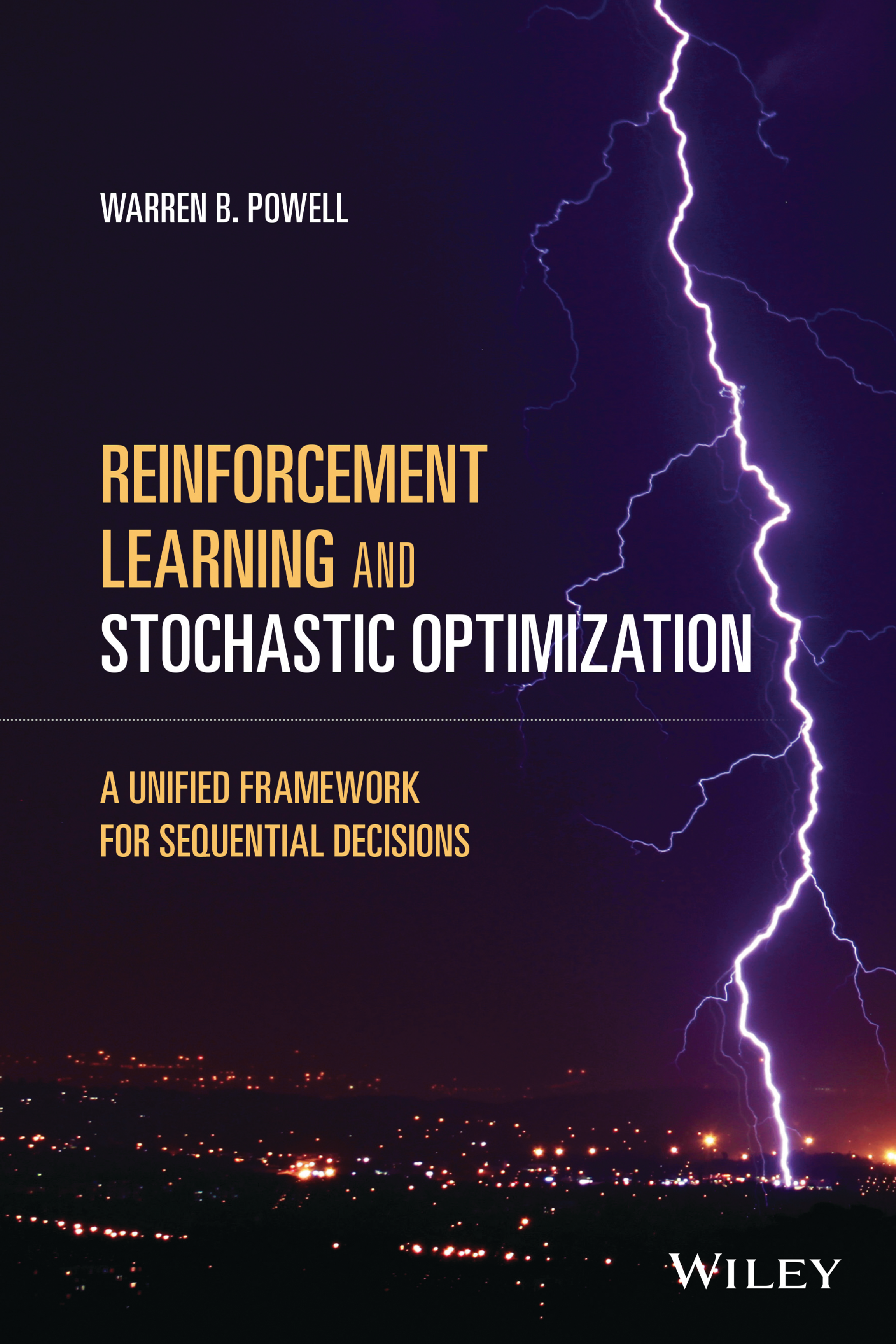

WARREN B. POWELL

REINFORCEMENT LEARNING AND STOCHASTIC OPTIMIZATION

A UNIFIED FRAMEWORK FOR SEQUENTIAL DECISIONS

WILEY

Reinforcement Learning and Stochastic Optimization:
A Unified Framework for Sequential Decisions

Reinforcement Learning and Stochastic Optimization

A Unified Framework for Sequential Decisions

Warren B. Powell
Princeton University
Princeton, NJ

WILEY

This edition first published 2022

Registered Office
John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030, USA

Editorial Office
111 River Street, Hoboken, NJ 07030, USA

For details of our global editorial offices, customer services, and more information about Wiley products visit us at www.wiley.com.

Wiley also publishes its books in a variety of electronic formats and by print-on-demand. Some content that appears in standard print versions of this book may not be available in other formats.

A catalogue record for this book is available from the Library of Congress

Hardback ISBN: 9781119815037; ePub ISBN: 9781119815051; ePDF ISBN: 9781119815044;
Obook ISBN: 9781119815068.

Cover image: © DeniseSmit/Getty Images
Cover design by Wiley

Set in 9.5/12.5pt STIXTwoText by Integra Software Services Pvt. Ltd, Pondicherry, India

10 9 8 7 6 5 4 3 2 1

Contents

Preface

Preface to *Reinforcement Learning and Stochastic Optimization: A unified framework for sequential decisions*

This books represents a lifetime of research into what I now call sequential decision problems, which dates to 1982 when I was introduced to the problem arising in truckload trucking (think of Uber/Lyft for trucks) where we have to choose which driver to assign to a load, and which loads to accept to move, given the high level of randomness in future customer demands, representing requests to move full truckloads of freight.

It took me 20 years to figure out a practical algorithm to solve this problem, which led to my first book (in 2007) on approximate dynamic programming, where the major breakthrough was the introduction of the post-decision state and the use of hierarchical aggregation for approximating value functions to solve these high-dimensional problems. However, I would argue today that the most important chapter in the book (and I recognized it at the time), was chapter 5 on how to model these problems, without any reference to algorithms to solve the problem. I identified five elements to a sequential decision problem, leading up to the objective function which was written

$$\max_{\pi} \mathbb{E}\left\{\sum_{t=0}^{T} C(S_t, X^{\pi}(S_t))|S_0\right\}.$$

It was not until the second edition (in 2011) that I realized that approximate dynamic programming (specifically, policies that depend on value functions) was not the only way to solve these problems; rather, there were four classes of policies, and only one used value functions. The 2011 edition of the book listed three of the four classes of policies that are described in this book, but most of the book still focused on approximating value functions. It was not until a 2014 paper ("Clearing the Jungle of Stochastic Optimization") that I identified the four classes of policies I use now. Then, in 2016 I realized that the four classes of policies could be divided between two major strategies: the policy search strategy, where we search over a family of functions to find the one that works best, and the lookahead strategy, where we make good decisions by approximating the downstream impact of a decision made now.

Finally, I combined these ideas in a 2019 paper ("A Unified Framework for Stochastic Optimization" published in the *European Journal for Operational Research*) with a better appreciation of major classes of problems such as state-independent problems (the pure learning problems that include derivative-based and derivative-free stochastic search) and the more general state-dependent problems; cumulative and final reward objective functions; and the realization that any adaptive search algorithm was a sequential decision problem. The material in the 2019 paper is effectively the outline for this book.

This book builds on the 2011 edition of my approximate dynamic programming book, and includes a number of chapters (some heavily edited) from the ADP book. It would be nice to call this a third edition, but the entire framework of this book is completely different. "Approximate dynamic programming" is a term that still refers to making decisions based on the idea of approximating the downstream value of being in a state. After decades of working with this approach (which is still covered over a span of five chapters in this volume), I can now say with confidence that value function approximations, despite all the attention they have received, is a powerful methodology for a surprisingly narrow set of decision problems.

By contrast, I finally developed the confidence to claim that the four classes of policies are universal. This means that *any* method for making decisions will fall in one of these four classes, or a hybrid of two or more. This is a game changer, because it shifts the focus from an algorithm (the method for making decisions) to the model (specifically the optimization problem above, along with the state-transition function and the model of the exogenous information process). This means we write out the elements of a problem *before* we tackle the problem of designing policies to decisions. I call this:

Model first, then solve.

The communities working on sequential decision problems are very focused on methods, just as I was with my earlier work with approximate dynamic programming. The problem is that any particular method will be inherently limited to a narrow class of problems. In this book, I demonstrate how you can take a simple inventory problem, and then tweak the data to make each of the four classes work best.

This new approach has opened up an entirely new way of approaching a problem class that, in the last year of writing the book, I started calling "sequential decision analytics," which is any problem consisting of the sequence:

Decision, information, decision, information, ….

I allow decisions to range from binary (selling an asset) to discrete choices (favored in computer science) to the high-dimensional resource allocation problems popular in operations research. This approach starts with a problem, shifts to the challenging task of modeling uncertainty, and then finishes with designing policies to make decisions to optimize some metric. The approach is practical, scalable, and universally applicable.

It is exciting to be able to create a single framework that spans 15 different communities, and which represents every possible method for solving sequential decision problems. While having a common language to model any sequential decision problem, combined with the general approach of the four classes of policies, is clearly of value, this framework has been developed by standing on the shoulders of the giants who have laid the foundational work for all of these methods. I have had to make choices regarding the best notation and modeling conventions, but my framework is completely inclusive of all the methods that have been developed to solve these problems. Rather than joining the chorus of researchers promoting specific algorithmic strategies (as I once did), my goal is to raise the visibility of all methods, so that someone looking to solve a real problem is working with the biggest possible toolbox, rather than just the tools developed within a specific community.

A word needs to be said about the title of the book. As this is being written, there is a massive surge of interest in "reinforcement learning," which started as a form of approximate dynamic programming (I used to refer to ADP and RL as similar to American English and British English). However, as the RL community has grown and started working on harder problems, they encountered the same experience that I and everyone else working in ADP found: value function approximations are not a panacea. Not only is it the case that they often do not work, they usually do not work. As a result, the RL community branched out (just as I did) into other methods such as "policy gradient methods" (my "policy function approximations" or PFA), upper confidence bounding (a form of "cost function approximation" or CFA), the original Q-learning (which produces a policy based on "value function approximations" or VFA), and finally

Monte Carlo tree search (a policy based on "direct lookahead approximations" or DLA). All of these methods are found in the second edition of Sutton and Barto's landmark book *Reinforcement Learning: An introduction*, but only as specific methods rather than general classes. This book takes the next step and identifies the general classes.

This evolution from one core method to all four classes of policies is being repeated among other fields that I came to call the "jungle of stochastic optimization." Stochastic search, simulation-optimization, and bandit problems all feature methods from each of the four classes of policies. Over time, I came to realize that all these fields (including reinforcement learning) were playing catchup to the grandfather of all of this work, which is optimal control (and stochastic control). The field of optimal control was the first to introduce and seriously explore the use of value function approximations (they call these cost-to-go functions), linear decision rules (a form of PFA), and the workhorse "model predictive control" (a great name for a simple rolling horizon procedure, which is a "direct lookahead approximation" in this book). I also found that my modeling framework was closest to that used in the optimal control literature, which was the first field to introduce the concept of a transition function, a powerful modeling device that has been largely overlooked by the other communities. I make a few small tweaks such as using state S_t instead of x_t, and decision x_t (widely used in the field of math programming) instead of u_t.

Then I introduce one big change, which is to maximize over all four classes of policies. Perhaps the most important innovation of this book is to break the almost automatic link between optimizing over policies, and then assuming that we are going to compute an optimal policy from either Bellman's equation or the Hamilton-Jacobi equations. These are rarely computable for real problems, which then leads people to assume that the natural next step is to approximate these equations. This is simply false, supported by decades of research where people have developed methods that do not depend on HJB equations. I recognize this body of research developing different classes of policies by making the inclusion of all four classes of policies fundamental to the original statement of the optimization problem above.

It will take some time for people from the different communities to learn to speak this common language. More likely, there will be an adaptation of existing modeling languages to this framework. For example, the optimal control community could keep their notation, but learn to write their objective functions as I have above, recognizing that the search over policies needs to span all four classes (which, I might point out, they are already using). I would hope that the reinforcement learning community, which adopted the notation for discrete action a, might learn to use the more general x (as the bandit community has already done).

I have tried to write this book to appeal to newcomers to the field, as well as people who already have training in one or more of the subfields that deal with decisions and uncertainty; recognizing these two broad communities was easily the biggest challenge while writing this book. Not surprisingly, the book is quite long. I have tried to make it more accessible to people who are new to the field by marking many sections with an * as an indication that this section can be skipped on a first-read. I also hope that the book will appeal to people from many application domains. However, the core audience is people who are looking to solve real problems by modeling applications and implementing the work in software. The notation is designed to facilitate writing computer programs, where there should be a direct relationship between the mathematical model and the software. This is particularly important when modeling the flow of information, something that is often overlooked in mainstream reinforcement learning papers.

Warren B. Powell

Princeton, New Jersey
August, 2021