distributions. Forecasts of interest in practice are, for example, probabilistic predictions on the state of the weather, the interest rate for money, and the demand for inventory.

Generally, forecasts can be handled by introducing additional state

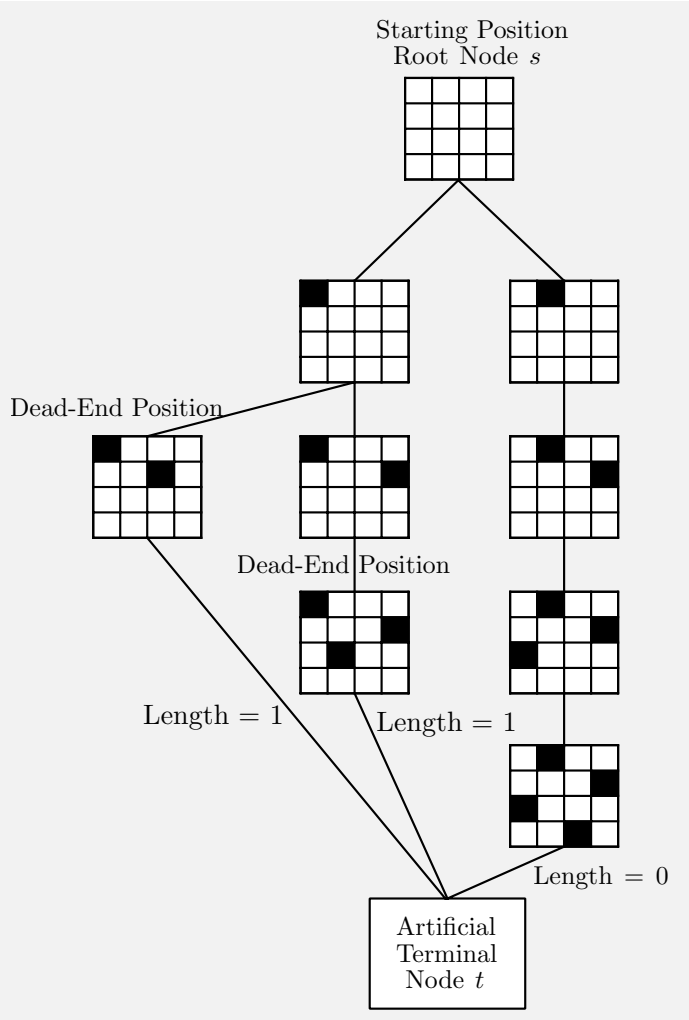

Figure 1.3.4 Discrete optimization formulation of the four queens problem. Symmetric positions resulting from placing a queen in one of the rightmost squares in the top row have been ignored. Squares containing a queen have been darkened. All arcs have length zero except for those connecting dead-end positions to the artificial terminal node.

variables corresponding to the information that the forecasts provide.† We

† The device of introducing additional states to modify a given problem so that it fits the DP formalism is known as *state augmentation*. It finds extensive use in the DP formulation of practical problems (see e.g. [Ber17]). Examples of problem reformulations that are based on state augmentation arise when

will illustrate the process with a simple example.

Assume that at the beginning of each stage k, the controller receives an accurate prediction that the next disturbance w_k will be selected according to a particular probability distribution out of a given collection of distributions $\{P_1, \ldots, P_m\}$; i.e., if the forecast is i, then w_k is selected according to P_i. The a priori probability that the forecast will be i is denoted by p_i and is given.

The forecasting process can be represented by means of the equation

$$y_{k+1} = \xi_k,$$

where y_{k+1} can take the values $1, \ldots, m$, corresponding to the m possible forecasts, and ξ_k is a random variable taking the value i with probability p_i. The interpretation here is that when ξ_k takes the value i, then w_{k+1} will occur according to the distribution P_i.

By combining the system equation with the forecast equation $y_{k+1} = \xi_k$, we obtain an augmented system given by

$$\begin{pmatrix} x_{k+1} \\ y_{k+1} \end{pmatrix} = \begin{pmatrix} f_k(x_k, u_k, w_k) \\ \xi_k \end{pmatrix}.$$

The new state is

$$\tilde{x}_k = (x_k, y_k).$$

The new disturbance is

$$\tilde{w}_k = (w_k, \xi_k),$$

and its probability distribution is determined by the distributions P_i and the probabilities p_i, and depends explicitly on $\tilde{x}_k$ (via y_k) but not on the prior disturbances.

Thus, by suitable reformulation of the cost, the problem can be cast as a stochastic DP problem. Note that the control applied depends on both the current state and the current forecast. The DP algorithm takes the form

$$\begin{aligned} J_N^*(x_N, y_N) &= g_N(x_N), \\ J_k^*(x_k, y_k) &= \min_{u_k \in U_k(x_k)} \mathop{E}_{w_k} \Big\{ g_k(x_k, u_k, w_k) \\ &\qquad + \sum_{i=1}^{m} p_i J_{k+1}^*\big(f_k(x_k, u_k, w_k), i\big) \;\Big|\; y_k \Big\}, \end{aligned} \tag{1.14}$$

---

the disturbances exhibit correlations over time. They also arise in the presence of *post-decision* states, where the system function $f_k(x_k, u_k, w_k)$ has the form $\tilde{f}_k\big(h_k(x_k, u_k), w_k\big)$, where $\tilde{f}_k$ is some function, and $h_k(x_k, u_k)$ represents an intermediate "state" that occurs after the control is applied. With post-decision states, the DP algorithm may be reformulated to a simpler form (see [Ber12], Section 6.1.5).

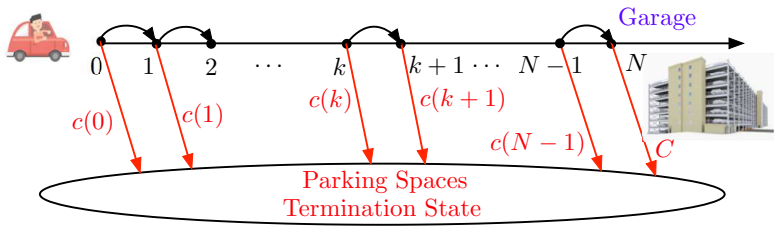

Figure 1.3.5 Cost structure of the parking problem. The driver may park at space $k = 0, 1, \ldots, N-1$ at cost $c(k)$, if the space is free, or continue to the next space $k+1$ at no cost. At space N (the garage) the driver must park at cost C.

where y_k may take the values $1, \ldots, m$, and the expectation over w_k is taken with respect to the distribution P_{y_k}.

It should be clear that the preceding formulation admits several extensions. One example is the case where forecasts can be influenced by the control action (e.g., pay extra for a more accurate forecast) and involve several future disturbances. However, the price for these extensions is increased complexity of the corresponding DP algorithm.

1.3.5 Problems with Uncontrollable State Components

In many problems of interest the natural state of the problem consists of several components, some of which cannot be affected by the choice of control. In such cases the DP algorithm can be simplified considerably, and be executed over the controllable components of the state. Before describing how this can be done in generality, let us consider an example.

Example 1.3.3 (Parking)

A driver is looking for inexpensive parking on the way to his destination. The parking area contains N spaces, numbered $0, \ldots, N-1$, and a garage following space $N-1$. The driver starts at space 0 and traverses the parking spaces sequentially, i.e., from space k he goes next to space $k+1$, etc. Each parking space k costs $c(k)$ and is free with probability $p(k)$ independently of whether other parking spaces are free or not. If the driver reaches the last parking space $N-1$ and does not park there, he must park at the garage, which costs C. The driver can observe whether a parking space is free only when he reaches it, and then, if it is free, he makes a decision to park in that space or not to park and check the next space. The problem is to find the minimum expected cost parking policy.

We formulate the problem as a DP problem with N stages, corresponding to the parking spaces, and an artificial termination state that corresponds to having parked; see Fig. 1.3.5. At each stage $k = 1, \ldots, N-1$, we have three states: the artificial termination state, denoted by (k, t), and the two states (k, F) and $(k, \overline{F})$, corresponding to space k being free or taken, respectively.

At stage 0, we have only two states, $(0, F)$ and $(0, \overline{F})$, and at the final stage there is only one state, the termination state t. The decision/control is to park or continue at state (k, F) [there is no choice at states $(k, \overline{F})$ and states (k, t), $k = 1, \ldots, N-1$]. The termination state t is reached at cost $c(k)$ when a parking decision is made at the states (k, F), $k = 0, \ldots, N-1$, at cost C, when the driver continues at states $(N-1, F)$ or $(N-1, \overline{F})$, and at no cost at (k, t), $k = 1, \ldots, N-1$.

Let us now derive the form of the DP algorithm, denoting:

$J_k^*(F)$: The optimal cost-to-go upon arrival at a space k that is free.

$J_k^*(\overline{F})$: The optimal cost-to-go upon arrival at a space k that is taken.

$J_k^*(t)$: The cost-to-go of the "parked"/termination state t.

The DP algorithm for $k = 0, \ldots, N-1$ takes the form

$$J_k^*(F) = \begin{cases} \min\big[c(k),\, p(k+1)J_{k+1}^*(F) + \big(1 - p(k+1)\big)J_{k+1}^*(\overline{F})\big] & \text{if } k < N-1, \\ \min\big[c(N-1),\, C\big] & \text{if } k = N-1, \end{cases}$$

$$J_k^*(\overline{F}) = \begin{cases} p(k+1)J_{k+1}^*(F) + \big(1 - p(k+1)\big)J_{k+1}^*(\overline{F}) & \text{if } k < N-1, \\ C & \text{if } k = N-1, \end{cases}$$

for the states other than the termination state t, while for t we have

$$J_k^*(t) = 0, \qquad k = 1, \ldots, N.$$

While this algorithm is easily executed, it can be written in a simpler and equivalent form, which takes advantage of the fact that the second component (F or $\overline{F}$) of the state is uncontrollable. This can be done by introducing the scalars

$$\hat{J}_k = p(k)J_k^*(F) + \big(1 - p(k)\big)J_k^*(\overline{F}), \qquad k = 0, \ldots, N-1,$$

which can be viewed as the optimal expected cost-to-go upon arriving at space k but *before verifying its free or taken status*.

Indeed, from the preceding DP algorithm, we have

$$\hat{J}_{N-1} = p(N-1)\min\big[c(N-1),\, C\big] + \big(1 - p(N-1)\big)C,$$

$$\hat{J}_k = p(k)\min\big[c(k),\, \hat{J}_{k+1}\big] + \big(1 - p(k)\big)\hat{J}_{k+1}, \qquad k = 0, \ldots, N-2.$$

From this algorithm we can also obtain the optimal parking policy, which is to park at space $k = 0, \ldots, N-1$ if it is free and we have $c(k) \leq \hat{J}_{k+1}$.

Figure 1.3.6 provides a plot for $\hat{J}_k$ for the case where

$$p(k) \equiv 0.05, \quad c(k) = N - k, \quad C = 100, \quad N = 200. \tag{1.15}$$

The optimal policy is to travel to space 165 and then to park at the first available space. The reader may verify that this type of policy, characterized by a single threshold distance, is optimal not just for the form of $c(k)$ given

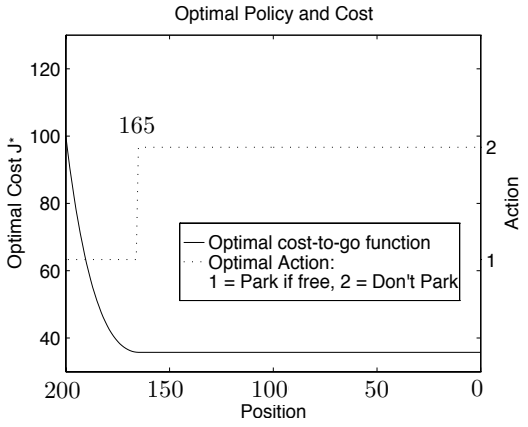

Figure 1.3.6 Optimal cost-to-go and optimal policy for the parking problem with the data in Eq. (1.15). The optimal policy is to travel from space 0 to space 165 and then to park at the first available space.

above, but also for any form of $c(k)$ that is monotonically decreasing as k increases.

We will now formalize the procedure illustrated in the preceding example. Let the state of the system be a composite (x_k, y_k) of two components x_k and y_k. The evolution of the main component, x_k, is affected by the control u_k according to the equation

$$x_{k+1} = f_k(x_k, y_k, u_k, w_k),$$

where the distribution $P_k(w_k \mid x_k, y_k, u_k)$ is given. The evolution of the other component, y_k, is governed by a given conditional distribution $P_k(y_k \mid x_k)$ and cannot be affected by the control, except indirectly through x_k. One is tempted to view y_k as a disturbance, but there is a difference: y_k is observed by the controller before applying u_k, while w_k occurs after u_k is applied, and indeed w_k may probabilistically depend on u_k.

It turns out that we can formulate a DP algorithm that is executed over the controllable component of the state, with the dependence on the uncontrollable component being "averaged out" as in the preceding example. In particular, let $J_k^*(x_k, y_k)$ denote the optimal cost-to-go at stage k and state (x_k, y_k), and define

$$\hat{J}_k(x_k) = \mathop{E}_{y_k}\big\{ J_k^*(x_k, y_k) \mid x_k \big\}.$$

Then, similar to the preceding parking example, a DP algorithm that gen-

erates $\hat{J}_k(x_k)$ can be obtained, and has the following form:†

$$\hat{J}_k(x_k) = \mathop{E}_{y_k}\Big\{ \min_{u_k \in U_k(x_k,y_k)} \mathop{E}_{w_k}\Big\{g_k(x_k,y_k,u_k,w_k) + \hat{J}_{k+1}\big(f_k(x_k,y_k,u_k,w_k)\big) \;\Big|\; x_k,y_k,u_k\Big\}\Big|\; x_k\Big\}. \tag{1.16}$$

Note that the minimization in the right-hand side of the preceding equation must still be performed for all values of the full state (x_k, y_k) in order to yield an optimal control law as a function of (x_k, y_k). Nonetheless, the equivalent DP algorithm (1.16) has the advantage that it is executed over a significantly reduced state space. Later, when we consider approximation in value space, we will find that it is often more convenient to approximate $\hat{J}_k(x_k)$ than to approximate $J_k^*(x_k, y_k)$; see the following discussions of forecasts and of the game of tetris.

As an example, consider the augmented state resulting from the incorporation of forecasts, as described earlier in Section 1.3.4. Then, the forecast y_k represents an uncontrolled state component, so that the DP algorithm can be simplified as in Eq. (1.16). In particular, using the notation of Section 1.3.4, by defining

$$\hat{J}_k(x_k) = \sum_{i=1}^{m} p_i J_k^*(x_k, i), \qquad k = 0, 1, \ldots, N-1,$$

and

$$\hat{J}_N(x_N) = g_N(x_N),$$

we have, using Eq. (1.14),

$$\hat{J}_k(x_k) = \sum_{i=1}^{m} p_i \min_{u_k \in U_k(x_k)} \mathop{E}_{w_k}\Big\{g_k(x_k,u_k,w_k) + \hat{J}_{k+1}\big(f_k(x_k,u_k,w_k)\big) \;\Big|\; y_k = i\Big\},$$

† This is a consequence of the calculation

$$\begin{aligned}
\hat{J}_k(x_k) &= E_{y_k}\big\{J_k^*(x_k,y_k) \mid x_k\big\} \\
&= E_{y_k}\Big\{ \min_{u_k \in U_k(x_k,y_k)} E_{w_k,x_{k+1},y_{k+1}}\big\{g_k(x_k,y_k,u_k,w_k) \\
&\qquad + J_{k+1}^*(x_{k+1},y_{k+1}) \;\big|\; x_k,y_k,u_k\big\} \;\big|\; x_k\Big\} \\
&= E_{y_k}\Big\{ \min_{u_k \in U_k(x_k,y_k)} E_{w_k,x_{k+1}}\Big\{g_k(x_k,y_k,u_k,w_k) \\
&\qquad + E_{y_{k+1}}\big\{J_{k+1}^*(x_{k+1},y_{k+1}) \;\big|\; x_{k+1}\big\} \;\big|\; x_k,y_k,u_k\Big\} \;\big|\; x_k\Big\}.
\end{aligned}$$

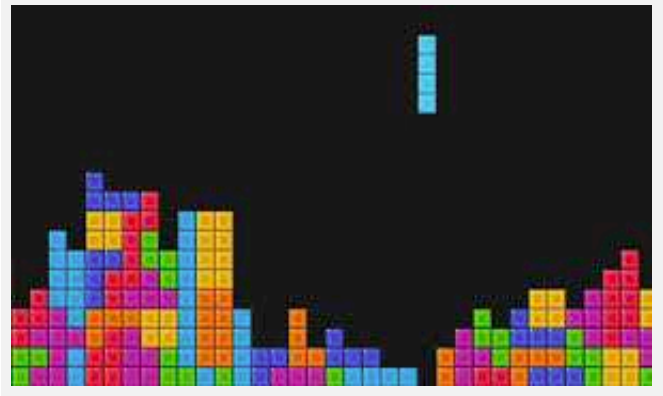

Figure 1.3.7 Illustration of a tetris board.

which is executed over the space of x_k rather than x_k *and* y_k. Note that this is a simpler algorithm to approximate than the one of Eq. (1.14).

Uncontrollable state components often occur in arrival systems, such as queueing, where action must be taken in response to a random event (such as a customer arrival) that cannot be influenced by the choice of control. Then the state of the arrival system must be augmented to include the random event, but the DP algorithm can be executed over a smaller space, as per Eq. (1.16). Here is an example of this type.

Example 1.3.4 (Tetris)

Tetris is a popular video game played on a two-dimensional grid. Each square in the grid can be full or empty, making up a "wall of bricks" with "holes" and a "jagged top" (see Fig. 1.3.7). The squares fill up as blocks of different shapes fall from the top of the grid and are added to the top of the wall. As a given block falls, the player can move horizontally and rotate the block in all possible ways, subject to the constraints imposed by the sides of the grid and the top of the wall. The falling blocks are generated independently according to some probability distribution, defined over a finite set of standard shapes. The game starts with an empty grid and ends when a square in the top row becomes full and the top of the wall reaches the top of the grid. When a row of full squares is created, this row is removed, the bricks lying above this row move one row downward, and the player scores a point. The player's objective is to maximize the score attained (total number of rows removed) within N steps or up to termination of the game, whichever occurs first.

We can model the problem of finding an optimal tetris playing strategy as a stochastic DP problem. The control, denoted by u, is the horizontal positioning and rotation applied to the falling block. The state consists of two components:

(1) The board position, i.e., a binary description of the full/empty status of each square, denoted by x.

(2) The shape of the current falling block, denoted by y.

There is also an additional termination state which is cost-free. Once the state reaches the termination state, it stays there with no change in cost.

The shape y is generated according to a probability distribution $p(y)$, independently of the control, so it can be viewed as an uncontrollable state component. The DP algorithm (1.16) is executed over the space of board positions x and has the intuitive form

$$\hat{J}_k(x) = \sum_y p(y) \max_u \Big[g(x,y,u) + \hat{J}_{k+1}\big(f(x,y,u)\big) \Big], \qquad \text{for all } x,$$

where

$g(x,y,u)$ is the number of points scored (rows removed),

$f(x,y,u)$ is the next board position (or termination state),

when the state is (x,y) and control u is applied, respectively. Note, however, that despite the simplification in the DP algorithm achieved by eliminating the uncontrollable portion of the state, the number of states x is still enormous, and the problem can only be addressed by suboptimal methods, which will be discussed later in this book.

1.3.6 Partial State Information and Belief States

We have assumed so far that the controller has access to the exact value of the current state x_k, so a policy consists of a sequence of functions $\mu_k(x_k)$, $k = 0, \ldots, N-1$. However, in many practical settings this assumption is unrealistic, because some components of the state may be inaccessible for measurement, the sensors used for measuring them may be inaccurate, or the cost of obtaining accurate measurements may be prohibitive.

Often in such situations the controller has access to only some of the components of the current state, and the corresponding measurements may also be corrupted by stochastic uncertainty. For example in three-dimensional motion problems, the state may consist of the six-tuple of position and velocity components, but the measurements may consist of noise-corrupted radar measurements of the three position components. This gives rise to problems of *partial* or *imperfect* state information, which have received a lot of attention in the optimization and artificial intelligence literature (see e.g., [Ber17], [RuN16]). Even though there are DP algorithms for partial information problems, these algorithms are far more computationally intensive than their perfect information counterparts. For this reason, in the absence of an analytical solution, partial information problems are typically solved suboptimally in practice.

On the other hand it turns out that conceptually, partial state information problems are no different than the perfect state information problems we have been addressing so far. In fact by various reformulations, we can reduce a partial state information problem to one with perfect state

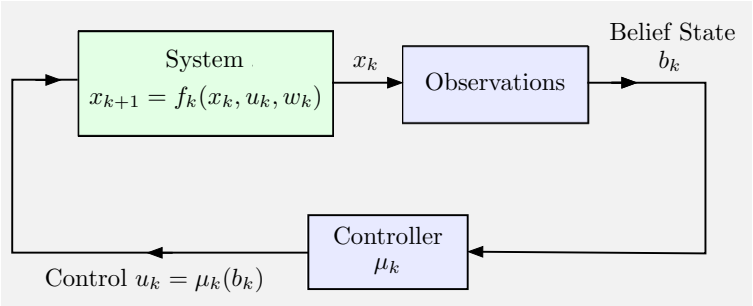

Figure 1.3.8 Schematic illustration of a control system with imperfect state observations. The belief state b_k is the conditional probability distribution of x_k given all the observations up to time k.

information (see [Ber17], Ch. 4). The most common approach is to replace the state x_k with a *belief state*, which we will often denote by b_k. It is the probability distribution of x_k given all the observations that have been obtained by the controller up to time k (see Fig. 1.3.8). This probability distribution can in principle be computed, and it can serve as "state" in an appropriate DP algorithm. We illustrate this process with a simple example.

Example 1.3.5 (Treasure Hunting)

In a classical problem of search, one has to decide at each of N periods whether to search a site that may contain a treasure. If a treasure is present, the search reveals it with probability ξ, in which case the treasure is removed from the site. Here the state x_k has two values: either a treasure is present in the site or it is not. The control u_k takes two values: search and not search. If the site is searched, we obtain an observation, which takes one of two values: treasure found or not found. If the site is not searched, no information is obtained.

Denote

b_k : probability a treasure is present at the beginning of period k given the search results so far.

This is the belief state at time k and it evolves according to the equation

$$b_{k+1} = \begin{cases} b_k & \text{if the site is not searched at time } k, \\ 0 & \text{if the site is searched and a treasure is found,} \\ \dfrac{b_k(1-\xi)}{b_k(1-\xi)+1-b_k} & \text{if the site is searched but no treasure is found.} \end{cases} \tag{1.17}$$

The third relation above follows by application of Bayes' rule (b_{k+1} is equal to the kth period probability of a treasure being present *and* the search being unsuccessful, divided by the probability of an unsuccessful search). The second relation holds because the treasure is removed after a successful search.

Let us view b_k as the state of a "belief system" given by Eq. (1.17), and write a DP algorithm, assuming that the treasure's worth is V, that each search costs C. Denoting by $J_k^*(b)$ the optimal cost-to-go from belief state b at time k, the algorithm takes the form

$$J_k^*(b_k) = \max \left[J_{k+1}^*(b_k), \right. \\ \left. - C + b_k \xi V + (1 - b_k \xi) J_{k+1}^* \left(\frac{b_k(1-\xi)}{b_k(1-\xi) + 1 - b_k} \right) \right], \tag{1.18}$$

with $J_N^*(b_N) = 0$. The two options in the maximization of Eq. (1.18) correspond to not searching (in which case b_k remains unchanged), and searching [in which case b_k evolves according to Eq. (1.17)].

Thanks to the simplicity of the problem, this DP algorithm can be used to obtain an analytical solution. In particular, it is straightforward to show by induction (starting with $k = N - 1$) that the functions J_k^* satisfy $J_k^*(b_k) \geq 0$ for all $b_k \in [0, 1]$ and

$$J_k^*(b_k) = 0 \qquad \text{if} \qquad b_k \leq \frac{C}{\xi V}.$$

From this it follows that it is optimal to search at period k if and only if

$$\frac{C}{\xi V} \leq b_k.$$

Thus, it is optimal to search if and only if the expected reward from the next search, $b_k \xi V$, is greater or equal to the cost C of the search - a myopic policy that focuses on just the next stage.

Of course the preceding example is extremely simple, involving a state x_k that takes just two values. As a result, the belief state b_k takes values within the interval $[0, 1]$. Still there are infinitely many values in this interval, and if a computational solution were necessary, the belief state would have to be discretized and the DP algorithm (1.18) would have to be accordingly modified and executed over the discretized state space (discretization methods will be discussed in Chapter 6).

In problems where the state x_k can take a finite but large number of values, say n, the belief states comprise an n-dimensional simplex, so discretization becomes problematic. As a result, alternative suboptimal solution methods are often used in partial state information problems. Some of these methods will be described in future chapters.

The following is a simple example of a partial state information problem whose belief state has large enough size to make an exact DP solution impossible.

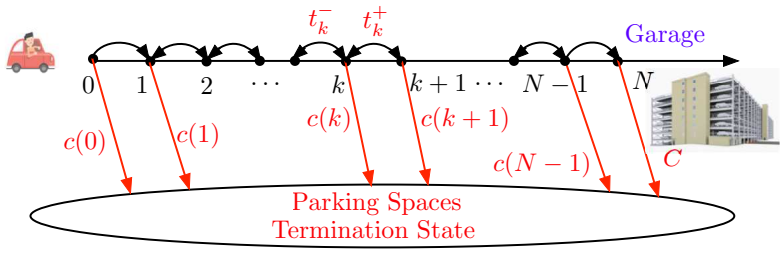

Figure 1.3.9 Cost structure and transitions of the bidirectional parking problem. The driver may park at space $k = 0, 1, \ldots, N-1$ at cost $c(k)$, if the space is free, can move to $k-1$ at cost t_k^- or can move to $k+1$ at cost t_k^+. At space N (the garage) the driver must park at cost C.

Example 1.3.6 (Bidirectional Parking)

Let us consider a more complex version of the parking problem of Example 1.3.3. As in that example, a driver is looking for inexpensive parking on the way to his destination, along a line of N parking spaces with a garage at the end. The difference is that the driver can move in either direction, rather than just forward towards the garage. In particular, at space i, the driver can park at cost $c(i)$ if i is free, can move to $i-1$ at a cost t_i^- or can move to $i+1$ at a cost t_i^+. Moreover, the driver records the free/taken status of the spaces previously visited and may return to any of these spaces; see Fig. 1.3.9.

Let us assume that the probability $p(i)$ of a space i being free changes over time, i.e., a space found free (or taken) at a given visit may get taken (or become free, respectively) by the time of the next visit. The initial probabilities $p(i)$, before visiting any spaces, are known, and the mechanism by which these probabilities change over time is also known to the driver. As an example, we may assume that at each time period, $p(i)$ increases by a certain known factor with some probability ξ and decreases by another known factor with the complementary probability $1-\xi$.

Here the belief state is the vector of current probabilities

$$\big(p(0), \ldots, p(N)\big),$$

and it is updated at each time based on the new observation: the free/taken status of the space visited at that time. This belief state can be computed exactly by the driver, given the parking status observations of the spaces visited thus far. While it is possible to state an exact DP algorithm that is defined over the set of belief states, and we will do so later, the algorithm is impossible to execute in practice.† Thus the problem can only be solved approximately, using methods that we will discuss in subsequent chapters.

† The problem as stated is an infinite horizon problem because there is nothing to prevent the driver from moving forever in the parking lot without ever parking. We can convert the problem to a similarly difficult finite horizon problem by restricting the number of moves to a given upper limit $\overline{N} > N$, and requiring that if the driver is at distance of k spaces from the garage at time $\overline{N} - k$, then driving in the direction away from the garage is not an option.

1.3.7 Linear Quadratic Optimal Control

In a few exceptional special cases the DP algorithm yields an analytical solution, which can be used among other purposes, as a starting point for approximate DP schemes to solve related problems. Prominent among such cases are various linear quadratic optimal control problems, which involve a linear (possibly multidimensional) system, a quadratic cost function, and no constraints on the control. Let us illustrate this with the deterministic scalar linear quadratic Example 1.1.2. We will apply the DP algorithm for the case of just two stages ($N = 2$), and illustrate the method for obtaining a nice analytical solution.

As defined in Example 1.1.2, the terminal cost is

$$g_2(x_2) = r(x_2 - T)^2.$$

Thus the DP algorithm starts with

$$J_2^*(x_2) = g_2(x_2) = r(x_2 - T)^2,$$

[cf. Eq. (1.3)].

For the next-to-last stage, we have [cf. Eq. (1.4)]

$$J_1^*(x_1) = \min_{u_1}\big[u_1^2 + J_2^*(x_2)\big] = \min_{u_1}\Big[u_1^2 + J_2^*\big((1-a)x_1 + au_1\big)\Big].$$

Substituting the previous form of J_2^*, we obtain

$$J_1^*(x_1) = \min_{u_1}\Big[u_1^2 + r\big((1-a)x_1 + au_1 - T\big)^2\Big]. \tag{1.19}$$

This minimization will be done by setting to zero the derivative with respect to u_1. This yields

$$0 = 2u_1 + 2ra\big((1-a)x_1 + au_1 - T\big),$$

and by collecting terms and solving for u_1, we obtain the optimal temperature for the last oven as a function of x_1:

$$\mu_1^*(x_1) = \frac{ra\big(T - (1-a)x_1\big)}{1 + ra^2}. \tag{1.20}$$

By substituting the optimal u_1 in the expression (1.19) for J_1^*, we obtain

$$\begin{aligned} J_1^*(x_1) &= \frac{r^2a^2\big((1-a)x_1 - T\big)^2}{(1+ra^2)^2} + r\left((1-a)x_1 + \frac{ra^2\big(T-(1-a)x_1\big)}{1+ra^2} - T\right)^2 \\ &= \frac{r^2a^2\big((1-a)x_1 - T\big)^2}{(1+ra^2)^2} + r\left(\frac{ra^2}{1+ra^2} - 1\right)^2\big((1-a)x_1 - T\big)^2 \\ &= \frac{r\big((1-a)x_1 - T\big)^2}{1+ra^2}. \end{aligned}$$

We now go back one stage. We have [cf. Eq. (1.4)]

$$J_0^*(x_0) = \min_{u_0}\big[u_0^2 + J_1^*(x_1)\big] = \min_{u_0}\Big[u_0^2 + J_1^*\big((1-a)x_0 + au_0\big)\Big],$$

and by substituting the expression already obtained for J_1^*, we have

$$J_0^*(x_0) = \min_{u_0}\left[u_0^2 + \frac{r\big((1-a)^2x_0 + (1-a)au_0 - T\big)^2}{1+ra^2}\right].$$

We minimize with respect to u_0 by setting the corresponding derivative to zero. We obtain

$$0 = 2u_0 + \frac{2r(1-a)a\big((1-a)^2x_0 + (1-a)au_0 - T\big)}{1+ra^2}.$$

This yields, after some calculation, the optimal temperature of the first oven:

$$\mu_0^*(x_0) = \frac{r(1-a)a\big(T - (1-a)^2x_0\big)}{1+ra^2\big(1+(1-a)^2\big)}. \tag{1.21}$$

The optimal cost is obtained by substituting this expression in the formula for J_0^*. This leads to a straightforward but lengthy calculation, which in the end yields the rather simple formula

$$J_0^*(x_0) = \frac{r\big((1-a)^2x_0 - T\big)^2}{1+ra^2\big(1+(1-a)^2\big)}.$$

This completes the solution of the problem.

Note that the algorithm has simultaneously yielded an optimal policy $\{\mu_0^*, \mu_1^*\}$ via Eqs. (1.21) and (1.20): a rule that tells us the optimal oven temperatures $u_0 = \mu_0^*(x_0)$ and $u_1 = \mu_1^*(x_1)$ for every possible value of the states x_0 and x_1, respectively. Thus the DP algorithm (as expected) solves all the tail subproblems and provides a feedback policy.

A noteworthy feature in this example is the facility with which we obtained an analytical solution. A little thought while tracing the steps of the algorithm will convince the reader that what simplifies the solution is the quadratic nature of the cost and the linearity of the system equation [see the derivation of Eq. (1.20)]. Indeed, it can be shown in generality that when the system is linear and the cost is quadratic, the optimal policy and cost-to-go function are given by closed-form expressions, regardless of the number of stages N (see [Ber17], Section 3.1).

Stochastic Linear Quadratic Problems - Certainty Equivalence

Let us now introduce a zero-mean stochastic additive disturbance in the linear system equation. Remarkably, it turns out that the optimal policy remains unaffected. To see this, assume that the material's temperature evolves according to

$$x_{k+1} = (1-a)x_k + au_k + w_k, \qquad k = 0, 1,$$

where w_0 and w_1 are independent random variables with given distribution, zero mean

$$E\{w_0\} = E\{w_1\} = 0,$$

and finite variance. Then the equation for J_1^* [cf. Eq. (1.4)] becomes

$$\begin{aligned} J_1^*(x_1) &= \min_{u_1} \mathop{E}_{w_1} \Big\{ u_1^2 + r\big((1-a)x_1 + au_1 + w_1 - T\big)^2 \Big\} \\ &= \min_{u_1} \Big[u_1^2 + r\big((1-a)x_1 + au_1 - T\big)^2 \\ &\qquad + 2rE\{w_1\}\big((1-a)x_1 + au_1 - T\big) + rE\{w_1^2\} \Big]. \end{aligned}$$

Since $E\{w_1\} = 0$, we obtain

$$J_1^*(x_1) = \min_{u_1} \Big[u_1^2 + r\big((1-a)x_1 + au_1 - T\big)^2 \Big] + rE\{w_1^2\}.$$

Comparing this equation with Eq. (1.19), we see that the presence of w_1 has resulted in an additional inconsequential constant term, $rE\{w_1^2\}$. Therefore, the optimal policy for the last stage remains unaffected by the presence of w_1, while $J_1^*(x_1)$ is increased by $rE\{w_1^2\}$. It can be seen that a similar situation also holds for the first stage. In particular, the optimal cost is given by the same expression as before except for an additive constant that depends on $E\{w_0^2\}$ and $E\{w_1^2\}$.

Generally, if the optimal policy is unaffected when the disturbances are replaced by their means, we say that *certainty equivalence* holds. This occurs in several types of problems involving a linear system and a quadratic cost; see [Ber17], Sections 3.1 and 4.2. For other problems, certainty equivalence can be used as a basis for problem approximation, e.g., assume that certainty equivalence holds (i.e., replace stochastic quantities by some typical values, such as their expected values) and apply exact DP to the resulting deterministic optimal control problem (see Section 2.3.2).

1.3.8 Systems with Unknown Parameters - Adaptive Control

We have been dealing so far with systems having a known system equation. In practice, however, there are many cases where the system parameters are either not known exactly or change over time.

As an example consider a car cruise control system. The car's velocity at time k, denoted by x_k, evolves according to

$$x_{k+1} = x_k + bu_k,$$

where u_k is the force that propels the car forward (u_k can be related to the pressure applied on the car's gas pedal). However, the coefficient b changes frequently and cannot be modeled with any precision because it depends on unpredictable time-varying conditions, such as the slope and condition of the road, and the weight of the car (which is affected by the number of passengers). This points to the need of controllers that yield satisfactory performance over a potentially broad range of system parameters.

To construct a formal optimization framework for dealing with such a situation, we may embed the problem within an imperfect state information framework by modeling the unknown parameters as unobservable states. Indeed, let the system equation be of the form

$$x_{k+1} = f_k(x_k, \theta, u_k, w_k),$$

where θ is a vector of unknown parameters, which for simplicity we assume to be fixed over time. We introduce an additional state variable $y_k = \theta$ and obtain a system equation of the form

$$\begin{pmatrix} x_{k+1} \\ y_{k+1} \end{pmatrix} = \begin{pmatrix} f_k(x_k, y_k, u_k, w_k) \\ y_k \end{pmatrix}.$$

This equation can be written compactly as

$$\tilde{x}_{k+1} = \tilde{f}_k(\tilde{x}_k, u_k, w_k),$$

where $\tilde{x}_k = (x_k, y_k)$ is the new state, and $\tilde{f}_k$ is an appropriate function. The initial state is

$$\tilde{x}_0 = (x_0, \theta).$$

Unfortunately, however, since y_k (i.e., θ) is unobservable, the problem is one of partial state information even if the controller knows the state x_k exactly. This makes the exact solution by DP intractable. To address this situation, several suboptimal solution approaches have been suggested.

An apparently reasonable form of suboptimal control is to separate the control process into two phases, a *parameter estimation (or identification) phase* and a *control phase*. In the first phase the unknown parameters are identified, while the control takes no account of the interim results of estimation. The final parameter estimates from the first phase are then used to implement an optimal control law in the second phase. This alternation of estimation and control phases may be repeated several times during any system run in order to take into account subsequent changes of the parameters.

One drawback of this approach is that it is not always easy to determine when to terminate one phase and start the other. A second difficulty, of a more fundamental nature, is that the unknown parameters must often be estimated as the system is being controlled. Then, unfortunately, the control process may make some of the unknown parameters invisible to the estimation process. This is known as the problem of parameter *identifiability*, which is discussed in the context of optimal control in several sources, including [BoV79] and [Kum83]; see also [Ber17], Section 6.7. For a simple example, consider the scalar system

$$x_{k+1} = ax_k + bu_k, \qquad k = 0, \ldots, N-1,$$

and the quadratic cost

$$\sum_{k=1}^{N} (x_k)^2.$$

Assuming perfect state information, if the parameters a and b are known, it can be seen that the optimal control law is

$$\mu_k^*(x_k) = -\frac{a}{b} x_k,$$

which sets all future states to 0. Assume now that the parameters a and b are unknown, and consider the two-phase method. During the first phase the control law

$$\tilde{\mu}_k(x_k) = \gamma x_k \tag{1.22}$$

is used (γ is some scalar; for example, $\gamma = -\overline{a}/\overline{b}$, where $\overline{a}$ and $\overline{b}$ are some a priori estimates of a and b, respectively). At the end of the first phase, the control law is changed to

$$\overline{\mu}_k(x_k) = -\frac{\hat{a}}{\hat{b}} x_k,$$

where $\hat{a}$ and $\hat{b}$ are the estimates obtained from the estimation process. However, with the control law (1.22), the closed-loop system is

$$x_{k+1} = (a + b\gamma)x_k,$$

so the estimation process can at best yield the value of $(a + b\gamma)$ but not the values of both a and b. In other words, the estimation process cannot discriminate between pairs of values (a_1, b_1) and (a_2, b_2) such that $a_1 + b_1\gamma = a_2 + b_2\gamma$. Therefore, a and b are not identifiable when feedback control of the form (1.22) is applied.

The issues discussed above and the methods to address them are part of a broad field known as *adaptive control*, which deals with the design of controllers for systems with unknown parameters. This is a rich subject

with many and diverse applications. We will not discuss adaptive control in this book, and we refer to textbook treatments, such as Aström and Wittenmark [AsW94], Goodwin and Sin [GoS84], Ioannou and Sun [IoS96], Krstic, Kanellakopoulos, and Kokotovic [KKK95], Kumar and Varaiya [KuV86], Sastry and Bodson [SaB11], and Slotine and Li [SlL91].

We note, however, a simple and popular methodology, *PID (Proportional-Integral-Derivative) control*, which can be used for problems involving an unknown or changing mathematical model; see e.g., the books by Aström and Hagglund [AsH95], [AsH06]. In particular, PID control aims to maintain the output of a single-input single-output dynamic system around a set point or to follow a given trajectory, as the system parameters change within a relatively broad range. In its simplest form, the PID controller is parametrized by three scalar parameters, which may be determined by a variety of methods, some of them manual/heuristic. We will later discuss briefly PID control in Section 5.7, and point out that the automatic choice of its parameters can be considered within the context of the broader methodology of approximation in policy space.

1.4 REINFORCEMENT LEARNING AND OPTIMAL CONTROL - SOME TERMINOLOGY

There has been intense interest in DP-related approximations in view of their promise to deal with the *curse of dimensionality* (the explosion of the computation as the number of states increases is dealt with the use of approximate cost functions) and the *curse of modeling* (a simulator/computer model may be used in place of a mathematical model of the problem). The current state of the subject owes much to an enormously beneficial cross-fertilization of ideas from optimal control (with its traditional emphasis on sequential decision making and formal optimization methodologies), and from artificial intelligence (and its traditional emphasis on learning through observation and experience, heuristic evaluation functions in game-playing programs, and the use of feature-based and other representations).

The boundaries between these two fields are now diminished thanks to a deeper understanding of the foundational issues, and the associated methods and core applications. Unfortunately, however, there have been substantial differences in language and emphasis in RL-based discussions (where artificial intelligence-related terminology is used) and DP-based discussions (where optimal control-related terminology is used). This includes the typical use of maximization/value function/reward in the former field and the use of minimization/cost function/cost per stage in the latter field, and goes much further.

The terminology used in this book is standard in DP and optimal control, and in an effort to forestall confusion of readers that are accustomed to either the RL or the optimal control terminology, we provide a list of terms commonly used in RL, and their optimal control counterparts.

(a) **Environment** = System.

(b) **Agent** = Decision maker or controller.

(c) **Action** = Decision or control.

(d) **Reward of a stage** = (Opposite of) Cost of a stage.

(e) **State value** = (Opposite of) Cost starting from a state.

(f) **Value (or reward) function** = (Opposite of) Cost function.

(g) **Maximizing the value function** = Minimizing the cost function.

(h) **Action (or state-action) value** = Q-factor (or Q-value) of a state-control pair. (Q-value is also used often in RL.)

(i) **Planning** = Solving a DP problem with a known mathematical model.

(j) **Learning** = Solving a DP problem without using an explicit mathematical model. (This is the principal meaning of the term "learning" in RL. Other meanings are also common.)

(k) **Self-learning** (or self-play in the context of games) = Solving a DP problem using some form of policy iteration.

(l) **Deep reinforcement learning** = Approximate DP using value and/or policy approximation with deep neural networks.

(m) **Prediction** = Policy evaluation.

(n) **Generalized policy iteration** = Optimistic policy iteration.

(o) **State abstraction** = State aggregation.

(p) **Temporal abstraction** = Time aggregation.

(q) **Learning a model** = System identification.

(r) **Episodic task or episode** = Finite-step system trajectory.

(s) **Continuing task** = Infinite-step system trajectory.

(t) **Experience replay** = Reuse of samples in a simulation process.

(u) **Bellman operator** = DP mapping or operator.

(v) **Backup** = Applying the DP operator at some state.

(w) **Sweep** = Applying the DP operator at all states.

(x) **Greedy policy with respect to a cost function** J = Minimizing policy in the DP expression defined by J.

(y) **Afterstate** = Post-decision state.

(z) **Ground truth** = Empirical evidence or information provided by direct observation.

Some of the preceding terms will be introduced in future chapters. The reader may then wish to return to this section as an aid in connecting with the relevant RL literature.

Notation

Unfortunately the confusion arising from different terminology has been exacerbated by the use of different notations. The present textbook roughly follows the "standard" notation of the Bellman/Pontryagin optimal control era; see e.g., the classical books by Athans and Falb [AtF66], Bellman [Bel67], and Bryson and Ho [BrH75]. This notation is consistent with the author's other DP books.

A summary of our most prominently used symbols is as follows:

(a) x: state.

(b) u: control.

(c) J: cost function.

(d) g: cost per stage.

(e) f: system function.

(f) i: discrete state.

(g) $p_{ij}(u)$: transition probability from state i to state j under control u.

(h) α: discount factor in discounted problems.

The x-u-J notation is standard in optimal control textbooks (e.g., the books by Athans and Falb [AtF66], and Bryson and Ho [BrH75], as well as the more recent book by Liberzon [Lib11]). The notations f and g are also used most commonly in the literature of the early optimal control period as well as later (unfortunately the more natural symbol "c" has not been used much in place of "g" for the cost per stage). The discrete system notations i and $p_{ij}(u)$ are very common in the discrete-state Markov decision problem and operations research literature, where discrete-state problems have been treated extensively [sometimes the alternative notation $p(j \mid i, u)$ is used for the transition probabilities].

The RL literature addresses for the most part finite-state Markov decision problems, most frequently the discounted and stochastic shortest path infinite horizon problems that are discussed in Chapter 4. The most commonly used notation is s for state, a for action, $r(s, a, s')$ for reward per stage, $p(s' \mid s, a)$ or $P_{s,a}(s')$ for transition probability from s to s' under action a, and γ for discount factor (see e.g., Sutton and Barto [SuB18]).

1.5 NOTES AND SOURCES

Our discussion of exact DP in this chapter has been brief since our focus in this book will be on approximate DP and RL. The author's DP text-

book [Ber17] provides an extensive discussion of finite horizon exact DP, and its applications to discrete and continuous spaces problems, using a notation and style that is consistent with the present book. The books by Puterman [Put94] and by the author [Ber12] provide detailed treatments of infinite horizon finite-state Markovian decision problems. Continuous spaces infinite horizon problems are covered in the author's book [Ber12], while some of the more complex mathematical aspects of exact DP are discussed in the monograph by Bertsekas and Shreve [BeS78] (particularly the probabilistic/measure-theoretic issues associated with stochastic optimal control).

The author's abstract DP monograph [Ber18a] aims at a unified development of the core theory and algorithms of total cost sequential decision problems, and addresses simultaneously stochastic, minimax, game, risk-sensitive, and other DP problems, through the use of the abstract DP operator (or Bellman operator as it is often called in RL). The idea here is to gain insight through abstraction. In particular, the structure of a DP model is encoded in its abstract Bellman operator, which serves as the "mathematical signature" of the model. Thus, characteristics of this operator (such as monotonicity and contraction) largely determine the analytical results and computational algorithms that can be applied to that model. It is likely that some of the approximation algorithms of the present book are transferable to a broad variety of DP models, well beyond the ones studied here. The design and analysis of these algorithms can then benefit from the broad algorithmic principles that have been developed through an abstract viewpoint.

The approximate DP and RL literature has expanded tremendously since the connections between DP and RL became apparent in the late 80s and early 90s. We restrict ourselves to mentioning textbooks, research monographs, and broad surveys, which supplement our discussions, express related viewpoints, and collectively provide a guide to the literature. Moreover, inevitably our referencing reflects a cultural bias, and an overemphasis on sources that are familiar to the author and are written in a similar style to the present book (including the author's own works). Thus we wish to apologize in advance for the many omissions of important research references that are somewhat outside our own understanding and view of the field.

Two books were written on our subject in the 1990s, setting the tone for subsequent developments in the field. One in 1996 by Bertsekas and Tsitsiklis [BeT96], which reflects a decision, control, and optimization viewpoint, and another in 1998 by Sutton and Barto, which reflects an artificial intelligence viewpoint (a 2nd edition, [SuB18], was published in 2018). We refer to the former book and also to the author's DP textbooks [Ber12], [Ber17] for a broader discussion of some of the topics of the present book, including algorithmic convergence issues and additional DP models, such as those based on average cost optimization. For historical accounts of

the early development of the subject, see [BeT96], Section 6.7, and [SuB18], Section 1.7.

More recent books are by Gosavi [Gos15] (a much expanded 2nd edition of his 2003 monograph), which emphasizes simulation-based optimization and RL algorithms, Cao [Cao07], which focuses on a sensitivity approach to simulation-based methods, Chang, Fu, Hu, and Marcus [CFH13] (a 2nd edition of their 2007 monograph), which emphasizes finite-horizon/limited lookahead schemes and adaptive sampling, Busoniu et al. [BBD10], which focuses on function approximation methods for continuous space systems and includes a discussion of random search methods, Powell [Pow11], which emphasizes resource allocation and operations research applications, Vrabie, Vamvoudakis, and Lewis [VVL13], which discusses neural network-based methods, on-line adaptive control methods, and continuous-time optimal control applications, Kochenderfer et al. [KAC15], which selectively discusses applications and approximations in DP and the treatment of uncertainty, Jiang and Jiang [JiJ17], which develops adaptive control within an approximate DP framework, and Liu et al. [LWW17], which deals with forms of adaptive dynamic programming, and topics in both RL and optimal control. The book by Krishnamurthy [Kri16] focuses on partial state information problems, with discussion of both exact DP, and approximate DP/RL methods. The book by Haykin [Hay08] discusses approximate DP in the broader context of neural network-related subjects. The book by Borkar [Bor08] is an advanced monograph that addresses rigorously many of the convergence issues of iterative stochastic algorithms in approximate DP, mainly using the so called ODE approach. The book by Meyn [Mey07] is broader in its coverage, but touches upon some of the approximate DP algorithms that we discuss.

Influential early surveys were written, from an artificial intelligence viewpoint, by Barto, Bradtke, and Singh [BBS95] (which dealt with the methodologies of real-time DP and its antecedent, real-time heuristic search [Kor90], and the use of asynchronous DP ideas [Ber82], [Ber83], [BeT89] within their context), and by Kaelbling, Littman, and Moore [KLM96] (which focused on general principles of RL). The volume by White and Sofge [WhS92] also contains several surveys describing early work in the field.

Several overview papers in the volume by Si, Barto, Powell, and Wunsch [SBP04] describe some approximation methods that we will not be covering in much detail in this book: linear programming approaches (De Farias [DeF04]), large-scale resource allocation methods (Powell and Van Roy [PoV04]), and deterministic optimal control approaches (Ferrari and Stengel [FeS04], and Si, Yang, and Liu [SYL04]). Updated accounts of these and other related topics are given in the survey collections by Lewis, Liu, and Lendaris [LLL08], and Lewis and Liu [LeL13].

Recent extended surveys and short monographs are Borkar [Bor09] (a methodological point of view that explores connections with other Monte

Carlo schemes), Lewis and Vrabie [LeV09] (a control theory point of view), Szepesvari [Sze10] (which discusses approximation in value space from a RL point of view), Deisenroth, Neumann, and Peters [DNP11], and Grondman et al. [GBL12] (which focus on policy gradient methods), Browne et al. [BPW12] (which focuses on Monte Carlo Tree Search), Mausam and Kolobov [MaK12] (which deals with Markovian decision problems from an artificial intelligence viewpoint), Schmidhuber [Sch15], Arulkumaran et al. [ADB17], Li [Li17], Busoniu et al. [BDT18], and Caterini and Chang [CaC18] (which deal with reinforcement learning schemes that are based on the use of deep neural networks), the author's [Ber05a] (which focuses on rollout algorithms and model predictive control), [Ber11a] (which focuses on approximate policy iteration), and [Ber18b] (which focuses on aggregation methods), and Recht [Rec18a] (which focuses on continuous spaces optimal control).

References

[ACF02] Auer, P., Cesa-Bianchi, N., and Fischer, P., 2002. "Finite Time Analysis of the Multiarmed Bandit Problem," Machine Learning, Vol. 47, pp. 235-256.

[ACV09] Argall, B. D., Chernova, S., Veloso, M., and Browning, B., 2009. "A Survey of Robot Learning from Demonstration," Robotics and Autonomous Systems, Vol. 57, pp. 469-483.

[ADB17] Arulkumaran, K., Deisenroth, M. P., Brundage, M., and Bharath, A. A., 2017. "A Brief Survey of Deep Reinforcement Learning," arXiv preprint arXiv:1708.05866.

[ALZ08] Asmuth, J., Littman, M. L., and Zinkov, R., 2008. "Potential-Based Shaping in Model-Based Reinforcement Learning," Proc. of 23rd AAAI Conference, pp. 604-609.

[AMS07] Antos, A., Munos, R., and Szepesvari, C., 2007. "Fitted Q-Iteration in Continuous Action-Space MDPs," Proc. of NIPS, pp. 9-16.

[AMS09] Audibert, J.Y., Munos, R., and Szepesvari, C., 2009. "Exploration-Exploitation Tradeoff Using Variance Estimates in Multi-Armed Bandits," Theoretical Computer Science, Vol. 410, pp. 1876-1902.

[ASS68] Aleksandrov, V. M., Sysoyev, V. I, and Shemeneva, V. V., 1968. "Stochastic Optimization of Systems," Engineering Cybernetics, Vol. 5, pp.11-16.

[AbN04] Abbeel, P., and Ng, A. Y., 2004. "Apprenticeship Learning via Inverse Reinforcement Learning," in Proceedings of the 21st ICML.

[Abr90] Abramson, B., 1990. "Expected-Outcome: A General Model of Static Evaluation," IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 12, pp. 182-193.

[Agr95] Agrawal, R., 1995. "Sample Mean Based Index Policies with $O(\log n)$ Regret for the Multiarmed Bandit Problem," Advances in Applied Probability, Vol. 27, pp. 1054-1078.

[AnH14] Antunes, D., and Heemels, W.P.M.H., 2014. "Rollout Event-Triggered Control: Beyond Periodic Control Performance," IEEE Transactions on Automatic Control, Vol. 59, pp. 3296-3311.

[AnM79] Anderson, B. D. O., and Moore, J. B., 1979. Optimal Filtering, Prentice-Hall, Englewood Cliffs, N. J.

[ArK03] Ariyur, K. B., and Krstic, M., 2003. Real-Time Optimization by Extremum-Seeking Control, John Wiley and Sons, N. Y.

[AsG10] Asmussen, S., and Glynn, P. W., 2010. Stochastic Simulation: Algorithms and Analysis, Springer, N. Y.

[AsH95] Aström, K. J., and Hagglund, T., 1995. PID Controllers: Theory, Design, and Tuning, Instrument Society of America, Research Triangle Park, N. C.

[AsH06] Aström, K. J., and Hagglund, T., 2006. Advanced PID Control, Instrument Society of America, Research Triangle Park, N. C.

[AsW94] Aström, K. J., and Wittenmark, B., 1994. Adaptive Control, 2nd Edition, Prentice-Hall, Englewood Cliffs, N. J.

[AtF66] Athans, M., and Falb, P., 1966. Optimal Control, McGraw-Hill, N. Y.

[BBD10] Busoniu, L., Babuska, R., De Schutter, B., and Ernst, D., 2010. Reinforcement Learning and Dynamic Programming Using Function Approximators, CRC Press, N. Y.

[BBG13] Bertazzi, L., Bosco, A., Guerriero, F., and Lagana, D., 2013. "A Stochastic Inventory Routing Problem with Stock-Out," Transportation Research, Part C, Vol. 27, pp. 89-107.

[BBM17] Borelli, F., Bemporad, A., and Morari, M., 2017. Predictive Control for Linear and Hybrid Systems, Cambridge Univ. Press, Cambridge, UK.

[BBN04] Bertsekas, D. P., Borkar, V., and Nedić, A., 2004. "Improved Temporal Difference Methods with Linear Function Approximation," in Learning and Approximate Dynamic Programming, by J. Si, A. Barto, W. Powell, and D. Wunsch, (Eds.), IEEE Press, N. Y.

[BBP13] Bhatnagar, S., Borkar, V. S., and Prashanth, L. A., 2013. "Adaptive Feature Pursuit: Online Adaptation of Features in Reinforcement Learning," in *Reinforcement Learning and Approximate Dynamic Programming for Feedback Control*, by F. Lewis and D. Liu (eds.), IEEE Press, Piscataway, N. J., pp. 517-534.

[BBS87] Bean, J. C., Birge, J. R., and Smith, R. L., 1987. "Aggregation in Dynamic Programming," Operations Research, Vol. 35, pp. 215-220.

[BBS95] Barto, A. G., Bradtke, S. J., and Singh, S. P., 1995. "Real-Time Learning and Control Using Asynchronous Dynamic Programming," Artificial Intelligence, Vol. 72, pp. 81-138.

[BCN18] Bottou, L., Curtis, F. E., and Nocedal, J., 2018. "Optimization Methods for Large-Scale Machine Learning," SIAM Review, Vol. 60, pp. 223-311.

[BDT18] Busoniu, L., de Bruin, T., Tolic, D., Kober, J., and Palunko, I., 2018. "Reinforcement Learning for Control: Performance, Stability, and Deep Approximators," Annual Reviews in Control, Vol. 46, pp. 8-28.

[BGM95] Bertsekas, D. P., Guerriero, F., and Musmanno, R., 1995. "Parallel Shortest Path Methods for Globally Optimal Trajectories," High Performance Computing: Technology, Methods, and Applications, (J. Dongarra et al., Eds.), Elsevier.

[BKM05] de Boer, P. T., Kroese, D. P., Mannor, S., and Rubinstein, R. Y. 2005. "A Tutorial on the Cross-Entropy Method," Annals of Operations Research, Vol. 134, pp. 19-67.

[BLL19] Bartlett, P. L., Long, P. M., Lugosi, G., and Tsigler, A., 2019. "Benign Overfitting in Linear Regression," arXiv preprint arXiv:1906.11300.

[BMM18] Belkin, M., Ma, S., and Mandal, S., 2018. "To Understand Deep Learning we Need to Understand Kernel Learning," arXiv preprint arXiv:1802.01396.

[BNO03] Bertsekas, D. P., Nedić, A., and Ozdaglar, A. E., 2003. Convex Analysis and Optimization, Athena Scientific, Belmont, MA.

[BPP13] Bhatnagar, S., Prasad, H., and Prashanth, L. A., 2013. Stochastic Recursive Al-

gorithms for Optimization, Lecture Notes in Control and Information Sciences, Springer, N. Y.

[BPW12] Browne, C., Powley, E., Whitehouse, D., Lucas, L., Cowling, P. I., Rohlfshagen, P., Tavener, S., Perez, D., Samothrakis, S., and Colton, S., 2012. "A Survey of Monte Carlo Tree Search Methods," IEEE Trans. on Computational Intelligence and AI in Games, Vol. 4, pp. 1-43.

[BRT18] Belkin, M., Rakhlin, A., and Tsybakov, A. B., 2018. "Does Data Interpolation Contradict Statistical Optimality?," arXiv preprint arXiv:1806.09471.

[BSA83] Barto, A. G., Sutton, R. S., and Anderson, C. W., 1983. "Neuronlike Elements that Can Solve Difficult Learning Control Problems," IEEE Trans. on Systems, Man, and Cybernetics, Vol. 13, pp. 835-846.

[BTW97] Bertsekas, D. P., Tsitsiklis, J. N., and Wu, C., 1997. "Rollout Algorithms for Combinatorial Optimization," Heuristics, Vol. 3, pp. 245-262.

[BVE13] Ben Amor, H, Vogt, D., Ewerton, M., Berger, E., Jung, B., and Peters, J., 2013. "Learning Responsive Robot Behavior by Imitation," in 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 3257-3264.

[BWL19] Beuchat, P.N., Warrington, J., and Lygeros, J., 2019. "Accelerated Point-Wise Maximum Approach to Approximate Dynamic Programming," arXiv preprint arXiv:1901.03619.

[BYB94] Bradtke, S. J., Ydstie, B. E., and Barto, A. G., 1994. "Adaptive Linear Quadratic Control Using Policy Iteration," Proc. IEEE American Control Conference, Vol. 3, pp. 3475-3479.

[BaB01] Baxter, J., and Bartlett, P. L., 2001. "Infinite-Horizon Policy-Gradient Estimation," Journal of Artificial Intelligence Research, Vol. 15, pp. 319-350.

[Bac96] Back, T., 1996. Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms, Oxford University Press.

[Bai93] Baird, L. C., 1993. "Advantage Updating," Report WL-TR-93-1146, Wright Patterson AFB, OH.

[Bai94] Baird, L. C., 1994. "Reinforcement Learning in Continuous Time: Advantage Updating," International Conf. on Neural Networks, Orlando, Fla.

[BeC89] Bertsekas, D. P., and Castanon, D. A., 1989. "Adaptive Aggregation Methods for Infinite Horizon Dynamic Programming," IEEE Trans. on Aut. Control, Vol. AC-34, pp. 589-598.

[BeC99] Bertsekas, D. P., and Castanon, D. A., 1999. "Rollout Algorithms for Stochastic Scheduling Problems," Heuristics, Vol. 5, pp. 89-108.

[BeC08] Besse, C., and Chaib-draa, B., 2008. "Parallel Rollout for Online Solution of DEC-POMDPs," Proc. of 21st International FLAIRS Conference, pp. 619-624.

[BeL14] Beyme, S., and Leung, C., 2014. "Rollout Algorithm for Target Search in a Wireless Sensor Network," 80th Vehicular Technology Conference (VTC2014), IEEE, pp. 1-5.

[BeI96] Bertsekas, D. P., and Ioffe, S., 1996. "Temporal Differences-Based Policy Iteration and Applications in Neuro-Dynamic Programming," Lab. for Info. and Decision Systems Report LIDS-P-2349, Massachusetts Institute of Technology.

[BeP03] Bertsimas, D., and Popescu, I., 2003. "Revenue Management in a Dynamic Network Environment," Transportation Science, Vol. 37, pp. 257-277.

[BeR71] Bertsekas, D. P., and Rhodes, I. B., 1971. "On the Minimax Reachability of Target Sets and Target Tubes," Automatica, Vol. 7, pp. 233-247.

[BeR73] Bertsekas, D. P., and Rhodes, I. B., 1973. "Sufficiently Informative Functions and the Minimax Feedback Control of Uncertain Dynamic Systems," IEEE Trans. Automatic Control, Vol. AC-18, pp. 117-124.

[BeS78] Bertsekas, D. P., and Shreve, S. E., 1978. Stochastic Optimal Control: The Discrete Time Case, Academic Press, N. Y.; republished by Athena Scientific, Belmont, MA, 1996 (can be downloaded from the author's website).

[BeS18] Bertazzi, L., and Secomandi, N., 2018. "Faster Rollout Search for the Vehicle Routing Problem with Stochastic Demands and Restocking," European J. of Operational Research, Vol. 270, pp.487-497.

[BeT89] Bertsekas, D. P., and Tsitsiklis, J. N., 1989. Parallel and Distributed Computation: Numerical Methods, Prentice-Hall, Englewood Cliffs, N. J.; republished by Athena Scientific, Belmont, MA, 1997 (can be downloaded from the author's website).

[BeT91] Bertsekas, D. P., and Tsitsiklis, J. N., 1991. "An Analysis of Stochastic Shortest Path Problems," Math. Operations Res., Vol. 16, pp. 580-595.

[BeT96] Bertsekas, D. P., and Tsitsiklis, J. N., 1996. Neuro-Dynamic Programming, Athena Scientific, Belmont, MA.

[BeT97] Bertsimas, D., and Tsitsiklis, J. N., 1997. Introduction to Linear Optimization, Athena Scientific, Belmont, MA.

[BeT00] Bertsekas, D. P., and Tsitsiklis, J. N., 2000. "Gradient Convergence of Gradient Methods with Errors," SIAM J. on Optimization, Vol. 36, pp. 627-642.

[BeT08] Bertsekas, D. P., and Tsitsiklis, J. N., 2008. Introduction to Probability, 2nd Edition, Athena Scientific, Belmont, MA.

[BeY09] Bertsekas, D. P., and Yu, H., 2009. "Projected Equation Methods for Approximate Solution of Large Linear Systems," J. of Computational and Applied Mathematics, Vol. 227, pp. 27-50.

[BeY10] Bertsekas, D. P., and Yu, H., 2010. "Asynchronous Distributed Policy Iteration in Dynamic Programming," Proc. of Allerton Conf. on Communication, Control and Computing, Allerton Park, Ill, pp. 1368-1374.

[BeY12] Bertsekas, D. P., and Yu, H., 2012. "Q-Learning and Enhanced Policy Iteration in Discounted Dynamic Programming," Math. of Operations Research, Vol. 37, pp. 66-94.

[BeY16] Bertsekas, D. P., and Yu, H., 2016. "Stochastic Shortest Path Problems Under Weak Conditions," Lab. for Information and Decision Systems Report LIDS-2909, MIT.

[Bel57] Bellman, R., 1957. Dynamic Programming, Princeton University Press, Princeton, N. J.

[Bel67] Bellman, R., 1967. Introduction to the Mathematical Theory of Control Processes, Academic Press, Vols. I and II, New York, N. Y.

[Bel84] Bellman, R., 1984. Eye of the Hurricane, World Scientific Publishing, Singapore.

[Ben09] Bengio, Y., 2009. "Learning Deep Architectures for AI," Foundations and Trends in Machine Learning, Vol. 2, pp. 1-127.

[Ber71] Bertsekas, D. P., 1971. "Control of Uncertain Systems With a Set-Membership Description of the Uncertainty," Ph.D. Dissertation, Massachusetts Institute of Technology, Cambridge, MA (can be downloaded from the author's website).

[Ber72] Bertsekas, D. P., 1972. "Infinite Time Reachability of State Space Regions by Using Feedback Control," IEEE Trans. Automatic Control, Vol. AC-17, pp. 604-613.

[Ber73] Bertsekas, D. P., 1973. "Stochastic Optimization Problems with Nondifferentiable Cost Functionals," J. of Optimization Theory and Applications, Vol. 12, pp. 218-231.

[Ber82] Bertsekas, D. P., 1982. "Distributed Dynamic Programming," IEEE Trans. Automatic Control, Vol. AC-27, pp. 610-616.

[Ber83] Bertsekas, D. P., 1983. "Asynchronous Distributed Computation of Fixed Points," Math. Programming, Vol. 27, pp. 107-120.

[Ber91] Bertsekas, D. P., 1991. Linear Network Optimization: Algorithms and Codes, M.I.T. Press, Cambridge, MA (can be downloaded from the author's website).

[Ber95] Bertsekas, D. P., 1995. "A Counterexample to Temporal Differences Learning," Neural Computation, Vol. 7, pp. 270-279.

[Ber96a] Bertsekas, D. P., 1996. "Incremental Least Squares Methods and the Extended Kalman Filter," SIAM J. on Optimization, Vol. 6, pp. 807-822.

[Ber96b] Bertsekas, D. P., 1996. Lecture at NSF Workshop on Reinforcement Learning, Hilltop House, Harper's Ferry, N. Y.

[Ber97a] Bertsekas, D. P., 1997. "A New Class of Incremental Gradient Methods for Least Squares Problems," SIAM J. on Optimization, Vol. 7, pp. 913-926.

[Ber97b] Bertsekas, D. P., 1997. "Differential Training of Rollout Policies," Proc. of the 35th Allerton Conference on Communication, Control, and Computing, Allerton Park, Ill.

[Ber98] Bertsekas, D. P., 1998. Network Optimization: Continuous and Discrete Models, Athena Scientific, Belmont, MA (can be downloaded from the author's website).

[Ber05a] Bertsekas, D. P., 2005. "Dynamic Programming and Suboptimal Control: A Survey from ADP to MPC," European J. of Control, Vol. 11, pp. 310-334.

[Ber05b] Bertsekas, D. P., 2005. "Rollout Algorithms for Constrained Dynamic Programming," LIDS Report 2646, MIT.

[Ber07] Bertsekas, D. P., 2007. "Separable Dynamic Programming and Approximate Decomposition Methods," IEEE Trans. on Aut. Control, Vol. 52, pp. 911-916.

[Ber10] Bertsekas, D. P., 2010. "Incremental Gradient, Subgradient, and Proximal Methods for Convex Optimization: A Survey," Lab. for Information and Decision Systems Report LIDS-P-2848, MIT; a condensed version with the same title appears in Optimization for Machine Learning, by S. Sra, S. Nowozin, and S. J. Wright, (eds.), MIT Press, Cambridge, MA, 2012, pp. 85-119.

[Ber11a] Bertsekas, D. P., 2011. "Approximate Policy Iteration: A Survey and Some New Methods," J. of Control Theory and Applications, Vol. 9, pp. 310-335.

[Ber11b] Bertsekas, D. P., 2011. "Temporal Difference Methods for General Projected Equations," IEEE Trans. on Aut. Control, Vol. 56, pp. 2128-2139.

[Ber11c] Bertsekas, D. P., 2011. "Incremental Proximal Methods for Large Scale Convex Optimization," Math. Programming, Vol. 129, pp. 163-195.

[Ber12] Bertsekas, D. P., 2012. Dynamic Programming and Optimal Control, Vol. II, 4th Edition, Athena Scientific, Belmont, MA.

[Ber13a] Bertsekas, D. P., 2013. "Rollout Algorithms for Discrete Optimization: A Survey," Handbook of Combinatorial Optimization, Springer.

[Ber13b] Bertsekas, D. P., 2013. "λ-Policy Iteration: A Review and a New Implementation," in Reinforcement Learning and Approximate Dynamic Programming for Feedback Control, by F. Lewis and D. Liu (eds.), IEEE Press, Piscataway, N. J., pp. 381-409.

[Ber15a] Bertsekas, D. P., 2015. Convex Optimization Algorithms, Athena Scientific, Belmont, MA.

[Ber15b] Bertsekas, D. P., 2015. "Incremental Aggregated Proximal and Augmented Lagrangian Algorithms," Lab. for Information and Decision Systems Report LIDS-P-3176, MIT; arXiv preprint arXiv:1507.1365936.

[Ber16a] Bertsekas, D. P., 2016. Nonlinear Programming, 3rd Edition, Athena Scientific, Belmont, MA.

[Ber16b] Bertsekas, D. P., 2016. "Affine Monotonic and Risk-Sensitive Models in Dynamic Programming," arXiv preprint arXiv:1608.01393; IEEE Trans. on Automatic Control, Vol. 64, 2019, pp. 3117-3128.

[Ber16c] Bertsekas, D. P., 2016. "Proximal Algorithms and Temporal Differences for Large Linear Systems: Extrapolation, Approximation, and Simulation," arXiv preprint arXiv:1610.05427; Computational Optimization and Applications J., Vol. 70, 2018, pp. 709-736.

[Ber17] Bertsekas, D. P., 2017. Dynamic Programming and Optimal Control, Vol. I, 4th Edition, Athena Scientific, Belmont, MA.

[Ber18a] Bertsekas, D. P., 2018. Abstract Dynamic Programming, 2nd Edition, Athena Scientific, Belmont, MA (can be downloaded from the author's website).

[Ber18b] Bertsekas, D. P., 2018. "Feature-Based Aggregation and Deep Reinforcement Learning: A Survey and Some New Implementations," Lab. for Information and Decision Systems Report, MIT; arXiv preprint arXiv:1804.04577; IEEE/CAA Journal of Automatica Sinica, Vol. 6, 2019, pp. 1-31.

[Ber18c] Bertsekas, D. P., 2018. "Biased Aggregation, Rollout, and Enhanced Policy Improvement for Reinforcement Learning," Lab. for Information and Decision Systems Report, MIT; arXiv preprint arXiv:1910.02426.

[Ber18d] Bertsekas, D. P., 2018. "Proximal Algorithms and Temporal Difference Methods for Solving Fixed Point Problems," Computational Optim. Appl., Vol. 70, pp. 709-736.

[Ber18e] Bertsekas, D.P., 2018. "Proper Policies in Infinite-State Stochastic Shortest Path Problems," IEEE Trans. on Automatic Control, Vol. 63, pp. 3787-3792.

[Ber19a] Bertsekas, D. P., 2019. Lecture Slides and Videolectures on Reinforcement Learning and Optimal Control, ASU, at http://web.mit.edu/dimitrib/www/RLbook.html.

[Ber19b] Bertsekas, D. P., 2019. "Robust Shortest Path Planning and Semicontractive Dynamic Programming," Naval Research Logistics, Vol. 66, pp. 15-37.

[Ber19c] Bertsekas, D. P., 2019. "Multiagent Rollout Algorithms and Reinforcement Learning," arXiv preprint arXiv:1910.00120.

[Bet10] Bethke, B. M., 2010. Kernel-Based Approximate Dynamic Programming Using Bellman Residual Elimination, Ph.D. Thesis, MIT.

[Bia16] Bianchi, P., 2016. "Ergodic Convergence of a Stochastic Proximal Point Algorithm," SIAM J. on Optimization, Vol. 26, pp. 2235-2260.

[Bis95] Bishop, C. M, 1995. Neural Networks for Pattern Recognition, Oxford University Press, N. Y.

[Bis06] Bishop, C. M, 2006. Pattern Recognition and Machine Learning, Springer, N. Y.

[Bla99] Blanchini, F., 1999. "Set Invariance in Control – A Survey," Automatica, Vol. 35, pp. 1747-1768.

[BoV79] Borkar, V., and Varaiya, P. P., 1979. "Adaptive Control of Markov Chains, I: Finite Parameter Set," IEEE Trans. Automatic Control, Vol. AC-24, pp. 953-958.

[Bor08] Borkar, V. S., 2008. Stochastic Approximation: A Dynamical Systems Viewpoint, Cambridge Univ. Press.

[Bor09] Borkar, V. S., 2009. "Reinforcement Learning: A Bridge Between Numerical Methods and Monte Carlo," in World Scientific Review, Vol. 9, Ch. 4.

[Boy02] Boyan, J. A., 2002. "Technical Update: Least-Squares Temporal Difference Learning," Machine Learning, Vol. 49, pp. 1-15.

[BrB96] Bradtke, S. J., and Barto, A. G., 1996. "Linear Least-Squares Algorithms for Temporal Difference Learning," Machine Learning, Vol. 22, pp. 33-57.

[BrH75] Bryson, A., and Ho, Y. C., 1975. Applied Optimal Control: Optimization, Estimation, and Control, (revised edition), Taylor and Francis, Levittown, Penn.

[BuK97] Burnetas, A. N., and Katehakis, M. N., 1997. "Optimal Adaptive Policies for Markov Decision Processes," Math. of Operations Research, Vol. 22, pp. 222-255.

[CFH05] Chang, H. S., Hu, J., Fu, M. C., and Marcus, S. I., 2005. "An Adaptive Sampling Algorithm for Solving Markov Decision Processes," Operations Research, Vol. 53, pp. 126-139.

[CFH13] Chang, H. S., Hu, J., Fu, M. C., and Marcus, S. I., 2013. Simulation-Based Algorithms for Markov Decision Processes, 2nd Edition, Springer, N. Y.

[CFH16] Chang, H. S., Hu, J., Fu, M. C., and Marcus, S. I., 2016. "Google DeepMind's AlphaGo," ORMS Today, INFORMS, Vol. 43.

[CGC04] Chang, H. S., Givan, R. L., and Chong, E. K. P., 2004. "Parallel Rollout for Online Solution of Partially Observable Markov Decision Processes," Discrete Event Dynamic Systems, Vol. 14, pp. 309-341.

[CLT19] Chapman, M. P., Lacotte, J., Tamar, A., Lee, D., Smith, K. M., Cheng, V., Fisac, J. F., Jha, S., Pavone, M., and Tomlin, C. J., 2019. "A Risk-Sensitive Finite-Time Reachability Approach for Safety of Stochastic Dynamic Systems," arXiv preprint arXiv:1902.11277.

[CRV06] Cogill, R., Rotkowitz, M., Van Roy, B., and Lall, S., 2006. "An Approximate Dynamic Programming Approach to Decentralized Control of Stochastic Systems," in Control of Uncertain Systems: Modelling, Approximation, and Design, Springer, Berlin, pp. 243-256.

[CXL19] Chu, Z., Xu, Z., and Li, H., 2019. "New Heuristics for the RCPSP with Multiple Overlapping Modes," Computers and Industrial Engineering, Vol. 131, pp. 146-156.

[CaB04] Camacho, E. F., and Bordons, C., 2004. Model Predictive Control, 2nd Edition, Springer, New York, N. Y.

[CaC97] Cao, X. R., and Chen, H. F., 1997. "Perturbation Realization Potentials and Sensitivity Analysis of Markov Processes," IEEE Trans. on Aut. Control, Vol. 32, pp. 1382-1393.

[CaC18] Caterini, A. L., and Chang, D. E., 2018. Deep Neural Networks in a Mathematical Framework, Springer, Oxford, UK.

[CaW98] Cao, X. R., and Wan, Y. W., 1998. "Algorithms for Sensitivity Analysis of Markov Systems Through Potentials and Perturbation Realization," IEEE Trans. Control Systems Technology, Vol. 6, pp. 482-494.

[Can16] Candy, J. V., 2016. Bayesian Signal Processing: Classical, Modern, and Particle Filtering Methods, Wiley-IEEE Press.

[Cao07] Cao, X. R., 2007. Stochastic Learning and Optimization: A Sensitivity-Based Approach, Springer, N. Y.

[ChC17] Chui, C. K., and Chen, G., 2017. Kalman Filtering, Springer International Publishing.

[ChM82] Chatelin, F., and Miranker, W. L., 1982. "Acceleration by Aggregation of Successive Approximation Methods," Linear Algebra and its Applications, Vol. 43, pp. 17-47.

[ChS00] Christianini, N., and Shawe-Taylor, J., 2000. Support Vector Machines and Other Kernel-Based Learning Methods, Cambridge Univ. Press.

[ChT91] Chow, C.-S., and Tsitsiklis, J. N., 1991. "An Optimal One–Way Multigrid Algorithm for Discrete–Time Stochastic Control," IEEE Trans. on Aut. Control, Vol. AC-36, pp. 898-914.

[ChV12] Chacon, A., and Vladimirsky, A., 2012. "Fast Two-Scale Methods for Eikonal Equations," SIAM J. on Scientific Computing, Vol. 34, pp. A547-A578.

[ChV13] Chacon, A., and Vladimirsky, A., 2013. "A Parallel Heap-Cell Method for Eikonal Equations," arXiv preprint arXiv:1306.4743.

[ChV15] Chacon, A., and Vladimirsky, A., 2015. "A Parallel Two-Scale Method for Eikonal Equations," SIAM J. on Scientific Computing, Vol. 37, pp. A156-A180.

[CiS15] Ciosek, K., and Silver, D., 2015. "Value Iteration with Options and State Aggregation," Report, Centre for Computational Statistics and Machine Learning University College London.

[Cla17] Clawson, Z., 2017. Shortest path problems: Domain restriction, anytime planning, and multi-objective optimization. Ph.D. Thesis, Cornell University.

[Cou06] Coulom, R., 2006. "Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search," International Conference on Computers and Games, Springer, pp. 72-83.

[CrS00] Cristianini, N., and Shawe-Taylor, J., 2000. An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods, Cambridge Univ. Press.

[Cyb89] Cybenko, 1989. "Approximation by Superpositions of a Sigmoidal Function," Math. of Control, Signals, and Systems, Vol. 2, pp. 303-314.

[DDF19] Daubechies, I., DeVore, R., Foucart, S., Hanin, B., and Petrova, G., 2019. "Nonlinear Approximation and (Deep) ReLU Networks," arXiv preprint arXiv:1905.02199.

[D'Ep60] D'Epenoux, F., 1960. "Sur un Probleme de Production et de Stockage Dans l'Aleatoire," Rev. Francaise Aut. Infor. Recherche Operationnelle, Vol. 14, (English Transl.: Management Sci., Vol. 10, 1963, pp. 98-108).

[DFM12] Desai, V. V., Farias, V. F., and Moallemi, C. C., 2012. "Aproximate Dynamic Programming via a Smoothed Approximate Linear Program," Operations Research, Vol. 60, pp. 655-674.

[DFM13] Desai, V. V., Farias, V. F., and Moallemi, C. C., 2013. "Bounds for Markov Decision Processes," in Reinforcement Learning and Approximate Dynamic Programming for Feedback Control, by F. Lewis and D. Liu (eds.), IEEE Press, Piscataway, N. J., pp. 452-473.

[DFV03] de Farias, D. P., and Van Roy, B., 2003. "The Linear Programming Approach to Approximate Dynamic Programming," Operations Research, Vol. 51, pp. 850-865.

[DFV04] de Farias, D. P., and Van Roy, B., 2004. "On Constraint Sampling in the Linear Programming Approach to Approximate Dynamic Programming," Mathematics of Operations Research, Vol. 29, pp. 462-478.

[DHS12] Duda, R. O., Hart, P. E., and Stork, D. G., 2012. Pattern Classification, J. Wiley, N. Y.

[DJW12] Duchi, J., Jordan, M. I., Wainwright, M. J., and Wibisono, A., 2012. "Finite Sample Convergence Rate of Zero-Order Stochastic Optimization Methods," NIPS, pp. 1448-1456.

[DJW15] Duchi, J., Jordan, M. I., Wainwright, M. J., and Wibisono, A., 2015. "Optimal Rates for Zero-Order Convex Optimization: The Power of Two Function Evaluations," IEEE Trans. on Information Theory, Vol. 61, pp. 2788-2806.

[DNP11] Deisenroth, M. P., Neumann, G., and Peters, J., 2011. "A Survey on Policy Search for Robotics," Foundations and Trends in Robotics, Vol. 2, pp. 1-142.

[DNW16] David, O. E., Netanyahu, N. S., and Wolf, L., 2016. "Deepchess: End-to-End Deep Neural Network for Automatic Learning in Chess," in International Conference on Artificial Neural Networks, pp. 88-96.

[DeF04] De Farias, D. P., 2004. "The Linear Programming Approach to Approximate Dynamic Programming," in Learning and Approximate Dynamic Programming, by J. Si, A. Barto, W. Powell, and D. Wunsch, (Eds.), IEEE Press, N. Y.

[DeJ06] De Jong, K. A., 2006. Evolutionary Computation: A Unified Approach, MIT Press, Cambridge, MA.

[DeK11] Devlin, S., and Kudenko, D., 2011. "Theoretical Considerations of Potential-Based Reward Shaping for Multi-Agent Systems," in Proceedings of AAMAS.

[DeR79] Denardo, E. V., and Rothblum, U. G., 1979. "Optimal Stopping, Exponential Utility, and Linear Programming," Math. Programming, Vol. 16, pp. 228-244.

[DiL08] Dimitrakakis, C., and Lagoudakis, M. G., 2008. "Rollout Sampling Approximate Policy Iteration," Machine Learning, Vol. 72, pp. 157-171.

[DiM10] Di Castro, D., and Mannor, S., 2010. "Adaptive Bases for Reinforcement Learning," Machine Learning and Knowledge Discovery in Databases, Vol. 6321, pp. 312-327.

[DiW02] Dietterich, T. G., and Wang, X., 2002. "Batch Value Function Approximation via Support Vectors," in Advances in Neural Information Processing Systems, pp. 1491-1498.

[Die00] Dietterich, T., 2000. "Hierarchical Reinforcement Learning with the MAXQ Value Function Decomposition," J. of Artificial Intelligence Research, Vol. 13, pp. 227-303.

[DoD93] Douglas, C. C., and Douglas, J., 1993. "A Unified Convergence Theory for Abstract Multigrid or Multilevel Algorithms, Serial and Parallel," SIAM J. Num. Anal., Vol. 30, pp. 136-158.

[DoJ09] Doucet, A., and Johansen, A. M., 2009. "A Tutorial on Particle Filtering and

Smoothing: Fifteen Years Later," Handbook of Nonlinear Filtering, Oxford University Press, Vol. 12, p. 3.

[Dre02] Dreyfus, S. D., 2002. "Richard Bellman on the Birth of Dynamic Programming," Operations Research, Vol. 50, pp. 48-51.

[EGW06] Ernst, D., Geurts, P., and Wehenkel, L., 2006. "Tree-Based Batch Mode Reinforcement Learning," J. of Machine Learning Research, Vol. 6, pp. 503-556.

[ELP12] Estanjini, R. M., Li, K., and Paschalidis, I. C., 2012. "A Least Squares Temporal Difference Actor-Critic Algorithm with Applications to Warehouse Management," Naval Research Logistics, Vol. 59, pp. 197-211.

[EMM05] Engel, Y., Mannor, S., and Meir, R., 2005. "Reinforcement Learning with Gaussian Processes," in Proc. of the 22nd ICML, pp. 201-208.

[FHS14] Feinberg, E. A., Huang, J., and Scherrer, B., 2014. "Modified Policy Iteration Algorithms are not Strongly Polynomial for Discounted Dynamic Programming," Operations Research Letters, Vol. 42, pp. 429-431.

[FKB13] Frihauf, P., Krstic, M., and Basar, T., 2013. "Finite-Horizon LQ Control for Unknown Discrete-Time Linear Systems via Extremum Seeking," European Journal of Control, Vol. 19, pp. 399-407.

[FPB15] Farahmand, A. M., Precup, D., Barreto, A. M., and Ghavamzadeh, M., 2015. "Classification-Based Approximate Policy Iteration," IEEE Trans. on Automatic Control, Vol. 60, pp. 2989-2993.

[FYG06] Fern, A., Yoon, S., and Givan, R., 2006. "Approximate Policy Iteration with a Policy Language Bias: Solving Relational Markov Decision Processes," J. of Artificial Intelligence Research, Vol. 25, pp. 75-118.

[Fal87] Falcone, M., 1987. "A Numerical Approach to the Infinite Horizon Problem of Deterministic Control Theory," Appl. Math. Opt., Vol. 15, pp. 1-13.

[FeS04] Ferrari, S., and Stengel, R. F., 2004. "Model-Based Adaptive Critic Designs," in Learning and Approximate Dynamic Programming, by J. Si, A. Barto, W. Powell, and D. Wunsch, (Eds.), IEEE Press, N. Y.

[FeV02] Ferris, M. C., and Voelker, M. M., 2002. "Neuro-Dynamic Programming for Radiation Treatment Planning," Numerical Analysis Group Research Report NA-02/06, Oxford University Computing Laboratory, Oxford University.

[FeV04] Ferris, M. C., and Voelker, M. M., 2004. "Fractionation in Radiation Treatment Planning," Mathematical Programming B, Vol. 102, pp. 387-413.

[Fer10] Fernau, H., 2010. "Minimum Dominating Set of Queens: A Trivial Programming Exercise?" Discrete Applied Mathematics, Vol. 158, pp. 308-318.

[Fu17] Fu, M. C., 2017. "Markov Decision Processes, AlphaGo, and Monte Carlo Tree Search: Back to the Future," Leading Developments from INFORMS Communities, INFORMS, pp. 68-88.

[FuH94] Fu, M. C., and Hu, J.-Q., 1994. "Smoothed Perturbation Analysis Derivative Estimation for Markov Chains," Oper. Res. Letters, Vol. 41, pp. 241-251.

[Fun89] Funahashi, K., 1989. "On the Approximate Realization of Continuous Mappings by Neural Networks," Neural Networks, Vol. 2, pp. 183-192.

[GBB04] Greensmith, E., Bartlett, P. L., and Baxter, J., 2004. "Variance Reduction Techniques for Gradient Estimates in Reinforcement Learning," Journal of Machine Learning Research, Vol. 5, pp. 1471-1530.

[GBC16] Goodfellow, I., Bengio, J., and Courville, A., Deep Learning, MIT Press, Cambridge, MA.

[GBL12] Grondman, I., Busoniu, L., Lopes, G. A. D., and Babuska, R., 2012. "A Survey of Actor-Critic Reinforcement Learning: Standard and Natural Policy Gradients," IEEE Trans. on Systems, Man, and Cybernetics, Part C, Vol. 42, pp. 1291-1307.

[GDP18] Guerriero, F., Di Puglia Pugliese, L., and Macrina, G., 2018. "A Rollout Algorithm for the Resource Constrained Elementary Shortest Path Problem," Optimization Methods and Software, pp. 1-19.

[GGS13] Gabillon, V., Ghavamzadeh, M., and Scherrer, B., 2013. "Approximate Dynamic Programming Finally Performs Well in the Game of Tetris," in NIPS, pp. 1754-1762.

[GLG11] Gabillon, V., Lazaric, A., Ghavamzadeh, M., and Scherrer, B., 2011. "Classification-Based Policy Iteration with a Critic," in Proc. of ICML.

[GTO15] Goodson, J. C., Thomas, B. W., and Ohlmann, J. W., 2015. "Restocking-Based Rollout Policies for the Vehicle Routing Problem with Stochastic Demand and Duration Limits," Transportation Science, Vol. 50, pp. 591-607.

[Gla13] Glasserman, P., 2013. Monte Carlo Methods in Financial Engineering, Springer, N. Y.

[Gly87] Glynn, P. W., 1987. "Likelihood Ratio Gradient Estimation: An Overview," Proc. of the 1987 Winter Simulation Conference, pp. 366-375.

[Gly90] Glynn, P. W., 1990. "Likelihood Ratio Gradient Estimation for Stochastic Systems," Communications of the ACM, Vol. 33, pp. 75-84.

[GoR85] Gonzalez, R., and Rofman, E., 1985. "On Deterministic Control Problems: An Approximation Procedure for the Optimal Cost, Parts I, II," SIAM J. Control Optimization, Vol. 23, pp. 242-285.

[GoS84] Goodwin, G. C., and Sin, K. S. S., 1984. Adaptive Filtering, Prediction, and Control, Prentice-Hall, Englewood Cliffs, N. J.

[Gor95] Gordon, G. J., 1995. "Stable Function Approximation in Dynamic Programming," in Machine Learning: Proceedings of the Twelfth International Conference, Morgan Kaufmann, San Francisco, CA.

[Gos15] Gosavi, A., 2015. Simulation-Based Optimization: Parametric Optimization Techniques and Reinforcement Learning, 2nd Edition, Springer, N. Y.

[Gre05] Greensmith, E., 2005. Policy Gradient Methods: Variance Reduction and Stochastic Convergence, Ph.D. Thesis, The Australian National University.

[Grz17] Grzes, M., 2017. "Reward Shaping in Episodic Reinforcement Learning," in Proc. of the 16th Conference on Autonomous Agents and MultiAgent Systems, pp. 565-573.

[GuM03] Guerriero, F., and Mancini, M., 2003. "A Cooperative Parallel Rollout Algorithm for the Sequential Ordering Problem," Parallel Computing, Vol. 29, pp. 663-677.

[GuS18] Guillot, M., and Stauffer, G., 2018. The Stochastic Shortest Path Problem: A Polyhedral Combinatorics Perspective, European J. of Operational Research.

[HJG16] Huang, Q., Jia, Q. S., and Guan, X., 2016. "Robust Scheduling of EV Charging Load with Uncertain Wind Power Integration," IEEE Trans. on Smart Grid, Vol. 9, pp. 1043-1054.

[HLZ18] Hanawal, M. K., Liu, H., Zhu, H., and Paschalidis, I. C., 2018. "Learning Policies for Markov Decision Processes from Data," IEEE Trans. on Automatic Control.

[HMR19] Hastie, T., Montanari, A., Rosset, S., and Tibshirani, R. J., 2019. "Surprises in High-Dimensional Ridgeless Least Squares Interpolation," arXiv preprint arXiv:1903-.08560.

[HPC96] Helmsen, J., Puckett, E. G., Colella, P., and Dorr, M., 1996. "Two New Methods for Simulating Photolithography Development," SPIE's 1996 International Symposium on Microlithography, pp. 253-261.

[HSS08] Hofmann, T., Scholkopf, B., and Smola, A. J., 2008. "Kernel Methods in Machine Learning," The Annals of Statistics, Vol. 36, pp. 1171-1220.

[HSW89] Hornick, K., Stinchcombe, M., and White, H., 1989. "Multilayer Feedforward Networks are Universal Approximators," Neural Networks, Vol. 2, pp. 359-159.

[Han98] Hansen, E. A., 1998. "Solving POMDPs by Searching in Policy Space," in Proc. of the 14th Conf. on Uncertainty in Artificial Intelligence, pp. 211-219.

[Hay08] Haykin, S., 2008. Neural Networks and Learning Machines, 3rd Edition, Prentice-Hall, Englewood-Cliffs, N. J.

[IJT18] Iusem, Jofre, A., and Thompson, P., 2018. "Incremental Constraint Projection Methods for Monotone Stochastic Variational Inequalities," Math. of Operations Research, Vol. 44, pp. 236-263.

[IoS96] Ioannou, P. A., and Sun, J., 1996. Robust Adaptive Control, Prentice-Hall, Englewood Cliffs, N. J.

[Iva68] Ivakhnenko, A. G., 1968. "The Group Method of Data Handling: A Rival of the Method of Stochastic Approximation," Soviet Automatic Control, Vol. 13, pp. 43-55.

[Iva71] Ivakhnenko, A. G., 1971. "Polynomial Theory of Complex Systems," IEEE Trans. on Systems, Man and Cybernetics, Vol. 4, pp. 364-378.

[JSJ95] Jaakkola, T., Singh, S. P., and Jordan, M. I., 1995. "Reinforcement Learning Algorithm for Partially Observable Markov Decision Problems," NIPS, Vol. 7, pp. 345-352.

[JiJ17] Jiang, Y., and Jiang, Z. P., 2017. Robust Adaptive Dynamic Programming, J. Wiley, N. Y.

[JoB16] Joseph, A. G., and Bhatnagar, S., 2016. "Revisiting the Cross Entropy Method with Applications in Stochastic Global Optimization and Reinforcement Learning," in Proc. of the 22nd European Conference on Artificial Intelligence, pp. 1026-1034.

[JoB18] Joseph, A. G., and Bhatnagar, S., 2018. "A Cross Entropy Based Optimization Algorithm with Global Convergence Guarantees," arXiv preprint arXiv:1801.10291.

[Jon90] Jones, L. K., 1990. "Constructive Approximations for Neural Networks by Sigmoidal Functions," Proceedings of the IEEE, Vol. 78, pp. 1586-1589.

[JuP07] Jung, T., and Polani, D., 2007. "Kernelizing LSPE(λ)," Proc. 2007 IEEE Symposium on Approximate Dynamic Programming and Reinforcement Learning, Honolulu, Ha., pp. 338-345.

[KAC15] Kochenderfer, M. J., with Amato, C., Chowdhary, G., How, J. P., Davison Reynolds, H. J., Thornton, J. R., Torres-Carrasquillo, P. A., Ore, N. K., Vian, J., 2015. Decision Making under Uncertainty: Theory and Application, MIT Press, Cambridge, MA.

[KAH15] Khashooei, B. A., Antunes, D. J. and Heemels, W.P.M.H., 2015. "Rollout Strategies for Output-Based Event-Triggered Control," in Proc. 2015 European Control Conference, pp. 2168-2173.

[KGB82] Kimemia, J., Gershwin, S. B., and Bertsekas, D. P., 1982. "Computation of Production Control Policies by a Dynamic Programming Technique," in Analysis and Optimization of Systems, A. Bensoussan and J. L. Lions (eds.), Springer, N. Y., pp. 243-269.

[KKK95] Krstic, M., Kanellakopoulos, I., Kokotovic, P., 1995. Nonlinear and Adaptive Control Design, J. Wiley, N. Y.

[KLC98] Kaelbling, L. P., Littman, M. L., and Cassandra, A. R., 1998. "Planning and Acting in Partially Observable Stochastic Domains," Artificial Intelligence, Vol. 101, pp. 99-134.

[KLM96] Kaelbling, L. P., Littman, M. L., and Moore, A. W., 1996. "Reinforcement Learning: A Survey," J. of Artificial Intelligence Res., Vol. 4, pp. 237-285.

[KMP06] Keller, P. W., Mannor, S., and Precup, D., 2006. "Automatic Basis Function Construction for Approximate Dynamic Programming and Reinforcement Learning," Proc. of the 23rd ICML, Pittsburgh, Penn.

[KRC13] Kroese, D. P., Rubinstein, R. Y., Cohen, I., Porotsky, S., and Taimre, T., 2013. "Cross-Entropy Method," in Encyclopedia of Operations Research and Management Science, Springer, Boston, MA, pp. 326-333.

[Kak02] Kakade, S. A., 2002. "Natural Policy Gradient," NIPS, Vol. 14, pp. 1531-1538.

[KeG88] Keerthi, S. S., and Gilbert, E. G., 1988. "Optimal, Infinite Horizon Feedback Laws for a General Class of Constrained Discrete Time Systems: Stability and Moving-Horizon Approximations," J. Optimization Theory Appl., Vo. 57, pp. 265-293.

[KiK06] Killingsworth, N. J., and Krstic, M., 2006. "PID Tuning Using Extremum Seeking," IEEE Control Systems Magazine, pp. 70-79.

[Kim82] Kimemia, J., 1982. "Hierarchical Control of Production in Flexible Manufacturing Systems," Ph.D. Thesis, Dep. of Electrical Engineering and Computer Science, Massachusetts Institute of Technology.

[KoC15] Kouvaritakis, B., and Cannon, M., 2015. Model Predictive Control: Classical, Robust and Stochastic, Springer, N. Y.

[KoS06] Kocsis, L., and Szepesvari, C., 2006. "Bandit Based Monte-Carlo Planning," Proc. of 17th European Conference on Machine Learning, Berlin, pp. 282-293.

[KoT99] Konda, V. R., and Tsitsiklis, J. N., 1999. "Actor-Critic Algorithms," NIPS, Denver, Colorado, pp. 1008-1014.

[KoT03] Konda, V. R., and Tsitsiklis, J. N., 2003. "Actor-Critic Algorithms," SIAM J. on Control and Optimization, Vol. 42, pp. 1143-1166.

[Kor90] Korf, R. E., 1990. "Real-Time Heuristic Search," Artificial Intelligence, Vol. 42, pp. 189-211.

[Kre19] Krener, A. J., 2019. "Adaptive Horizon Model Predictive Control and Al'brekht's Method," arXiv preprint arXiv:1904.00053.

[Kri16] Krishnamurthy, V., 2016. Partially Observed Markov Decision Processes, Cambridge Univ. Press.

[KuD92] Kushner, H. J., and Dupuis, P. G., 1992. Numerical Methods for Stochastic Control Problems in Continuous Time, Springer, N. Y.

[KuV86] Kumar, P. R., and Varaiya, P. P., 1986. Stochastic Systems: Estimation, Identification, and Adaptive Control, Prentice-Hall, Englewood Cliffs, N. J.

[KuY03] Kushner, H. J., and Yin, G., 2003. Stochastic Approximation and Recursive Algorithms and Applications, (2nd Ed.), Springer-Verlag, New York.

[Kum83] Kumar, P. R., 1983. "Optimal Adaptive Control of Linear-Quadratic-Gaussian Systems," SIAM J. on Control and Optimization, Vol. 21, pp. 163-178.

[Kun14] Kung, S. Y., 2014. Kernel Methods and Machine Learning, Cambridge Univ. Press.

[L'Ec91] L'Ecuyer, P., 1991. "An Overview of Derivative Estimation," Proceedings of the 1991 Winter Simulation Conference, pp. 207-217.

[LGM03] Lequin, O., Gevers, M., Mossberg, M., Bosmans, E., and Triest, L., 2003. "Iterative Feedback Tuning of PID Parameters: Comparison with Classical Tuning Rules," Control Engineering Practice, Vol. 11, pp. 1023-1033.

[LGM10] Lazaric, A., Ghavamzadeh, M., and Munos, R., 2010. "Analysis of a Classification-Based Policy Iteration Algorithm," INRIA Report.

[LGW16] Lan, Y., Guan, X., and Wu, J., 2016. "Rollout Strategies for Real-Time Multi-Energy Scheduling in Microgrid with Storage System," IET Generation, Transmission and Distribution, Vol. 10, pp. 688-696.

[LLL08] Lewis, F. L., Liu, D., and Lendaris, G. G., 2008. Special Issue on Adaptive Dynamic Programming and Reinforcement Learning in Feedback Control, IEEE Trans. on Systems, Man, and Cybernetics, Part B, Vol. 38, No. 4.

[LLP93] Leshno, M., Lin, V. Y., Pinkus, A., and Schocken, S., 1993. "Multilayer Feedforward Networks with a Nonpolynomial Activation Function can Approximate any Function," Neural Networks, Vol. 6, pp. 861-867.

[LWW17] Liu, D., Wei, Q., Wang, D., Yang, X., and Li, H., 2017. Adaptive Dynamic Programming with Applications in Optimal Control, Springer, Berlin.

[LaP03] Lagoudakis, M. G., and Parr, R., 2003. "Reinforcement Learning as Classification: Leveraging Modern Classifiers," in Proc. of ICML, pp. 424-431.

[LaR85] Lai, T., and Robbins, H., 1985. "Asymptotically Efficient Adaptive Allocation Rules," Advances in Applied Mathematics, Vol. 6, pp. 4-22.

[LeL13] Lewis, F. L., and Liu, D., (Eds), 2013. Reinforcement Learning and Approximate Dynamic Programming for Feedback Control, Wiley, Hoboken, N. J.

[LeV09] Lewis, F. L., and Vrabie, D., 2009. "Reinforcement Learning and Adaptive Dynamic Programming for Feedback Control," IEEE Circuits and Systems Magazine, 3rd Q. Issue.

[Lee17] Lee, J., 2017. "A Survey of Robot Learning from Demonstrations for Human-Robot Collaboration," arXiv preprint arXiv:1710.08789.

[LiW14] Liu, D., and Wei, Q., 2014. "Policy Iteration Adaptive Dynamic Programming Algorithm for Discrete-Time Nonlinear Systems," IEEE Trans. on Neural Networks and Learning Systems, Vol. 25, pp. 621-634.

[Li17] Li, Y., 2017. "Deep Reinforcement Learning: An Overview," arXiv preprint ArXiv: 1701.07274v5.

[LiR06] Lincoln, B., and Rantzer, A., 2006. "Relaxing Dynamic Programming," IEEE Trans. Automatic Control, Vol. 51, pp. 1249-1260.

[LiS16] Liang, S., and Srikant, R., 2016. "Why Deep Neural Networks for Function Approximation?" arXiv preprint arXiv:1610.04161.

[LiW15] Li, H., and Womer, N. K., 2015. "Solving Stochastic Resource-Constrained Project Scheduling Problems by Closed-Loop Approximate Dynamic Programming," European J. of Operational Research, Vol. 246, pp. 20-33.

[Lib11] Liberzon, D., 2011. Calculus of Variations and Optimal Control Theory: A Concise Introduction, Princeton Univ. Press.

[Liu01] Liu, J. S., 2001. Monte Carlo Strategies in Scientific Computing, Springer, N. Y.

[LoS01] Longstaff, F. A., and Schwartz, E. S., 2001. "Valuing American Options by Simulation: A Simple Least-Squares Approach," Review of Financial Studies, Vol. 14, pp. 113-147.

[MBT05] Mitchell, I. M., Bayen, A. M., and Tomlin, C. J., 2005. "A Time-Dependent Hamilton-Jacobi Formulation of Reachable Sets for Continuous Dynamic Games," IEEE Trans. on Automatic Control, Vol. 50, pp. 947-957.

[MKS15] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., and Petersen, S., 2015. "Human-Level Control Through Deep Reinforcement Learning," Nature, Vol. 518, p. 529.

[MMB02] McGovern, A., Moss, E., and Barto, A., 2002. "Building a Basic Building Block Scheduler Using Reinforcement Learning and Rollouts," Machine Learning, Vol. 49, pp. 141-160.

[MMS05] Menache, I., Mannor, S., and Shimkin, N., 2005. "Basis Function Adaptation in Temporal Difference Reinforcement Learning," Ann. Oper. Res., Vol. 134, pp. 215-238.

[MPK99] Meuleau, N., Peshkin, L., Kim, K. E., and Kaelbling, L. P., 1999. "Learning Finite-State Controllers for Partially Observable Environments," in Proc. of the 15th Conference on Uncertainty in Artificial Intelligence, pp. 427-436.

[MPP04] Meloni, C., Pacciarelli, D., and Pranzo, M., 2004. "A Rollout Metaheuristic for Job Shop Scheduling Problems," Annals of Operations Research, Vol. 131, pp. 215-235.

[MRG03] Mannor, S., Rubinstein, R. Y., and Gat, Y., 2003. "The Cross Entropy Method for Fast Policy Search," in Proc. of the 20th International Conference on Machine Learning (ICML-03), pp. 512-519.

[MVS19] Muthukumar, V., Vodrahalli, K., and Sahai, A., 2019. "Harmless Interpolation of Noisy Data in Regression," arXiv preprint arXiv:1903.09139.

[MYF03] Moriyama, H., Yamashita, N., and Fukushima, M., 2003. "The Incremental Gauss-Newton Algorithm with Adaptive Stepsize Rule," Computational Optimization and Applications, Vol. 26, pp. 107-141.

[MaJ15] Mastin, A., and Jaillet, P., 2015. "Average-Case Performance of Rollout Algorithms for Knapsack Problems," J. of Optimization Theory and Applications, Vol. 165, pp. 964-984.

[MaK12] Mausam, and Kolobov, A., 2012. "Planning with Markov Decision Processes: An AI Perspective," Synthesis Lectures on Artificial Intelligence and Machine Learning, Vol. 6, pp. 1-210.

[MaT01] Marbach, P., and Tsitsiklis, J. N., 2001. "Simulation-Based Optimization of Markov Reward Processes," IEEE Trans. on Aut. Control, Vol. 46, pp. 191-209.

[MaT03] Marbach, P., and Tsitsiklis, J. N., 2003. "Approximate Gradient Methods in Policy-Space Optimization of Markov Reward Processes," J. Discrete Event Dynamic Systems, Vol. 13, pp. 111-148.

[Mac02] Maciejowski, J. M., 2002. Predictive Control with Constraints, Addison-Wesley, Reading, MA.

[Mat65] J. Matyas, J., 1965. "Random Optimization," Automation and Remote Control, Vol. 26, pp. 246-253.

[May14] Mayne, D. Q., 2014. "Model Predictive Control: Recent Developments and Future Promise," Automatica, Vol. 50, pp. 2967-2986.

[MeB99] Meuleau, N., and Bourgine, P., 1999. "Exploration of Multi-State Environments: Local Measures and Back-Propagation of Uncertainty," Machine Learning, Vol. 35, pp. 117-154.

[Mey07] Meyn, S., 2007. Control Techniques for Complex Networks, Cambridge Univ. Press, N. Y.

[MoL99] Morari, M., and Lee, J. H., 1999. "Model Predictive Control: Past, Present, and Future," Computers and Chemical Engineering, Vol. 23, pp. 667-682.

[MuS08] Munos, R., and Szepesvari, C, 2008. "Finite-Time Bounds for Fitted Value Iteration," J. of Machine Learning Research, Vol. 1, pp. 815-857.

[Mun14] Munos, R., 2014. "From Bandits to Monte-Carlo Tree Search: The Optimistic Principle Applied to Optimization and Planning," Foundations and Trends in Machine Learning, Vol. 7, pp. 1-129.

[NHR99] Ng, A. Y., Harada, D., and Russell, S. J., 1999. "Policy Invariance Under Reward Transformations: Theory and Application to Reward Shaping," in Proc. of the 16th International Conference on Machine Learning, pp. 278-287.

[NeB00] Nedić, A., and Bertsekas, D. P., 2000. "Convergence Rate of Incremental Subgradient Algorithms," in Stochastic Optimization: Algorithms and Applications, by S. Uryasev and P. M. Pardalos, Eds., Kluwer, pp. 263-304.

[NeB01] Nedić, A., and Bertsekas, D. P., 2001. "Incremental Subgradient Methods for Nondifferentiable Optimization," SIAM J. on Optimization, Vol. 12, pp. 109-138.

[NeB03] Nedić, A., and Bertsekas, D. P., 2003. "Least-Squares Policy Evaluation Algorithms with Linear Function Approximation," J. of Discrete Event Systems, Vol. 13, pp. 79-110.

[NeS12] Neu, G., and Szepesvari, C., 2012. "Apprenticeship Learning Using Inverse Reinforcement Learning and Gradient Methods," arXiv preprint arXiv:1206.5264.

[NeS17] Nesterov, Y., and Spokoiny, V., 2017. "Random Gradient-Free Minimization of Convex Functions," Foundations of Computational Mathematics, Vol. 17, pp. 527-566.

[Ned11] Nedić, A., 2011. "Random Algorithms for Convex Minimization Problems," Math. Programming, Ser. B, Vol. 129, pp. 225-253.

[OVR19] Osband, I., Van Roy, B., Russo, D. J., and Wen, Z., 2019. "Deep Exploration via Randomized Value Functions," arXiv preprint arXiv: 1703.07608v4.

[OrS02] Ormoneit, D., and Sen, S., 2002. "Kernel-Based Reinforcement Learning," Machine Learning, Vol. 49, pp. 161-178.

[PBT98] Polymenakos, L. C., Bertsekas, D. P., and Tsitsiklis, J. N., 1998. "Efficient Algorithms for Continuous-Space Shortest Path Problems," IEEE Trans. on Automatic Control, Vol. 43, pp. 278-283.

[PDC14] Pillonetto, G., Dinuzzo, F., Chen, T., De Nicolao, G., and Ljung, L., 2014. "Kernel Methods in System Identification, Machine Learning and Function Estimation: A Survey," Automatica, Vol. 50, pp. 657-682.

[PSS98] Precup, D., Sutton, R. S., and Singh, S., 1998. "Theoretical Results on Reinforcement Learning with Temporally Abstract Options," in European Conf. on Machine Learning, Springer, Berlin, pp. 382-393.

[PaT00] Paschalidis, I. C., and Tsitsiklis, J. N., 2000. "Congestion-Dependent Pricing of Network Services," IEEE/ACM Trans. on Networking, Vol. 8, pp. 171-184.

[Pat01] Patek, S. D., 2001. "On Terminating Markov Decision Processes with a Risk Averse Objective Function," Automatica, Vol. 37, pp. 1379-1386.

[Pat07] Patek, S. D., 2007. "Partially Observed Stochastic Shortest Path Problems with Approximate Solution by Neuro-Dynamic Programming," IEEE Trans. on Systems, Man, and Cybernetics Part A, Vol. 37, pp. 710-720.

[PeG04] Peret, L., and Garcia, F., 2004. "On-Line Search for Solving Markov Decision Processes via Heuristic Sampling," in Proc. of the 16th European Conference on Artificial Intelligence, pp. 530-534.

[PeS08] Peters, J., and Schaal, S., 2008. "Reinforcement Learning of Motor Skills with Policy Gradients," Neural Networks, Vol. 4, pp. 682-697.

[PeW96] Peng, J., and Williams, R., 1996. "Incremental Multi-Step Q-Learning," Machine Learning, Vol. 22, pp. 283-290.

[PoB04] Poupart, P., and Boutilier, C., 2004. "Bounded Finite State Controllers," in Advances in Neural Information Processing Systems, pp. 823-830.

[PoV04] Powell, W. B., and Van Roy, B., 2004. "Approximate Dynamic Programming for High-Dimensional Resource Allocation Problems," in Learning and Approximate Dynamic Programming, by J. Si, A. Barto, W. Powell, and D. Wunsch, (Eds.), IEEE Press, N. Y.

[Pow11] Powell, W. B., 2011. Approximate Dynamic Programming: Solving the Curses of Dimensionality, 2nd Edition, J. Wiley and Sons, Hoboken, N. J.

[Pre95] Prekopa, A., 1995. Stochastic Programming, Kluwer, Boston.

[PuS78] Puterman, M. L., and Shin, M. C., 1978. "Modified Policy Iteration Algorithms for Discounted Markov Decision Problems," Management Sci., Vol. 24, pp. 1127-1137.

[PuS82] Puterman, M. L., and Shin, M. C., 1982. "Action Elimination Procedures for Modified Policy Iteration Algorithms," Operations Research, Vol. 30, pp. 301-318.

[Put94] Puterman, M. L., 1994. Markovian Decision Problems, J. Wiley, N. Y.

[RPW91] Rogers, D. F., Plante, R. D., Wong, R. T., and Evans, J. R., 1991. "Aggregation and Disaggregation Techniques and Methodology in Optimization," Operations Research, Vol. 39, pp. 553-582.

[RSM08] Reisinger, J., Stone, P., and Miikkulainen, R., 2008. "Online Kernel Selection for Bayesian Reinforcement Learning," in Proc. of the 25th International Conference on Machine Learning, pp. 816-823.

[RVK18] Russo, D. J., Van Roy, B., Kazerouni, A., Osband, I., and Wen, Z., 2018. A Tutorial on Thompson Sampling, Foundations and Trends in Machine Learning.

[RaK18] Radenkovic, S., and Krstic, M., 2018. "Extremum Seeking-Based Perfect Adaptive Tracking of Non-PE References Despite Nonvanishing Variance of Perturbations," Automatica, Vol. 93, pp. 189-196.

[Ras63] Rastrigin, R. A., 1963. "About Convergence of Random Search Method in Extremal Control of Multi-Parameter Systems," Avtomat. i Telemekh., Vol. 24, pp. 1467-1473.

[Rec18a] Recht, B., 2018. "A Tour of Reinforcement Learning: The View from Continuous Control," Annual Review of Control, Robotics, and Autonomous Systems.

[Rec18b] Recht, B., 2018. "An Outsider's Tour of Reinforcement Learning," at http://www.argmin.net/2018/06/25/outsider-rl/

[RoC10] Robert, C. P., and Casella, G., 2010. Monte Carlo Statistical Methods, Springer, N. Y.

[Ros70] Ross, S. M., 1970. Applied Probability Models with Optimization Applications, Holden-Day, San Francisco, CA.

[Ros12] Ross, S. M., 2012. Simulation, 5th Edition, Academic Press, Orlando, Fla.

[RuK04] Rubinstein, R. Y., and Kroese, D. P., 2004. The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Springer, N. Y.

[RuK13] Rubinstein, R. Y., and Kroese, D. P., 2013. The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation and Machine Learning, Springer Science and Business Media.

[RuK16] Rubinstein, R. Y., and Kroese, D. P., 2016. Simulation and the Monte Carlo Method, 3rd Edition, J. Wiley, N. Y.

[RuN94] Rummery, G. A., and Niranjan, M., 1994. "On-Line Q-Learning Using Connectionist Systems," University of Cambridge, England, Department of Engineering, TR-166.

[RuN16] Russell, S. J., and Norvig, P., 2016. Artificial Intelligence: A Modern Approach, Pearson Education Limited, Malaysia.

[RuV16] Russo, D., and Van Roy, B., 2016. "An Information-Theoretic Analysis of Thompson Sampling," The Journal of Machine Learning Research, Vol. 17, pp. 2442-2471.

[Rub69] Rubinstein, R. Y., 1969. Some Problems in Monte Carlo Optimization, Ph.D. Thesis.

[SBP04] Si, J., Barto, A., Powell, W., and Wunsch, D., (Eds.) 2004. Learning and Approximate Dynamic Programming, IEEE Press, N. Y.

[SGC02] Savagaonkar, U., Givan, R., and Chong, E. K. P., 2002. "Sampling Techniques for Zero-Sum, Discounted Markov Games," in Proc. 40th Allerton Conference on Communication, Control and Computing, Monticello, Ill.

[SGG15] Scherrer, B., Ghavamzadeh, M., Gabillon, V., Lesner, B., and Geist, M., 2015. "Approximate Modified Policy Iteration and its Application to the Game of Tetris," J. of Machine Learning Research, Vol. 16, pp. 1629-1676.

[SHB15] Simroth, A., Holfeld, D., and Brunsch, R., 2015. "Job Shop Production Planning under Uncertainty: A Monte Carlo Rollout Approach," Proc. of the International Scientific and Practical Conference, Vol. 3, pp. 175-179.

[SHC17] Salimans, T., Ho, J., Chen, X., Sidor, S., and Sutskever, I., 2017. "Evolution Strategies as a Scalable Alternative to Reinforcement Learning," arXiv preprint arXiv:1703.03864.

[SHM16] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., and Dieleman, S., 2016. "Mastering the Game of Go with Deep Neural Networks and Tree Search," Nature, Vol. 529, pp. 484-489.

[SHS17] Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A.,

Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., and Lillicrap, T., 2017. "Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm," arXiv preprint arXiv:1712.01815.

[SJJ95] Singh, S. P., Jaakkola, T., and Jordan, M. I., 1995. "Reinforcement Learning with Soft State Aggregation," in Advances in Neural Information Processing Systems 7, MIT Press, Cambridge, MA.

[SJL18] Soltanolkotabi, M., Javanmard, A., and Lee, J. D., 2018. "Theoretical Insights into the Optimization Landscape of Over-Parameterized Shallow Neural Networks," IEEE Trans. on Information Theory, Vol. 65, pp. 742-769.

[SLJ13] Sun, B., Luh, P. B., Jia, Q. S., Jiang, Z., Wang, F., and Song, C., 2013. "Building Energy Management: Integrated Control of Active and Passive Heating, Cooling, Lighting, Shading, and Ventilation Systems," IEEE Trans. on Automation Science and Engineering, Vol. 10, pp. 588-602.

[SMS99] Sutton, R. S., McAllester, D., Singh, S. P., and Mansour, Y., 1999. "Policy Gradient Methods for Reinforcement Learning with Function Approximation," NIPS, Denver, Colorado.

[SSP18] Serban, I. V., Sankar, C., Pieper, M., Pineau, J., Bengio, J., 2018. "The Bottleneck Simulator: A Model-Based Deep Reinforcement Learning Approach," arXiv preprint arXiv:1807.04723.v1.

[SYL04] Si, J., Yang, L., and Liu, D., 2004. "Direct Neural Dynamic Programming," in Learning and Approximate Dynamic Programming, by J. Si, A. Barto, W. Powell, and D. Wunsch, (Eds.), IEEE Press, N. Y.

[SYL17] Saldi, N., Yuksel, S., and Linder, T., 2017. "Finite Model Approximations for Partially Observed Markov Decision Processes with Discounted Cost," arXiv preprint arXiv:1710.07009.

[SZL08] Sun, T., Zhao, Q., Lun, P., and Tomastik, R., 2008. "Optimization of Joint Replacement Policies for Multipart Systems by a Rollout Framework," IEEE Trans. on Automation Science and Engineering, Vol. 5, pp. 609-619.

[SaB11] Sastry, S., and Bodson, M., 2011. Adaptive Control: Stability, Convergence and Robustness, Courier Corporation.

[Sam59] Samuel, A. L., 1959. "Some Studies in Machine Learning Using the Game of Checkers," IBM J. of Research and Development, pp. 210-229.

[Sam67] Samuel, A. L., 1967. "Some Studies in Machine Learning Using the Game of Checkers. II – Recent Progress," IBM J. of Research and Development, pp. 601-617.

[ScS85] Schweitzer, P. J., and Seidman, A., 1985. "Generalized Polynomial Approximations in Markovian Decision Problems," J. Math. Anal. and Appl., Vol. 110, pp. 568-582.

[ScS02] Scholkopf, B., and Smola, A. J., 2002. Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond, MIT Press, Cambridge, MA.

[Sch99] Schaal, S., 1999. "Is Imitation Learning the Route to Humanoid Robots?" Trends in Cognitive Sciences, Vol. 3, pp. 233-242.

[Sch13] Scherrer, B., 2013. "Performance Bounds for Lambda Policy Iteration and Application to the Game of Tetris," J. of Machine Learning Research, Vol. 14, pp. 1181-1227.

[Sch15] Schmidhuber, J., 2015. "Deep Learning in Neural Networks: An Overview," Neural Networks, pp. 85-117.

[Sec00] Secomandi, N., 2000. "Comparing Neuro-Dynamic Programming Algorithms for the Vehicle Routing Problem with Stochastic Demands," Computers and Operations Research, Vol. 27, pp. 1201-1225.

[Sec01] Secomandi, N., 2001. "A Rollout Policy for the Vehicle Routing Problem with Stochastic Demands," Operations Research, Vol. 49, pp. 796-802.

[Sec03] Secomandi, N., 2003. "Analysis of a Rollout Approach to Sequencing Problems with Stochastic Routing Applications," J. of Heuristics, Vol. 9, pp. 321-352.

[Set99a] Sethian, J. A., 1999. Level Set Methods and Fast Marching Methods Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science, Cambridge University Press, N. Y.

[Set99b] Sethian, J. A., 1999. "Fast Marching Methods," SIAM Review, Vol. 41, pp. 199-235.

[ShC04] Shawe-Taylor, J., and Cristianini, N., 2004. Kernel Methods for Pattern Analysis, Cambridge Univ. Press.

[Sha50] Shannon, C., 1950. "Programming a Digital Computer for Playing Chess," Phil. Mag., Vol. 41, pp. 356-375.

[SlL91] Slotine, J.-J. E., and Li, W., Applied Nonlinear Control, Prentice-Hall, Englewood Cliffs, N. J.

[SpV05] Spaan, M. T., and Vlassis, N., 2005. "Perseus: Randomized Point-Based Value Iteration for POMDPs," J. of Artificial Intelligence Research, Vol. 24, pp. 195-220.

[Spa92] Spall, J. C., "Multivariate Stochastic Approximation Using a Simultaneous Perturbation Gradient Approximation," IEEE Trans. on Automatic Control, Vol. pp. 332-341.

[Spa03] Spall, J. C., 2003. Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control, J. Wiley, Hoboken, N. J.

[Str18] Strogatz, S., 2018. "One Giant Step for a Chess-Playing Machine," NY Times article on Dec. 26, 2018.

[SuB18] Sutton, R., and Barto, A. G., 2018. Reinforcement Learning, 2nd Edition, MIT Press, Cambridge, MA.

[Sut88] Sutton, R. S., 1988. "Learning to Predict by the Methods of Temporal Differences," Machine Learning, Vol. 3, pp. 9-44.

[SzL06] Szita, I., and Lorinz, A., 2006. "Learning Tetris Using the Noisy Cross-Entropy Method," Neural Computation, Vol. 18, pp. 2936-2941.

[Sze10] Szepesvari, C., 2010. Algorithms for Reinforcement Learning, Morgan and Claypool Publishers, San Franscisco, CA.

[TCW19] Tseng, W. J., Chen, J. C., Wu, I. C., and Wei, T. H., 2019. "Comparison Training for Computer Chinese Chess," IEEE Trans. on Games.

[TGL13] Tesauro, G., Gondek, D. C., Lenchner, J., Fan, J., and Prager, J. M., 2013. "Analysis of Watson's Strategies for Playing Jeopardy!," J. of Artificial Intelligence Research, Vol. 47, pp. 205-251.

[TeG96] Tesauro, G., and Galperin, G. R., 1996. "On-Line Policy Improvement Using Monte Carlo Search," NIPS, Denver, CO.

[Tes89a] Tesauro, G. J., 1989. "Neurogammon Wins Computer Olympiad," Neural Computation, Vol. 1, pp. 321-323.

[Tes89b] Tesauro, G. J., 1989. "Connectionist Learning of Expert Preferences by Comparison Training," in Advances in Neural Information Processing Systems, pp. 99-106.

[Tes92] Tesauro, G. J., 1992. "Practical Issues in Temporal Difference Learning," Machine Learning, Vol. 8, pp. 257-277.

[Tes94] Tesauro, G. J., 1994. "TD-Gammon, a Self-Teaching Backgammon Program, Achieves Master-Level Play," Neural Computation, Vol. 6, pp. 215-219.

[Tes95] Tesauro, G. J., 1995. "Temporal Difference Learning and TD-Gammon," Communications of the ACM, Vol. 38, pp. 58-68.

[Tes01] Tesauro, G. J., 2001. "Comparison Training of Chess Evaluation Functions," in Machines that Learn to Play Games, Nova Science Publishers, pp. 117-130.

[Tes02] Tesauro, G. J., 2002. "Programming Backgammon Using Self-Teaching Neural Nets," Artificial Intelligence, Vol. 134, pp. 181-199.

[ThS09] Thiery, C., and Scherrer, B., 2009. "Improvements on Learning Tetris with Cross-Entropy," International Computer Games Association J., Vol. 32, pp. 23-33.

[ThS10] Thiery, C., and Scherrer, B., 2010. "Performane Bound for Approximate Optimistic Policy Iteration," Technical Report, INRIA, France.

[TsV96] Tsitsiklis, J. N., and Van Roy, B., 1996. "Feature-Based Methods for Large-Scale Dynamic Programming," Machine Learning, Vol. 22, pp. 59-94.

[TsV97] Tsitsiklis, J. N., and Van Roy, B., 1997. "An Analysis of Temporal-Difference Learning with Function Approximation," IEEE Trans. on Aut. Control, Vol. 42, pp. 674-690.

[TsV99a] Tsitsiklis, J. N., and Van Roy, B., 1999. "Average Cost Temporal-Difference Learning," Automatica, Vol. 35, pp. 1799-1808.

[TsV99b] Tsitsiklis, J. N., and Van Roy, B., 1999. "Optimal Stopping of Markov Processes: Hilbert Space Theory, Approximation Algorithms, and an Application to Pricing Financial Derivatives", IEEE Trans. on Aut. Control, Vol. 44, pp. 1840-1851.

[Tse98] Tseng, P., 1998. "Incremental Gradient(-Projection) Method with Momentum Term and Adaptive Stepsize Rule," SIAM J. on Optimization, Vol. 8, pp. 506-531.

[Tsi94] Tsitsiklis, J. N., 1994. "Asynchronous Stochastic Approximation and Q-Learning," Machine Learning, Vol. 16, pp. 185-202.

[Tsi95] Tsitsiklis, J. N., 1995. "Efficient Algorithms for Globally Optimal Trajectories," IEEE Trans. Automatic Control, Vol. AC-40, pp. 1528-1538.

[TuP03] Tu, F., and Pattipati, K. R., 2003. "Rollout Strategies for Sequential Fault Diagnosis," IEEE Trans. on Systems, Man and Cybernetics, Part A, pp. 86-99.

[UGM18] Ulmer, M. W., Goodson, J. C., Mattfeld, D. C., and Hennig, M., 2018. "Offline-Online Approximate Dynamic Programming for Dynamic Vehicle Routing with Stochastic Requests," Transportation Science, Vol. 53, pp. 185-202.

[Ulm17] Ulmer, M. W., 2017. Approximate Dynamic Programming for Dynamic Vehicle Routing, Springer, Berlin.

[VVL13] Vrabie, D., Vamvoudakis, K. G., and Lewis, F. L., 2013. Optimal Adaptive Control and Differential Games by Reinforcement Learning Principles, The Institution of Engineering and Technology, London.

[Vla08] Vladimirsky, A., 2008. "Label-Setting Methods for Multimode Stochastic Shortest Path Problems on Graphs," Math. of Operations Research, Vol. 33, pp. 821-838.

[WCG03] Wu, G., Chong, E. K. P., and Givan, R. L., 2003. "Congestion Control Using Policy Rollout," Proc. 2nd IEEE CDC, Maui, Hawaii, pp. 4825-4830.

[WOB15] Wang, Y., O'Donoghue, B., and Boyd, S., 2015. "Approximate Dynamic Programming via Iterated Bellman Inequalities," International J. of Robust and Nonlinear Control, Vol. 25, pp. 1472-1496.

[WaB13a] Wang, M., and Bertsekas, D. P., 2013. "Stabilization of Stochastic Iterative Methods for Singular and Nearly Singular Linear Systems," Mathematics of Operations Research, Vol. 39, pp. 1-30.

[WaB13b] Wang, M., and Bertsekas, D. P., 2013. "Convergence of Iterative Simulation-Based Methods for Singular Linear Systems," Stochastic Systems, Vol. 3, pp. 39-96.

[WaB14] Wang, M., and Bertsekas, D. P., 2014. "Incremental Constraint Projection Methods for Variational Inequalities," Mathematical Programming, pp. 1-43.

[WaB16] Wang, M., and Bertsekas, D. P., 2016. "Stochastic First-Order Methods with Random Constraint Projection," SIAM Journal on Optimization, Vol. 26, pp. 681-717.

[WaP17] Wang, J., and Paschalidis, I. C., 2017. "An Actor-Critic Algorithm with Second-Order Actor and Critic," IEEE Trans. on Automatic Control, Vol. 62, pp. 2689-2703.

[Van76] Van Nunen, J. A., 1976. Contracting Markov Decision Processes, Mathematical Centre Report, Amsterdam.

[Wat89] Watkins, C. J. C. H., Learning from Delayed Rewards, Ph.D. Thesis, Cambridge Univ., England.

[WeB99] Weaver, L., and Baxter, J., 1999. "Reinforcement Learning From State and Temporal Differences," Tech. Report, Department of Computer Science, Australian National University.

[WhS92] White, D., and Sofge, D., (Eds.), 1992. Handbook of Intelligent Control, Van Nostrand, N. Y.

[WhS94] White, C. C., and Scherer, W. T., 1994. "Finite-Memory Suboptimal Design for Partially Observed Markov Decision Processes," Operations Research, Vol. 42, pp. 439-455.

[Whi88] Whittle, P., 1988. "Restless Bandits: Activity Allocation in a Changing World," J. of Applied Probability, pp. 287-298.

[Whi91] White, C. C., 1991. "A Survey of Solution Techniques for the Partially Observed Markov Decision Process," Annals of Operations Research, Vol. 32, pp. 215-230.

[WiB93] Williams, R. J., and Baird, L. C., 1993. "Analysis of Some Incremental Variants of Policy Iteration: First Steps Toward Understanding Actor-Critic Learning Systems," Report NU-CCS-93-11, College of Computer Science, Northeastern University, Boston, MA.

[WiS98] Wiering, M., and Schmidhuber, J., 1998. "Fast Online Q(λ)," Machine Learning, Vol. 33, pp. 105-115.

[Wie03] Wiewiora, E., 2003. Potential-Based Shaping and Q-Value Initialization are Equivalent," J. of Artificial Intelligence Research, Vol. 19, pp. 205-208.

[Wil92] Williams, R. J., 1992. "Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning," Machine Learning, Vol. 8, pp. 229-256.

[YDR04] Yan, X., Diaconis, P., Rusmevichientong, P., and Van Roy, B., 2004. "Solitaire: Man Versus Machine," Advances in Neural Information Processing Systems, Vol. 17, pp. 1553-1560.

[Yar17] Yarotsky, D., 2017. "Error Bounds for Approximations with Deep ReLU Networks," Neural Networks, Vol. 94, pp. 103-114.

[YuB04] Yu, H., and Bertsekas, D. P., 2004. "Discretized Approximations for POMDP with Average Cost," Proc. of the 20th Conference on Uncertainty in Artificial Intelligence, Banff, Canada.

[YuB06] Yu, H., and Bertsekas, D. P., 2006. "On Near-Optimality of the Set of Finite-State Controllers for Average Cost POMDP," Lab. for Information and Decision Systems Report LIDS-P-2689, MIT; Mathematics of Operations Research, Vol. 33, 2008, pp. 1-11.

[YuB07] Yu, H., and Bertsekas, D. P., 2007. "A Least Squares Q-Learning Algorithm for Optimal Stopping Problems," Proc. European Control Conference 2007, Kos, Greece, pp. 2368-2375; an extended version appears in Lab. for Information and Decision Systems Report LIDS-P-2731, MIT.

[YuB09a] Yu, H., and Bertsekas, D. P., 2009. "Convergence Results for Some Temporal Difference Methods Based on Least Squares," IEEE Trans. on Aut. Control, Vol. 54, pp. 1515-1531.

[YuB09b] Yu, H., and Bertsekas, D. P., 2009. "Basis Function Adaptation Methods for Cost Approximation in MDP," Proceedings of 2009 IEEE Symposium on Approximate Dynamic Programming and Reinforcement Learning (ADPRL 2009), Nashville, Tenn.

[YuB10] Yu, H., and Bertsekas, D. P., 2010. "Error Bounds for Approximations from Projected Linear Equations," Mathematics of Operations Research, Vol. 35, pp. 306-329.

[YuB12] Yu, H., and Bertsekas, D. P., 2012. "Weighted Bellman Equations and their Applications in Dynamic Programming," Lab. for Information and Decision Systems Report LIDS-P-2876, MIT.

[YuB13a] Yu, H., and Bertsekas, D. P., 2013. "Q-Learning and Policy Iteration Algorithms for Stochastic Shortest Path Problems," Annals of Operations Research, Vol. 208, pp. 95-132.

[YuB13b] Yu, H., and Bertsekas, D. P., 2013. "On Boundedness of Q-Learning Iterates for Stochastic Shortest Path Problems," Math. of OR, Vol. 38, pp. 209-227.

[YuB15] Yu, H., and Bertsekas, D. P., 2015. "A Mixed Value and Policy Iteration Method for Stochastic Control with Universally Measurable Policies," Math. of OR, Vol. 40, pp. 926-968.

[Yu05] Yu, H., 2005. "A Function Approximation Approach to Estimation of Policy Gradient for POMDP with Structured Policies," Proc. of the 21st Conference on Uncertainty in Artificial Intelligence, Edinburgh, Scotland.

[Yu12] Yu, H., 2012. "Least Squares Temporal Difference Methods: An Analysis Under General Conditions," SIAM J. on Control and Optimization, Vol. 50, pp. 3310-3343.

[Yu15] Yu, H., 2015. "On Convergence of Value Iteration for a Class of Total Cost Markov Decision Processes," SIAM J. on Control and Optimization, Vol. 53, pp. 1982-2016.

[ZBH16] Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O., 2016. "Understanding Deep Learning Requires Rethinking Generalization," arXiv preprint arXiv: 1611.03530.

[ZhO11] Zhang, C., and Ordonez, R., 2011. Extremum-Seeking Control and Applications: A Numerical Optimization-Based Approach, Springer Science and Business Media.

[ZhT11] Zhong, M., and Todorov, E., 2011. "Aggregation Methods for Linearly-Solvable Markov Decision Process," IFAC Proc., Vol. 44, pp. 11220-11225.