Practical Applications of Large-Scale Stochastic Control for Learning and Optimization

by

Eli Gutin

M.Eng, Imperial College London (2012)

Submitted to the MIT Sloan School of Management
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Operations Research

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2018

Author .....
MIT Sloan School of Management
July 20th, 2018

Certified by .....
Vivek F. Farias
Patrick J. McGovern (1959) Professor
Sloan School of Management
Thesis Supervisor

Accepted by .....
Dimitris Bertsimas
Boeing Leaders for Global Operations Professor
Sloan School of Management
Co-Director, Operations Research Center

Practical Applications of Large-Scale Stochastic Control for Learning and Optimization

by

Eli Gutin

Submitted to the MIT Sloan School of Management
on July 20th, 2018, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Operations Research

Abstract

This thesis explores a variety of techniques for large-scale stochastic control. These range from simple heuristics that are motivated by the problem structure and are amenable to analysis, to more general deep reinforcement learning (RL) which applies to broader classes of problems but is trickier to reason about.

In the first part of this thesis, we explore a less known application of stochastic control in Multi-armed bandits. By assuming a Bayesian statistical model, we get enough problem structure so that we can formulate an MDP to maximize total rewards. If the objective involved total discounted rewards over an infinite horizon, then the celebrated Gittins index policy would be optimal. Unfortunately, the analysis there does not carry over to the non-discounted, finite-horizon problem. In this work, we propose a tightening sequence of 'optimistic' approximations to the Gittins index. We show that the use of these approximations together with the use of an increasing discount factor appears to offer a compelling alternative to state-of-the-art algorithms. We prove that these optimistic indices constitute a regret optimal algorithm, in the sense of meeting the Lai-Robbins lower bound, including matching constants.

The second part of the thesis focuses on the collateral management problem (CMP). In this work, we study the CMP, faced by a prime brokerage, through the lens of multi-period stochastic optimization. We find that, for a large class of CMP instances, algorithms that select collateral based on appropriately computed asset prices are near-optimal. In addition, we back-test the method on data from a prime brokerage and find substantial increases in revenue.

Finally, in the third part, we propose novel deep reinforcement learning (DRL) methods for option pricing and portfolio optimization problems. Our work on option pricing enables one to compute tighter confidence bounds on the price, using the same number of Monte Carlo samples, than existing techniques. We also examine constrained portfolio optimization problems and test out policy gradient algorithms that work with somewhat different objective functions. These new objectives measure the performance of a projected version of the policy and penalize constraint violation.

Thesis Supervisor: Vivek F. Farias
Title: Patrick J. McGovern (1959) Professor
Sloan School of Management

Acknowledgments

First of all, I want to aknowledge my advisor Vivek Farias for everything he taught me. Vivek showed me, by example, how to correctly approach challenging technical problems and think about them in a natural, intuitive way. He also showed me how to generally become a more confident person and technical speaker. I am certain that this will prove immeasurably useful in my future career, in business and in life.

A large chunk of my PhD work would not have been successful without my fruitful collaboration with Hui Chen, Bart Coppens and Madhu Subbu. Each one of them was an amazing sounding board for my ideas while I was learning about collateral management. I enjoyed hanging out with Bart during his visits to Boston. It was also fun having Madhu visit and I'm pleased I got a chance to visit him and his team at Credit Suisse in NYC.

I was also fortunate to have worked with Ciamac Moallemi. Together with Vivek, the three of us tackled cutting-edge research problems in reinforcement learning. Ciamac has been a patient and fantastic listener. I've come to appreciate his clarity of thought, breadth of knowledge and easygoing nature.

Next, I want to thank my previous academic advisors, Daniel Kuhn and Wolfram Wiesemann. My interest in Operations Research really began after I took their classes at Imperial College as an undergraduate student. I also had the pleasure to write my Master's thesis under their supervision, which later led to our joint publication. Daniel and Wolfram always believed in and encouraged me, to an extent I could not otherwise fathom.

The ORC has been a wonderful community and I feel lucky to have been a part of it. I will always remember the parties, retreats and events, both academic and social, that we had organized. Everyone I met there was extremely friendly, down to earth, yet frighteningly smart!

Last and by no means least, I want to thank my parents, Irina and Gregory, who sacrificed a lot to get me to this position. I am forever indebted to them for their support and for instilling a great work ethic in me.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

A wide range of optimization problems in business, engineering and operations management can be formulated as stochastic dynamic control problems in discrete time. What all these problems have in common are two primary features: first is the need to control some system dynamically over time, and the second is the fact that the system's environment is affected by randomness. The stochastic dynamic control framework simultaneously models both these features. Unfortunately, what prevents the framework from being used more often in practice is that solving the resulting optimization problems is practically impossible given their computational cost. In particular, the solution runtimes and memory requirements would be exponential in the number of problems variables. This fundamental and pervasive challenge is infamously known as Bellman's 'curse of dimensionality'.

Usually, the most practical way to tackle a given real-world dynamic problem is to assume all the relevant data for it is certain and known a priori. If we also assume the problem's mathematical formulation is convex and/or linear, then one can find a globally optimal solution efficiently. Moreover, with the technology and solvers available nowadays, it might be possible to scale algorithms to enormous instances involving millions of variables and constraints. Despite this benefit from tractability, using static, deterministic formulations leaves a several, vital things on the table. To start with the most obvious disadvantage, ignoring randomness means we are sacrificing optimality at least to some extent. Second of all, dynamic formulations

capture the controller's ability to take recourse decisions after some of the uncertainty is resolved, while with static formulations this is impossible. Finally, if we rigidly stick to the solution prescribed by a static algorithm, then we may end up violating constraints so that our implemented controls are ultimately infeasible.

One way to deal with the aforementioned issues is to use a different approach from the one just discussed. Specifically, during the course of controlling a system, one would repeatedly re-solve a new deterministic problem in every period that approximates the remaining optimization model – this is usually referred to as Model Predictive Control (MPC) Garcia et al. [1989]. By using the MPC approach, the implemented controls are guaranteed to satisfy any problem constraints and the solution obtained is usually better than what would have been achieved by a static policy. However, there is the additional overhead of needing to solve a new optimization problem in every period, and some of the optimality gap remains.

Another approach to solving stochastic problems is robust optimization [Bertsimas and Sim, 2004, Ben-Tal et al., 2009, Bertsimas et al., 2011], wherein one optimizes over worst case realizations of the uncertain data within some prescribed uncertainty set of plausible realizations. Extending the robust optimization idea to dynamic problems often requires searching over a space of affine policies [Bemporad et al., 2003] by formulating a global, convex robust optimization problem over the policy parameters. The general issue with robust optimization is that it gains computational tractability by providing conservative solutions. Sometimes this is acceptable, however if one specifically aims for optimal solutions, in the truest sense, while relying on a specific stochastic model, then robust optimization might be settling for overly conservative and hence poor solutions. To our knowledge, these three aforementioned frameworks (determinstic approximation, MPC and robust optimization) are applied most often and our main goal will be to explore alternatives.

In this thesis, we aim to further the understanding of how large-scale stochastic control could be approached directly and in a principled way. We tackle this broad goal on two fronts. First, we examine the question from the lens of a specific complex application of collateral management, which is high-dimensional and involves complex

interactions with the environment. This will be the focus of Chapter 3. Secondly, we study and make contributions to deep reinforcement learning. Certain deep RL methods constitute potential, general-purpose techniques for addressing swathes of control problems with similar characteristics, such as those involving discrete time and partially linear-dynamics (Chapter 4). Advances in hardware, the fields of machine learning and neural networks in recent years, have uncovered new possibilities for these deep RL techniques and their practical applications. Chapter 4 specifically focuses on the application of certain deep RL methods to finance, since this area offers a host of crucial and interesting problems. We also study the problem of option pricing that falls under the class of control problems known as optimal stopping. Finally, in this thesis we focus on a less-known and unconventional use of stochastic control for the case of multi-armed bandits in Chapter 2. Most algorithms designed for the multi-armed bandit problem are ad-hoc heuristics, which did not emerge from a principled analysis of the underlying problem. On the other hand, by framing the problem as one about Bayesian learning and thereby giving it enough structure, the decision problem behind pulling an arm becomes a stochastic control one, and hence one that we need to address via clever, efficient algorithms.

We will now motivate and formulate the general, finite-horizon stochastic control problem, and then see, via a few salient example, its broad applicability to a myriad of problems[1].

1.1 Optimal stochastic control

We consider here a dynamical system, which we will control over a sequence discrete time periods $\mathcal{T} = \{0, 1, \ldots, T\}$. The system's state in period t is written as $x_t \in \mathcal{X}$ where $\mathcal{X}$ is the space of all possible states. An agent observes the current state x_t and then decides on a control to apply, $u_t \in \mathcal{U}$, from the set of admissible controls $\mathcal{U}_t$ at that time. Given the state and control applied, the agent both observes and *earns* a real-valued reward $r_t(x_t, u_t)$. State evolves according to dynamics defined through

[1]To keep things simple we won't delve into the infinite horizon problem here.

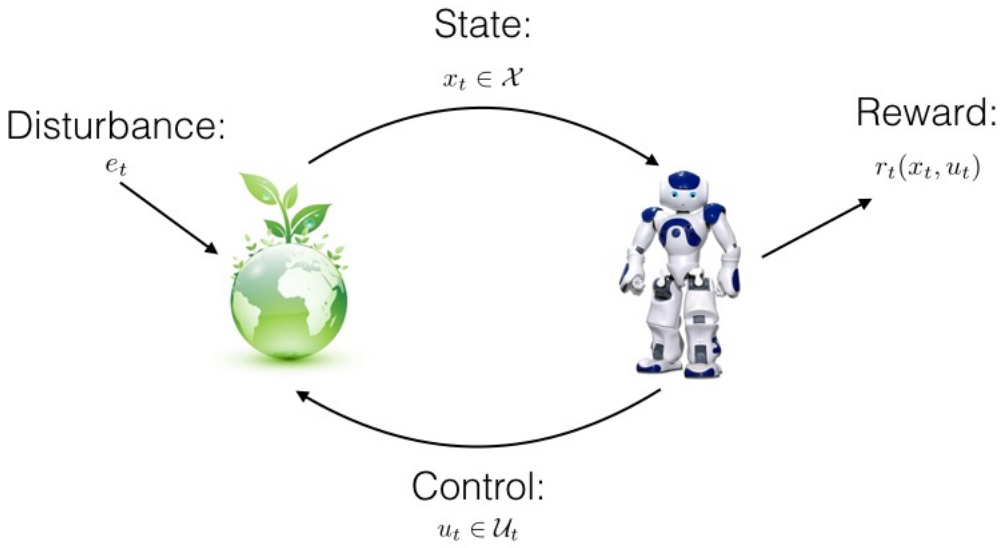

Figure 1-1: Stochastic control problem involving an agent (right) and its environment (left).

to the following equations:

$$x_{t+1} = f_t(x_t, u_t, e_t), \quad 0 \leq t < T$$

where f_t is a sequence of functions describing state transitions. The inputs to these functions are the current current state x_t, the control $u_t \in \mathcal{U}_t$ and disturbance $e_t \in \mathcal{E}$. We model the disturbance as some random variable. A condensed illustration of the interplay between states, controls and disturbances is given in Figure 1-1.

To define the information available to the agent, we let

$$\mathcal{F}_t = \sigma(x_0, u_0, e_0, \ldots, x_{t-1}, u_{t-1}, e_{t-1}, x_t)$$

be the σ-algebra generated by past controls, disturbances and both current and previous states observed in period t. The control at every time t must be measurable with respect to the σ-algebra $\mathcal{F}_t$, i.e. the control can only be made based on the current state and the agent's history of observations. We refer to the sequence of random controls as a policy (or what is sometimes called a decision rule), and we denote this policy by $\mathbf{u} = (u_0, u_1, \ldots, u_T)$.

Since our general problem fundamentally contains uncertainty, we need to formulate its objection function in terms of some deterministic quantity. We could optimize for the expected total rewards, variance in the total rewards, or some such metric, but we opt for the former since it's the most common and can handle (to some extent via utility functions) the notion of risk. The full optimization problem is given as

$$\begin{aligned} \underset{\mathbf{u}}{\text{maximize}} \quad & \mathbb{E}_{\mathbf{u}}\left[\sum_{t=0}^{T} r_t(x_t, u_t) \,\middle|\, x_0 = x\right] \\ \text{subject to} \quad & x_{t+1} = f_t(x_t, u_t, e_t), \qquad t = 0, 1, \ldots, T-1 \end{aligned} \tag{1.1}$$

where the expectation is defined with respect to sample paths generated by the policy $\mathbf{u}$, the initial state is given as deterministic value $x \in \mathcal{X}$, and the constraints must be satisfied in an almost sure sense. In general we denote the optimal value of such problems as $J_0^\star(x)$, since the initial state is x and we are controlling the system from period 0 onward.

In its nominal form, this problem is simply too hard to solve in high dimensions, and we illustrate why this is so in the following special case where we make a couple of assumptions. The first assumption we're going to make is that e_t is statistically independent of any the previous realizations $e_0, \ldots, e_{t-1}$ before it. One can show that this results in the problem being Markovian. We'll also assume that $\mathcal{U}_t$ is a finite set. In that case, we are dealing with what's otherwise known as a Markov Decision Problem (MDP). One can show that the optimal control at every state is only a function of the current time period t and the present state x_t. In fact, a way to find an optimal policy is to solve Bellman's equations, i.e. to find a sequence of *value functions* $J_t^\star$ that satisfy

$$J_t^\star(x) = \begin{cases} \max_{u \in \mathcal{U}_t} \left\{ r_t(x,u) + \mathbb{E}\left[J_{t+1}^\star(f_t(x,u,e_t)) \right] \right\}, & t \leq T \\ 0, & t = T+1 \end{cases}, \forall t \in [T{+}1], x \in \mathcal{X}.$$

One algorithm that finds these value functions is called value iteration, or backwards induction. Having found such a sequence of value functions, a greedy policy defined

as

$$u_t^\star(x) = \begin{cases} \text{argmax}_{u \in \mathcal{U}_t} \left\{ r_t(x,u) + \mathbb{E}\left[J_{t+1}^\star(f_t(x,u,e_t)) \right] \right\}, & t < T \\ \text{argmax}_{u \in \mathcal{U}_t} \, r_t(x,u), & t = T \end{cases}$$

can then be shown to be optimal for (1.1). If we closely inspect these equations, we can see that to find the value functions, we need to at the very least enumerate all possible states in $\mathcal{X}$. If this state space has more than, say 3 or 4 dimensions, then its size quickly explodes. This is the intuitive explanation of why dynamic control problems are intractable. Moreover, even if state-space explosion was not the main difficulty, we'd still need to compute expectations over $J_{t+1}^\star(f_t(x,u,e_t))$ as shown above, which is also onerous if $\mathcal{E}$ the space of possible disturbances is large.

1.2 Motivating examples

The simple-enough looking framework given in Problem (1.1), can capture a myriad of useful problem formulations. We outline some examples of concrete applications in this section.

1.2.1 Rocket control

We start with a somewhat two-dimensional, toy problem whose purpose is to illustrate the main ideas of stochastic control theory. It is a variation on an example from Chapter 1 in Bertsimas and Tsitsiklis [1997]. This example also lets us touch on the Linear Quadratic Regulator (LQR), which appears several times in the thesis.

Consider a rocket positioned on the ground that needs to travel a vertical distance of D kilometers in the air, within a certain window of time. Our job is to control the rocket's thrust throughout its ascent. At time t, we let y_t be the rocket's vertical position, v_t its velocity and a_t its acceleration. To keep the notation simple, we take a_t to be the 'control'. This could be justified since acceleration is proportional to the rocket's thrust, which produces the force necessary to counteract gravity and drag

(see Figure 1-2)[2]. The other components, y_t and v_t, both represent state components.

By using a discretized model of time with a unit increments, we suppose the state dynamics are linear according to

$$\begin{aligned} y_{t+1} &= y_t + v_t + e_{1,t} && t = 0, \ldots, T-1 \\ v_{t+1} &= v_t + a_t + e_{2,t} && t = 0, \ldots, T-1 \end{aligned}$$

where $e_{1,t}$ and $e_{2,t}$ are Gaussian error terms with unit variance. We also assume that $y_0 = v_0 = 0$. The error terms might represent systematic uncertainty in our model, or just noise in the physical environment. We assume that the terminal period is T, and that rocket should be at (or close to) the desired position by then. Thus keeping in line with our framework, we can rewrite some notation in terms of the general setup, as $x_t \triangleq (y_t, v_t)$ and $e_t \triangleq (e_{1,t}, e_{2,t})$, $u_t \triangleq a_t$, and then we have that the state transition function is

$$f_t(x_t, u_t, e_t) = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} x_t + \begin{pmatrix} 0 \\ 1 \end{pmatrix} u_t + e_t.$$

Suppose that our goal is to minimize total fuel consumption plus some measure of the rocket's distance from its desired position at time T. To do so, we need to formulate a good reward function $r_t(x_t, u_t)$. In some rough model, we could say that fuel consumption is proportional to the absolute acceleration $-|u_t|$, so that a sensible candidate reward function might be

$$r_t(x_t, u_t) = \begin{cases} -|u_t| & t < T \\ -\alpha \left| x_{1,t} - D \right| & t = T \end{cases}$$

where α is some scalar that trades off the two objectives of fuel consumption and final distance away from D. What is special about this problem however is that if we

[2]This is based on Newtonian law that states $F = ma$

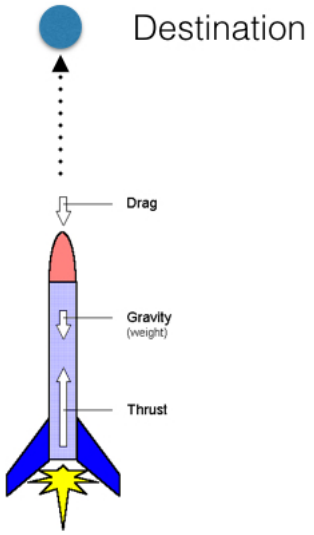

Figure 1-2: Rocket control problem

were to replace the modulus above with the squared value, i.e. define r_t as

$$r_t(x_t, u_t) = \begin{cases} -u_t^2 & t < T \\ -\alpha \left(x_{1,t} - D\right)^2 & t = T \end{cases}$$

then this problem would admit a closed form solution. This is a basic example of an LQR problem, which we will revisit later. Interestingly enough, this is very similar in nature to a portfolio optimization problem given in Gârleanu and Pedersen [2013].

1.2.2 Option pricing

Optimal control has famously been applied to option pricing. In general, financial derivatives are contracts whose payoff depends on the price of one more underlying assets: these could be stocks, bonds, commodities, among others. The simplest example is that of European call option on a stock, which gives the holder the right (but not the obligation) to buy the stock at a specific expiry date in the future at a pre-agreed strike price. There are different types of option contracts, which confer different rights: for example, American options include more flexibility in that the holder is allowed buy/sell the stock once at any time before the contract expires.

These derivatives are financial products that are traded daily in a variety of markets and on exchanges. Financial institutions typically hold billions of dollars worth of positions in option contracts, making it especially critical to accurately value them, and do so in a way that's computationally efficient.

If we are to model the simplest option pricing problem with our framework, in discrete-time, we would form the state as a vector $x_t = (p_t, y_t)$ whose first component is the price of the underlying stock, and the second component y_t is binary variable indicating whether the option has yet to be exercised still. That is, $y_t = 0$ if the option has been exercised and 1 otherwise. Our control will be binary valued, so that $u_t = 1$ if the option is exercised at time t and 0 otherwise. The reward is then $r_t(x_t, u_t) = r_t(p_t, y_t, u_t) = y_t u_t (p_t - K)^+$ where K is the strike price. We are going to define the dynamics in such a way that once the option is exercised, $y_{t+1}, y_{t+2}, \ldots$ gets set to zero, meaning there is no more payoff to be earned. In particular, the state transition equation is

$$\begin{aligned} p_{t+1} &= g(p_t, e_t) \\ y_{t+1} &= (1 - u_t) y_t, \end{aligned}$$

where $g(.,.)$ is some function that defines price dynamics and the initial state is $x_0 = (p_0, 1)$. Calculating the option price then boils down to finding $J_0^\star(x)$ for this problem. In fact, the type of problem discussed just now is much more general and often referred to as optimal stopping.

For the simplest options involving only a few underlying assets, and where it's assumed that asset prices follow a Geometric brownian motion, it becomes computationally easy to calculate the option price, via, say, the Binomial lattice method or the Black-Scholes formula. As soon as more assets are introduced and the problem becomes high-dimensional, we need to consider different techniques such as those appearing later in the thesis.

1.2.3 Multi-armed bandits

Multi-armed bandits, as described here, is a control problem with a special structure[3]. Consider N bandits, which one can think of as N projects that an agent can invest in over T periods. Each bandit is associated with a discrete state space S_i. At time t, the ith bandit is in the state $x_{i,t} \in S_i$, which one can think of as the state for that project. Let us denote the state of all of the bandits at time t by the tuple $x_t = (x_{1,t}, \ldots, x_{N,t})$, which is the overall state in our problem. For this reason, the overall state space is $\mathcal{X} = \prod_{i=1}^{N} S_i$, and we assume that the initial state is given by a known value $x \in \mathcal{X}$. At any given time, the control is an integer $u_t \in [N] = \{1, \ldots, N\}$ for which of the bandits the agent 'pulls' or activates, or equivalently which of the N projects the agent invests in.

If the agent pulls the ith bandit at time t, then it earns a reward $r_t(x_t, i) = r_i(x_{i,t})$, where $r_i(.)$ is some reward specific to the ith bandit, which is a function of its state. Subsequently, upon pulling the ith arm, the system's state evolves in the next period according to the equation

$$x_{t+1} = f_t(x_t, u_t, e_t) = \begin{pmatrix} x_{1,t} \\ \vdots \\ f_i(x_{i,t}, e_{i,t}) \\ \vdots \\ x_{N,t}, \end{pmatrix}$$

meaning the states of all the bandits, except for the ith one, stay the same (are frozen). Meanwhile the ith bandit's state changes according to the transition function $f_i(.,.)$. As usual, our goal is to maximize the total sum of expected rewards.

As we might expect, the problem in this form is intractable. However, if we assume that we're dealing with an infinite horizon, $T = \infty$, and that rewards are

[3]This example describes the MDP formulation of the multi-armed bandit problem, which is somewhat different to the one tackled in Chapter 2.

geometrically discounted over time, such that the objective is

$$\mathbb{E}_{\mathbf{u}}\left[\sum_{t=0}^{\infty}\gamma^t r_t(x_t,u_t)\;\middle|\;x_0=x\right],$$

then the optimal policy is substantially easier to compute. In fact, the optimal policy is given by the Gittins index rule, and this will be one of the main themes in Chapter 2. This fact means that the multi-armed bandit problem can be solved to optimality by addressing, essentially, a sequence of easier 1-dimensional problems, as opposed to a single N-dimensional one. Ultimately, this alternative algorithm with Gittins indices enjoys computational costs that are exponentially lower than a brute force method that solves Bellman's equations.

1.3 Organization of This Thesis

Below we summarize briefly the contents of each chapter in the thesis:

- **Chapter 2.** In this chapter, we develop a novel algorithm for the Bayesian multi-armed bandit (MAB) problem. At every step t, our policy approximately solves an infinite horizon discounted variant of the multi-armed bandit problem, equivalently the Gittins Index problem, where the arms' initial states take on their present values and the discount factor is $1-1/t$, and plays the arm with the highest current index. We prove that our policy's regret is $\mathcal{O}(\log n)$ where n is the time horizon. Moreover, in the case where rewards are binary, we prove that our policy is asymptotically optimal, in the sense of meeting the lower bound of Lai and Robbins [1985]. Numerical experiments demonstrate that the Bayesian regret from this approach outperforms state of the art algorithms.

- **Chapter 3.** In this chapter, we consider the stochastic control problem faced by a Prime Broker, a key agent in the securities lending market. We start by analyzing common decisions a Prime Broker needs to make, which consist of selecting collateral to hypothecate from clients and determining, in general,

what pool of assets to hold in its inventory in order to maximize revenues and improve operational efficiency. To address these questions, we model the Prime Broker's asset allocation decisions as a multi-period stochastic problem. We propose a simple framework for designing algorithms that are provably near-optimal and overcome the 'curse of dimensionality' inherent to such a problem.

The framework we propose hinges on the computation of asset prices as collateral. We find our methods are practical, efficient to implement, offer substantial performance gains over existing ad-hoc approaches currently used in industry. Furthermore, we provide numerical experiments by firstly backtesting our algorithms on data from Credit Suisse and, secondly, running them on simulations. By benchmarking our policy against existing ones, we see increases in profit of 5-10% at the same computational cost.

- **Chapter 4.** The focus in this chapter is applying deep learning techniques to stochastic control, also sometimes as deep reinforcement learning (RL). We explore model-based RL in the context of problems with clearly defined and special structures (such as minimally small action spaces like on optimal stopping or linear dynamics). The aim is to see if we can gain any insight from certain classes of problems that can inform the design of RL algorithms. Here we will focus on two problem types:

  - **Optimal stopping:** We explore two heuristics for deriving confidence intervals on option price, both of which use deep learning. The first of these, involves adapting the Longstaff-Schwartz algorithm by fitting a deep neural network at every iteration, as opposed to solving a linear regression problem. We see relative improvements in the lower bound of around of around 400 bps compared to the usual Longstaff-Schwartz algorithm, and 200 bps compared to the best-known lower bound from Pathwise Optimization method Desai et al. [2012].

    The second of these techniques is for deriving upper bounds based on the martingale duality technique [Rogers, 2002]. We propose an alternative

continuous representation, which is more efficient to work with, yields a 30 bps improvement in the upper bound and avoids the need to provide basis functions to the algorithm (as is otherwise common in ADP).

  - **Quasi-linear, convex control:** We develop policy gradient algorithms that work on constrained problems with partially linear state dynamics and convex rewards. Through realistic, large-scale benchmark problems, we demonstrate that the policies trained using our methods outperform, by a huge margin, existing heuristics – sometimes as much as 50% or more. Moreover, in some non-trivial example problem we see that our method achieves within 1% of the optimal value. These impressive results motivate us to analyze some aspects of these problems and provide open research problems.

- **Chapter 5.** In this final chapter, we conclude with the main messages of this thesis and motivate some open research problems and future challenges.

Chapter 2

Optimistic Gittins Indices

2.1 Introduction

The Multi-Armed Bandit (MAB) problem is perhaps the simplest example of a learning problem that exposes the tension between exploration and exploitation. In its simplest form, we are given a collection of random variables or 'arms'. By adaptively sampling these random variables, we seek to eventually sample consistently from the random variable with the highest mean. This is typically formalized by asking that we minimize cumulative 'regret'; a notion we make precise in a later section.

Recent years have seen a resurgence of interest in *Bayesian* algorithms for the MAB problem. In this variant of the MAB problem, we are endowed with a prior on arm means, and a number of algorithms that exploit this prior have been proposed and analyzed. These include Thompson Sampling [Thompson, 1933], Bayes-UCB [Kaufmann et al., 2012], KL-UCB [Garivier, 2011], and Information Directed Sampling [Russo and Van Roy, 2014]. The ultimate motivation for these algorithms appears to be the empirical performance they offer. Specifically, these Bayesian algorithms appear to incur smaller regret than their frequentist counterparts such as the UCB algorithm proposed by Auer et al. [2002], even when regret is measured in a frequentist sense. This empirical evidence has, very recently, been reinforced by theoretical performance guarantees. For instance, it has been shown that both Thompson sampling and Bayes-UCB enjoy upper bounds on frequentist regret that

match the Lai-Robbins lower bound [Lai and Robbins, 1985]. Interestingly, even amongst the various Bayesian algorithm proposed there appears to be a wide range in empirical performance. For instance, empirical evidence presented in Russo and Van Roy [2014] suggests that the IDS algorithm offer a substantial improvement in frequentist regret over Thompson sampling and the Bayes-UCB algorithm, among others. The former algorithm does not however enjoy the optimal data dependent frequentist regret bounds that the latter two do. Perhaps more importantly, these algorithms also vary substantially in their design (as opposed to being variations on a theme).

Now a prior on arm means endows us with the structure of a Markov Decision Process (MDP) and none of the Bayesian algorithms alluded to above exploit this structure. This is especially surprising in light of the celebrated Gittins Index Theorem. That breakthrough result proved the optimality of a certain index policy for a *horizon dependent* variant of the Bayesian MAB. Specifically, imagine that we cared about the expected (Bayes) regret incurred over an exponentially distributed horizon, where the mean horizon length is known to the algorithm designer. This problem is nominally a high dimensional MDP. Gittins, however, proved that a simple to compute index rule was optimal for this task resolving a problem that had remained open for several decades [Gittins, 1979]. Why does the Gittins Index Theorem not immediately help resolve the design of an optimal algorithm for the variant of the Bayesian MAB problem that is the subject of the approaches discussed in the preceding paragraph? As we will discus more carefully in our literature review, this is certainly not from lack of research effort [Lattimore, 2016]. In fact, one must deal with several substantial challenges:

1. Dependence on Horizon: The notion of regret optimality as popularized by Lai and Robbins [1985] is 'anytime'. Colloquially, this can be thought of as follows: we desire an algorithm that performs well for *any* time horizon. This fact is fundamentally at odds with Gittins' variant of the MAB problem that (via a discount factor) effectively specifies a (exponentially distributed) horizon. Gittins' result is intimately connected to this choice of horizon; even seemingly

minor changes appear to render the problem intractable. For instance, it is known that a Gittins-like index strategy is sub-optimal for a fixed, finite-horizon [Berry and Fristedt, 1985]. Algorithms for other notions of optimality that one may reasonably conjecture are better aligned with 'anytime' regret optimality (such as, say, Cesaro-overtaking optimality) are similarly elusive [Katehakis et al., 1996].

2. Computation: Separate from the issues made in the previous point, consider the task of computing a Gittins index at every point in time. The computation of a Gittins index can be reduced to the solution of a certain infinite horizon stopping problem. For the Bayesian MAB, the state space for this problem must describe all possible posteriors one may encounter on a given arm. Assuming conjugate priors, one may hope for a finite dimensional state space, but tractable computation will typically call for some form of state-space truncation. This computation is far more onerous than any of the aforementioned indices. Furthermore, it is reasonable to conjecture that as time progresses one may require increasingly more accurate estimates of the Gittins index, which further complicates computation, and calls into question the correctness of a naive state-space truncation scheme.

Against this backdrop, in this chapter we make the following contribution:

We show that picking arms according to a certain tractable approximation to their Gittins index, computed for a time dependent discount factor we characterize precisely, constitutes a regret optimal bandit policy. The resulting index rule is both simple to compute and in computational experiments appears to outperform state-of-the-art bandit algorithms by a material margin.

In greater detail, we outline our contributions as follows:

1. Optimistic Approximations: We propose a sequence of 'optimistic' approximations to the Gittins index. These optimistic approximations can be interpreted

as providing a tightening sequence of upper bounds on the optimal stopping problem defining a Gittins index, yielding the index itself in the limit. The computation associated with the simplest of these approximations is no more burdensome than the computation of indices for the Bayes UCB algorithm, and several orders of magnitude faster than the best performing alternative from an empirical perspective (the IDS algorithm).

2. Regret Optimality: We establish that an arm selection rule that is greedy with respect to any optimistic approximation to the Gittins index achieves optimal regret in the sense of meeting the Lai-Robbins lower bound (including matching constants) for the canonical case of Beta-Bernoulli bandits. A crucial ingredient required for this scheme to work is that as time progresses, the discount factor employed in computing the index must be increased at a certain rate which we characterize precisely. This implicitly resolves the challenge of horizon dependence.

3. Empirical Performance: We show empirically that even the simplest optimistic approximation to the Gittins index outperforms the state-of-the-art incumbent schemes discussed in this introduction by a non-trivial margin. Our empirical study is careful to recreate several ensembles of problem instances considered by previous authors (including a particularly computationally intensive study by Chapelle and Li [2011] that prompted the reexamination of the Thompson sampling algorithm in recent years). The margin of improvement we demonstrate increases further as one employs successfully tighter optimistic approximations, at the cost of computational effort.

In summary, we propose a new index rule for the Baysian MAB problem that employs Gittins indices in a novel way. This new index rule enjoys the strongest possible data-dependent regret guarantees while also offering excellent empirical performance.

2.1.1 Relevant Literature

We organize our literature review around the primary topics that this chapter touches on. The study of exploration-exploitation problems is vast, even if it is restricted to a problem with a finite number of arms. Consequently, our review will be focused on stochastic, non-contextual, versions of the MAB problem. Even with this restriction, the literature remains vast, and so we focus on papers that are either seminal in nature or particularly relevant to our own work; this review is by no means comprehensive with respect to the MAB problem.

Regret optimality and the bandit problem: Robbins [1952] motivated the study of the MAB problem and left open questions on how to design effective policies. Since then Lai and Robbins [1985] proved a cornerstone result, namely an asymptotic lower bound on regret that any consistent strategy incurs. The same paper proposes an upper-confidence bound (UCB) algorithm that asymptotically achieves the lower bound. Computationally efficient UCB algorithms were developed by Agrawal [1995] and Katehakis and Robbins [1995]. Later, Auer et al. [2002] and Audibert and Bubeck [2010] proved finite time regret bounds for UCB algorithms and demonstrated ways to tune them in order to improve performance. Garivier [2011] and Maillard et al. [2011] have proposed other UCB-type algorithms where the confidence bounds are calculated using the KL-divergence function. Those authors provide a finite-time analysis and their algorithms are shown to achieve the Lai-Robbins bound.

Bayesian bandit algorithms: Another powerful approach to bandit problems is to work with a Bayesian prior to model one's uncertainty about an arm's expected reward. Lai [1987] proves an asymptotic lower bound on Bayes' risk and develops a horizon-dependent algorithm that achieves it. Thompson Sampling [Thompson, 1933], one of the earliest algorithms proposed for the MAB problem, is in fact a Bayesian one. Empirical studies by Chapelle and Li [2011] and Scott [2010] highlight Thompson Sampling's hugely superior performance over some UCB algorithms even when the prior is mismatched. A series of tight regret bounds for Thompson Sampling have been established by Agrawal and Goyal [2012, 2013] and Kaufmann et al. [2012].

These authors have shown Thompson sampling to be regret optimal for the canonical Beta-Bernoulli bandit. Recently, Korda et al. [2013] generalized the aforementioned results to bandit problems where the arm distributions belong to a one dimensional exponential family. Interestingly enough, Robbins [1952] seems to have been unaware of Thompson Sampling and its effectiveness in the non-Bayesian setting.

Several other Bayesian algorithms exist. Kaufmann et al. [2012] propose Bayes UCB, which they show is competitive with Thompson Sampling. The main idea behind Bayes UCB is to treat quantiles of the arm's prior as an upper confidence bound and let the quantile grows at some pre-specified rate. Russo and Van Roy [2014] propose Information Directed Sampling (IDS), an algorithm that exploits information theoretic quantities arising from the prior distributions over the arms. In simulations, IDS is shown to dominate many of the aforementioned algorithms, including Thompson Sampling, Bayes UCB and KL-UCB. In our empirical investigation, we will see that IDS is the closest competitor to the approach we propose here (we recreate the experiments from Russo and Van Roy [2014]).

Gittins index and its approximations: There is another stream of literature that models the MAB problem as an MDP. For the case of two arms, where one arm's reward is deterministic, Bradt et al. [1956] showed that for this one-dimensional DP, an index rule is an optimal strategy. When the objective is to maximize the infinite sum of expected *discounted* rewards Gittins [1979] famously showed the optimality of an index policy. The Gittins index is similar to that proposed by Bradt et al. [1956] but takes discounting into account. Several alternative proofs of Gittins' result are available; see for example [Tsitsiklis, 1994, Weber et al., 1992, Whittle, 1980] and [Bertsimas and Niño-Mora, 1996]. These alternative proofs also provide illuminating alternative interpretations of the Gittins index.

Computing the Gittins index can be an onerous task, especially when the state space corresponding to posterior sufficient statistics is large or high dimensional. As such, approximations to the index have been proposed by Yao et al. [2006], Katehakis and Veinott Jr [1987] and Varaiya et al. [1985]; see [Chakravorty and Mahajan, 2013] for a survey. This chapter also relies on Gittins index approximations and we develop

simple, general ones that enable our algorithm to be regret optimal.

Finally, we note that others have contemporaneously attempted to leverage the Gittins index in the construction of a Bayesian MAB algorithm. For instance, Kaufmann [2016] considers a variety of heuristics based on a finite horizon version of the Gittins index (essentially, the index proposed by Bradt et al. [1956]), and shows promising empirical results. Lattimore [2016] analyzes the regret under a similar index and shows it to be logarithmic for a *fixed* horizon. Unfortunately, the index policies studied in both [Kaufmann, 2016] and [Lattimore, 2016] *require a-priori knowledge of a horizon*. As such this does not yield an index rule that works for any sufficiently large horizon, but rather one that only works for a fixed pre-specified horizon. In fact, such schemes cannot be expected to work well for time horizons other than the pre-specified horizon determining the index. In contrast, we seek to provide a compelling alternative to the host of state-of-the-art 'anytime' regret optimal index rules discussed heretofore.

2.2 The Optimistic Gittins Index Algorithm

This section introduces the notion of an optimistic Gittins index, and presents an algorithm for the MAB problem that we will subsequently show is optimal in that it achieves the Lai-Robbins lower bound. We will begin with reviewing the Gittins index theorem for the discounted infinite horizon bandit problem and show that one cannot expect the use of the index from that well known result to yield a regret optimal policy for the MAB problem. We then show that the use of the Gittins index in concert with an increasing discount factor yields poly-logarithmic Bayesian regret. This coarse result motivates the discount factor schedule we eventually propose. Finally, we present a series of 'optimistic' approximations to the Gittins index with the view of minimizing the computational burden of index computation. Putting these ingredients together yields the optimistic Gittins index algorithm that is the subject of this chapter. The regret optimality of the optimistic Gitiins index, for Beta-Bernoulli bandits, is proved in Section 2.3 (Theorem 1). That is our main theoretical result.

2.2.1 The Gittins Index and Regret

The Gittins index theorem presents a surprisingly simple solution to the problem of computing an optimal policy for the discounted infinite horizon bandit problem. Specifically, the theorem defines for each arm state $y \in \mathcal{Y}$, an index we denote $v_\gamma(y)$; we define this index shortly. The theorem shows that an arm selection rule which at every time selects the arm with the highest index is optimal. The result is powerful in that the computation of the index for a given arm requires the solution of an MDP on the state space $\mathcal{Y}$, as opposed to solving an MDP on the considerably larger state space $\mathcal{Y}^A$.

One way to compute the Gittins Index $v_\gamma(y)$ for an arm in state y is via the so-called retirement value formulation [Whittle, 1980]. Specifically, $v_\gamma(y)$ is defined as the value of λ that solves

$$\frac{\lambda}{1-\gamma} = \sup_{\tau>0} \mathbb{E}_y \left[\sum_{t=1}^{\tau} \gamma^{t-1} X_{i,t} + \gamma^\tau \frac{\lambda}{1-\gamma} \right], \tag{2.1}$$

where the subscript on the expectation indicates that the prior on the (say, ith) arm's mean at time $t = 1$, $y_{i,0}$, equals y. If one thought of the notion of retiring as receiving a deterministic reward λ in every period, then the value of λ that solves the above equation could be interpreted as the per-period retirement reward that makes us indifferent between retiring immediately, and the option of continuing to play arm i with the potential of retiring at some future time. The Gittins index policy itself, which we denote by $\pi^{G,\gamma}$, can succinctly be stated as follows:

$$\textit{At time } t\textit{, play an arm in the set } \operatorname{argmax}_i v_\gamma\left(y_{i,N_i(t-1)}\right),$$

where $N_i(0) \equiv 0$ and $y_{i,0}$ is understood to be the sufficient statistic corresponding to the prior on that arm. Ignoring computational considerations, we cannot hope for the policy $\pi^{G,\gamma}$ to be regret optimal. In fact, as the result below indicates, one cannot even hope for such a policy to be consistent (i.e. have sub-linear regret) in the sense of Lai and Robbins [1985]:

Lemma 1. *For any $\gamma > 0$, there exists an instance of the multi armed bandit problem for which*

$$\text{Regret}\left(\pi^{G,\gamma}, T\right) = \Omega(T).$$

The proof, given in Appendix A.1, rests on the simple fact that for any fixed discount factor, if the posterior means on the two arms are sufficiently apart, the Gittins index policy will pick the arm with the larger posterior mean. The threshold beyond which the Gittins policy 'exploits' depends on the discount factor and with a fixed discount factor there is a positive probability that the superior arm is never explored sufficiently so as to establish that it is, in fact, the superior arm.

2.2.2 Increasing Discount Factors yield sub-linear Bayesian Regret

Lemma 1 tells us that we cannot hope for sub-linear regret by applying the Gittins index policy with a constant discount factor. One may naturally wonder whether an increasing discount factor might fix this issue. Now observe that any schedule of increasing discount factors effectively implies a change in the trade-off between exploration and exploitation. With a fixed discount factor, we have already seen that once the priors between two arms are sufficiently far apart, the Gittins policy will not explore, thereby leading to the possibility of linear regret. As the discount factor increases, the 'gap' between priors above which exploration is not justified goes up over time. If we increase this 'gap' too fast, we might incur too much exploration. Too slow, and we might incur too little exploration. As such, the schedule at which we increase the discount factor is likely to play a significant role in determining the regret of the resulting policy.

Now notice that the Gittins index policy for a discount factor γ can be viewed as optimal for a *random* finite horizon, distributed geometrically with parameter $1 - \gamma$. As γ approaches one, this may be thought of as a near optimal policy for the fixed finite horizon $1/(1 - \gamma)$. Now consider for a moment that we had access to a policy that has optimal T period expected regret (assuming T is known in advance). One

way to convert such a policy into a policy that has 'low' regret for any T is to employ the so-called doubling trick: Apply the optimal policy for the horizon T for T steps, then the optimal policy for horizon $2T$ for the following $2T$ steps, followed by the optimal policy for $4T$ for the next $4T$ steps, and so-forth. Such a policy will be 'near'-optimal for any horizon, in a manner we now make precise.

Consider employing discount factors that increase at roughly the rate $1 - 1/t$; specifically, consider setting

$$\gamma_t = 1 - \frac{1}{2^{\lfloor \log_2 t \rfloor}}$$

and consider using the policy that at time t picks an arm from the set $\operatorname{argmax}_i \nu_{\gamma_t}(y_{i,N_i(t-1)})$. Denote this policy by π^{D}. The following proposition shows that this 'doubling' policy achieves Bayes risk that is within a factor of $\log T$ of the optimal Bayes risk. Specifically, we have:

Proposition 1.

$$\mathrm{Regret}(\pi^{\mathrm{D}}, T) = O\left(\log^3 T\right).$$

where the constant in the big-Oh term depends on the prior q and A.

The proof of this result (Appendix A.2) relies on showing that the finite horizon regret achieved by using a Gittins index with an appropriate fixed discount factor is within a constant factor of the optimal finite horizon regret. The second ingredient is the doubling trick described above. The coarse analysis above illustrates that the use of the Gittins index policy together with an increasing discount factor does indeed yield an algorithm with sub-linear Bayesian regret. It is worth noting that the result above does not show that such a policy achieves *optimal* Bayesian regret (the achievable lower bound being $\log^2 T$ [Lai, 1987]). The analysis does however suggest a candidate discount rate schedule that we will eventually show to yield a regret optimal policy.

2.2.3 Optimistic Approximations to The Gittins Index

The retirement value formulation makes clear that computing a Gittins index is equivalent to solving a discounted, infinite horizon stopping problem. Solving this problem requires substantially more computational effort than, say, Thompson Sampling or the Bayes UCB algorithm. In fact, this computation can even be rendered intractable in practice. Specifically, the set $\mathcal{Y}$ can be high dimensional; see [Chapelle and Li, 2011] for one such example that arises in the context of contextual news recommendations. This motivates an approximation to the Gittins index that is the subject of this section. Specifically, we introduce a sequence of 'optimistic' approximations to the Gittins index that will alleviate computational burden.

Consider the following alternative stopping problem that requires as input the parameters λ (which has the same interpretation as it did before), and K, an integer limiting the number of steps that we need to look ahead. For an arm in state y (recall that the state specifies sufficient statistics for the current prior on the arm reward), let $R(y)$ be a random variable distributed as the prior on expected arm reward specified by y. Define the retirement value $R_{\lambda,K}(s,y)$ according to

$$R_{\lambda,K}(s,y) = \begin{cases} \lambda, & \text{if } s < K \\ \max\left(\lambda, R(y)\right), & \text{otherwise} \end{cases}$$

For a given K, the *Optimistic Gittins Index* for arm i in state y is now defined as the value for λ that solves

$$\frac{\lambda}{1-\gamma} = \sup_{1\leq\tau\leq K} \mathbb{E}_y \left[\sum_{s=1}^{\tau} \gamma^{s-1} X_{i,s} + \gamma^{\tau} \frac{R_{\lambda,K}(\tau, y_{i,\tau-1})}{1-\gamma} \right], \tag{2.2}$$

where we recall that the subscript on the expectation indicates that $y_{i,0} = y$. We denote the solution to this equation by $v_{\gamma}^{K}(y)$.

Let us interpret the stopping problem above. Assume we choose to retire after τ pulls of the arm. If τ were less than K, we then receive a reward λ per period, over the rest of time, discounted at the rate γ. This is no different from what happens in the

stopping problem defining the usual Gittins index, (2.1). On the other hand, unlike that formulation we are forced to retire after the Kth arm pull if we have not done so already. Should we retire at that time, nature reveals the 'true' mean reward of the arm, and we receive the greater of that quantity and λ as our per period retirement payoff. In this manner one is better off than in the stopping problem inherent to the definition of the Gittins index, (2.1), so that we use the moniker 'optimistic'. The following Lemma formalizes this intuition

Lemma 2. *$v_\gamma^K(y)$ is non-increasing in K for all discount factors γ and states $y \in \mathcal{Y}$. Moreover, $v_\gamma^K(y) \to v_\gamma(y)$ as $K \to \infty$.*

Proof. See Appendix A.3.1 □

Now, since we need to look ahead at most K steps in solving the stopping problem implicit in the definition above, the computational burden in index computation is limited. In fact, we will see in a subsequent section that *even the choice of $K = 1$* will make for a compelling policy.

2.2.4 The Optimistic Gittins Index Algorithm

The discussion thus far suggests a simple class of bandit algorithms we dub the Optimistic Gittins Index (OGI) algorithm. The algorithm itself requires as input a prior on arm means (as does any Bayesian algorithm for the MAB), and a parameter K.

The OGI algorithm may be summarized succinctly as follows:

$$\textit{At time } t \textit{ play an arm in the set } \operatorname{argmax}_i v_{\gamma_t}^K(y_{i,N_i(t-1)})$$

where $\gamma_t = 1 - \frac{1}{t}$.

The following Section will establish that the algorithm above achieves the Lai-Robbins lower bound (and thus is regret optimal), for *any* finite K. We will establish this result for Beta priors and Bernoulli rewards. While we do not state this result formally until the next Section (see Theorem 1), it is worth pausing to reflect on the implications of such a result:

1. As K grows large the optimistic Gittins index approaches the Gittins index. The result thus establishes that the use of a set of arbitrarily close approximations to the Gittins index with the discount factor schedule $\gamma_t = 1 - 1/t$ is a regret optimal algorithm. This is a simple, surprising result that bridges two very different flavors of the multi-armed bandit problem. It also suggest the natural conjecture that the use of the Gittins index itself with the discount factor schedule $\gamma_t = 1 - 1/t$ is a regret optimal algorithm.

2. At the other end, since the result establishes regret optimality for any finite K, we have regret optimality for $K = 1$. Computing the optimistic Gittins index in this case is a particularly trivial task, and offers the spectre of a computationally practical algorithm. In fact, in Section 3.4 we shall see precisely this – the choice of $K = 1$ yields an index that materially outperforms a host of state-of-the-art alternatives, while requiring little to no computational overhead relative to even the simplest schemes.

We end this section, with some brief commentary on computation. For concreteness, let us focus on the case of a Beta-Bernoulli bandit. First, we note that solving the stopping problem implicit in the definition of $v_\gamma^K(y)$ for any given value of the retirement subsidy λ requires the solution of a relatively simple dynamic program with just $O(K)$ states. This dynamic program can be solved exactly in $O(K^2)$ time. The optimal value of λ can be found by bisection. For small values of K this is substantially less effort than computing a Gittins index. The case of $K = 1$ is particularly appealing. There, we note that equation (2.2) simplifies to

$$\lambda = \mathbb{E}[R(y)] + \gamma \mathbb{E}[(\lambda - R(y))^+]. \tag{2.3}$$

This equation is easily solved via a method such as Newton-Raphson. In fact, the gradients required for the use of the Newton-Raphson approach are often readily available in closed form. To wit, in the case of a Beta prior with sufficient statistics

(a, b), (2.3) reduces to

$$\lambda = \frac{a}{a+b}\left(1 - \gamma F^{\beta}_{a+1,b}(\lambda)\right) + \gamma\lambda\left(1 - F^{\beta}_{a,b}(\lambda)\right) \triangleq g_{a,b}(\lambda)$$

wherein we see that $\frac{\partial}{\partial\lambda} g_{a,b}(\lambda)$ can be computed in closed form. This makes the use of the Newton-Raphson method for the solution of the equation $\lambda = g_{a,b}(\lambda)$ particularly simple. In our computational experiments, we will see that the choice of $K = 1$ already provides a material improvement in empirical performance over state-of-the-art alternatives.

2.3 Analysis and Regret bounds

We establish a regret bound for the OGI algorithm that applies when the prior distribution q is uniform and arm rewards are Bernoulli. The result shows that the algorithm, in that case, meets the Lai-Robbins lower bound and is thus asymptotically optimal in both a frequentist and Bayesian sense. After stating the main theorem, we briefly discuss a generalization to the algorithm.

In the sequel, we will simplify notation and let $d(x, y) := d_{\mathrm{KL}}(\mathrm{Ber}(x), \mathrm{Ber}(y))$ denote the KL divergence between Bernoulli random variables with parameters x and y. We will also refer to the OGI policy, which uses a look-ahead parameter of K, as $\pi^{\mathrm{OG},K}$ and will write the Optimistic Gittins index of the i^{th} arm at time t as $v^K_{i,t} \triangleq v^K_{1-1/t}(y_{i,N_i(t-1)})$. That way, for the sake of brevity, we will suppress the index's dependence on $y_{i,N_i(t-1)}$. We are ready to state the main result below.

Theorem 1. *Let $\epsilon > 0$ and consider an OGI policy configured with a parameter $K \in \mathbb{N}$ and that assumes Beta$(1,1)$ priors. For the multi-armed bandit problem with Bernoulli rewards and any parameter vector $\theta \subset [0,1]^A$, there exists $T^* = T^*(\epsilon, \theta, K)$ and $C = C(\epsilon, \theta, K)$ such that for all $T \geq T^*$,*

$$\mathrm{Regret}\left(\pi^{\mathrm{OG},K}, T, \theta\right) \leq \sum_{\substack{i=1,\ldots,A \\ i \neq i^*}} \frac{(1+\epsilon)^2(\theta^* - \theta_i)}{d(\theta_i, \theta^*)} \log T + C(\epsilon, \theta, K) \qquad (2.4)$$

where $C(\epsilon, \theta, K)$ is a constant that is determined by ϵ, the parameter θ and K.

Proof. Assume, without loss of generality, that the first arm is uniquely optimal so that $\theta^* = \theta_1$. Fix an arbitrary sub-optimal arm, which for convenience we will say is the second arm. We will strategically fix three constants in between the expected rewards of the first and second arms, namely θ_1 and θ_2. In particular, we let $\eta_1, \eta_2, \eta_3 \in (\theta_2, \theta_1)$ be chosen such that $\eta_1 < \eta_2 < \eta_3$, $d(\eta_1, \eta_3) = \frac{d(\theta_2,\theta_1)}{1+\epsilon}$ and $d(\eta_2, \eta_3) = \frac{d(\eta_1,\eta_3)}{1+\epsilon}$. Next, we define the constant $L(T) := \frac{\log T}{d(\eta_2,\eta_3)}$ to be, intuitively, the optimal length of the exploration period.

The main step in this proof will be to upper bound the expected number of pulls of the second arm, as follows,

$$\begin{aligned}
\mathbb{E}[N_2(T)] &\leq L(T) + \sum_{t=\lfloor L(T) \rfloor + 1}^{T} \mathbb{P}\left(\pi_t^{\mathrm{OG},K} = 2,\ N_2(t-1) \geq L(T)\right) \\
&\leq L(T) + \sum_{t=1}^{T} \mathbb{P}\left(v_{1,t}^K < \eta_3\right) + \sum_{t=1}^{T} \mathbb{P}\left(\pi_t^{\mathrm{OG},K} = 2,\ v_{1,t}^K \geq \eta_3,\ N_2(t-1) \geq L(T)\right) \\
&\leq L(T) + \sum_{t=1}^{T} \mathbb{P}\left(v_{1,t}^K < \eta_3\right) + \sum_{t=1}^{T} \mathbb{P}\left(\pi_t^{\mathrm{OG},K} = 2,\ v_{2,t}^K \geq \eta_3,\ N_2(t-1) \geq L(T)\right) \\
&\leq \frac{(1+\epsilon)^2 \log T}{d(\theta_2, \theta_1)} + \\
&\quad \underbrace{\sum_{t=1}^{\infty} \mathbb{P}\left(v_{1,t}^K < \eta_3\right)}_{A} + \underbrace{\sum_{t=1}^{T} \mathbb{P}\left(\pi_t^{\mathrm{OG},K} = 2,\ v_{2,t}^K \geq \eta_3,\ N_2(t-1) \geq L(T)\right)}_{B},
\end{aligned} \tag{2.5}$$

where the first step is the same as in the analysis of Auer et al. [2002] and applies to any bandit policy. All that remains is to show that terms A and B are bounded by constants. These bounds are given in Lemmas 3 and 4 whose proofs we will now describe at a high-level and defer the full details to the Appendix.

Lemma 3 (Bound on term A)**.** *For any $\eta < \theta_1$, the following bounds holds for some constant $C_1 = C_1(\epsilon, \theta_1, K)$*

$$\sum_{t=1}^{\infty} \mathbb{P}\left(v_{1,t}^K < \eta\right) \leq C_1.$$

Proof outline. The goal is to bound $\mathbb{P}\left(v_{1,t}^K < \eta\right)$ by an expression that decays fast enough in t so that the series converges. This demonstrates that the algorithm encourages enough exploration such that the optimal arm is never underestimated for too long, in expectation. Specifically, we show that there exists a positive constant h so that $\mathbb{P}\left(v_{1,t}^K < \eta\right) = O\left(\frac{1}{t^{1+h}}\right)$ using an induction argument. Proving the base case requires using properties specific to Beta and Bernoulli random variables, while the inductive step is more general. The full proof is contained in Appendix A.4.2.

We remark that the core steps in the proof of Lemma 3, at least in the base case of the induction, rely on properties of the Beta and Bernoulli variables. Because of this, we suspect our analysis can strengthen a similar theoretical result for the Bayes UCB algorithm. In particular, the main theorem of Kaufmann et al. [2012] states that the quantile parameter in the Bayes UCB algorithm should be $1 - 1/(t \log^c T)$ for some constant $c \geq 5$. However, what is perplexing is that their simulation experiments suggest that using a simpler sequence of quantiles, namely $1 - 1/t$, results empirically in a lower mean regret. By utilizing techniques in our analysis, it is possible to prove that the use of the quantiles $1 - 1/t$ would lead to the same optimal regret bound. Therefore the 'scaling' by $\log^c T$ is unnecessary. □

Lemma 4 (Bound on term B)**.** *There exists $T^* = T^*(\epsilon, \theta)$ sufficiently large and a constant $C_2 = C_2(\epsilon, \theta_1, \theta_2)$ so that for any $T \geq T^*$, we have*

$$\sum_{t=1}^{T} \mathbb{P}\left(\pi_t^{\mathrm{OG},K} = 2,\ v_{2,t}^K \geq \eta_3,\ N_2(t-1) \geq L(T)\right) \leq C_2.$$

Proof outline. This relies on a concentration of measure result and the assumption that the 2^{nd} arm was sampled at least $L(T)$ times. Because our index is non-increasing in K, from Lemma 2, it is enough to only consider the simplest case when $K = 1$. The full proof is given in Appendix A.4.3. □

Lemma 3 and 4, together with (2.5), imply that

$$\mathbb{E}[N_2(T)] \leq \frac{(1+\epsilon)^2 \log T}{d(\theta_2, \theta_1)} + C_1 + C_2,$$

and from this the regret bound follows. □

2.3.1 Generalizations and a tuning parameter

As we have shown, the OGI algorithm is regret optimal for the Bernoulli bandit problem. Morever, it is possible to generalize our algorithm to problems with any bounded reward distribution and prove a weaker $O(\log T)$ regret bound. We see this immediately from the discussion in Agrawal and Goyal [2012], where it is shown that any bandit algorithm that is regret optimal for the Bernoulli bandit problem can be modified to yield an algorithm that has $O(\log T)$ regret in a general setting with (bounded) stochastic rewards. Informally, this would require 'emulating ' a Bernoulli bandit problem and assuming Beta$(1, 1)$ priors as before.

Yet another key observation is that the discount factor for Optimistic Gittins Indices does not need to be exactly $1 - 1/t$. In fact, a tuning parameter can be included to make the discount factor $\gamma_{t+\alpha} = 1 - 1/(t + \alpha)$ instead. Intuitively, this would encourage a greater degree of 'exploration' over the arms. An inspection of the proofs of Lemmas 3 and 4 shows that the result in Theorem 1 would still hold were one to use such a tuning parameter. In practice, performance is remarkably robust to our choice of K and α.

2.4 Computational Experiments

Our goal is to benchmark Optimistic Gittins Indices (OGI) against state-of-the-art Bayesian algorithms. Specifically, we compare ourselves against Thomson Sampling, Bayes UCB and IDS. Each of these algorithms has in turn been shown to substantially dominate other extant schemes. Our experimental setup closely follows that of Russo and Van Roy [2014], Kaufmann et al. [2012] and Chapelle and Li [2011]. The experiment from Kaufmann et al. [2012] is deferred to Appendix A.5.1 because it is brief and sends a similar message to the rest of this section. We conclude with a novel experiment to test the problem with multiple simultaneous arm pulls.

For the majority of experiments, we configure the OGI algorithm with $K = 1$

to keep the computational burden under control. In one experiment, included for completeness, we test OGI with $K = 3$ and $K = \infty$, where the latter is equivalent to using Gittins indices. The purpose of those experiments is to show the (limited) value of a higher lookahead in the OGI algorithm.

We use a common discount factor schedule in all experiments setting $\gamma_t = 1 - 1/(100+t)$. The choice of $\alpha = 100$ is second order; our conclusions remain unchanged, and actually appear to improve in an absolute sense with other choices of α. In addition, in one experiment we examine the regret of OGI relative to its competitors up to a horizon of 10^6 epochs, so that this choice of α does not represent an attempt to tune the performance of OGI for a specific time horizon.

2.4.1 Smaller scale experiments with IDS

This section considers a series of smaller scale experiments (10 arms, 1000 time periods) drawn from the paper introducing the IDS algorithm, [Russo and Van Roy, 2014]. A major consideration in running these experiments is that the CPU time required to execute IDS, the closest competitor, based on the current suggested implementation is orders of magnitudes greater than that of the index schemes or Thompson Sampling. The main bottleneck is that IDS uses numerical integration, requiring the calculation of a CDF over, at least, hundreds of iterations. By contrast, the version of OGI with $K = 1$ uses 10 iterations of the Newton-Raphson method.

Gaussian We replicate the Gaussian experiments from Russo and Van Roy [2014]. In the first experiment (Table 2.1), the arms generate Gaussian rewards $X_{i,t} \sim \mathcal{N}(\theta_i, 1)$ where each θ_i is independently drawn from a standard Gaussian distribution. We simulate 1,000 independent trials with 10 arms and 1,000 time periods. The implementation of OGI in this experiment uses $K = 1$. It is difficult to compute exact Gittins indices in this setting, but a classical approximation for Gaussian bandits does exist; see Powell and Ryzhov [2012], Chapter 6.1.3. We term the use of that approximation 'OGI(∞) Approx'. In addition to regret, we show the average CPU time taken, in seconds, to execute each trial.

Algorithm	OGI(1)	OGI(∞) Approx.	IDS	TS	Bayes UCB
Mean	49.19	47.64	55.83	67.40	60.30
Standard error	1.61	1.6	2.08	1.5	1.43
25%	17.49	16.88	18.61	37.46	31.41
50%	41.72	40.99	40.79	63.06	57.71
75%	73.24	72.26	78.76	94.52	86.40
CPU time (s)	0.02	0.01	11.18	0.01	0.02

Table 2.1: Gaussian experiment. OGI(1) denotes OGI with $K = 1$, while OGI Approx. uses the approximation to the Gaussian Gittins Index.

The key feature of the results here is that OGI offers an approximately 10% improvement in regret over its nearest competitor IDS, and larger improvements (20 and 40 % respectively) over Bayes UCB and Thompson Sampling. The best performing policy is OGI with the specialized Gaussian approximation since it gives a closer approximation to the Gittins Index. At the same time, OGI is essentially as fast as Thompson sampling, and three orders of magnitude faster than its nearest competitor (in terms of regret).

Bernoulli We next replicate the Beta-Bernoulli experiments from Russo and Van Roy [2014]. In this experiment regret is simulated over 1,000 periods, with 10 arms each having a uniformly distributed Bernoulli parameter. We simulate 1,000 independent trials and Table 2.2 summarizes the results.

Algorithm	OGI(1)	OGI(3)	OGI(∞)	IDS	TS	Bayes UCB
Mean	18.12	18.00	17.52	19.03	27.39	22.71
Standard error	0.65	0.64	0.68	0.67	0.57	0.56
25%	6.26	5.60	4.45	5.85	14.62	10.09
50%	15.08	14.84	12.06	14.06	23.53	18.52
75%	27.63	27.74	24.93	26.48	36.11	30.58
CPU time (s)	0.19	0.89	(?) hours	8.11	0.01	0.05

Table 2.2: Bernoulli experiment. OGI(K) denotes the OGI algorithm with a K step approximation and tuning parameter $\alpha = 100$. OGI(∞) is the algorithm that uses Gittins Indices.

Each version of OGI outperforms other algorithms and the one that uses exact Gittins Indices shows the lowest mean regret. Perhaps, unsurprisingly, when OGI

looks ahead 3 steps (or when the lookahead is not limited), it performs better than with a single step. It is however apparent that in each of these cases the improvement over simply setting $K = 1$ is marginal. Indeed, looking ahead 1 step is a reasonably close approximation to the Gittins Index in the Bernoulli problem. In the Appendix, we report the approximation error in approximating the Gittins index for various choice of K. When using an optimistic 1 step approximation, the error is around 15% and if K is increased to 3, the error drops to around 4% (see Tables A.1 and A.2 in the Appendix).

As an aside, we note that the regret we computed for the IDS algorithm is slightly different from that reported by Russo and Van Roy [2014]. Specifically, we obtain slightly lower regret for IDS than they report in the Gaussian experiments, and slightly higher values for the Beta-Bernoulli case; we include a link to the code we used to implement the algorithms[1] as a reference.

2.4.2 Large scale experiment

This experiment replicates a large scale synthetic experiment in Chapelle and Li [2011]. The key feature here is that we simulate a longer horizon of $T = 10^6$ and include a large number of arms, particularly we let $A = 100$. This is an order of magnitude greater than in the majority of synthetic bandit experiments we are aware of. Our goal is to see how the algorithms scale both computationally and in terms of performance. Such a setup is practically relevant because in applications such as e-commerce or online advertising, the problems of interest are typically modeled with many arms relative to the horizon, where each arm could represent a product or ad.

Because all the methods we test in our numerical experiments are regret optimal, any relative difference in regret must shrink after a sufficiently large number of time periods. The length of time for this 'burn in' period intuitively depends on the number of arms in the problem. In particular, we can think of the horizon as giving us a rough budget on the number of trials per arm via the ratio T/A. The idea is that with more trials per arm we should expect a smaller relative difference between

[1] `https://github.com/gutin/FastGittins`

the algorithms (and indeed the theoretical guarantees for the algorithms require this happen). We will see that even when the ratio T/A and A itself are large, there is a substantial difference between OGI and the competing benchmarks in both a relative and absolute sense.

As this experiment requires an order of magnitude more iterations than the earlier ones, we are only able to simulate the fastest algorithms, which are OGI with $K = 1$, Thompson Sampling and Bayes UCB. It was not possible to include IDS because its performance is hindered by the fact that each arm pull decision requires time that is quadratic in the number of arms to compute. Again, this is a Bernoulli experiment where arm means are independently sampled from a uniform prior and each algorithm assumes this same prior over the unknown mean rewards from the arms. We show the algorithms' regret averaged over 5,000 trials in Figure 2-1 and Table 2.3.

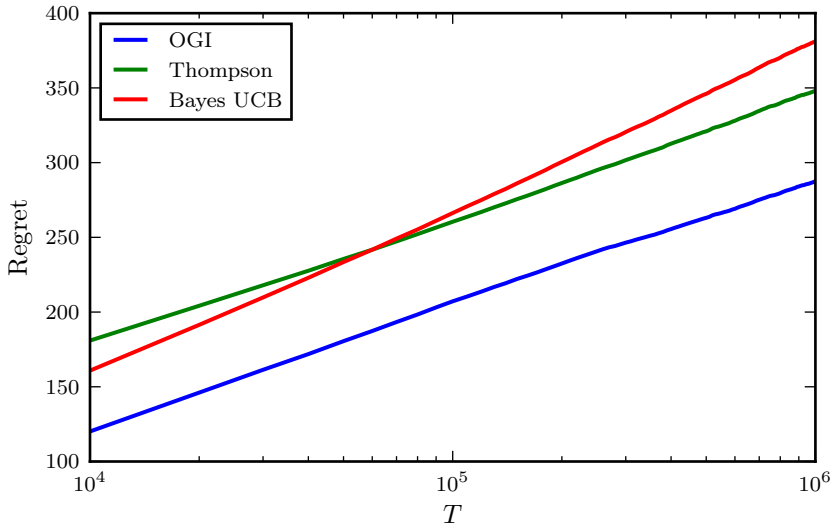

Figure 2-1: Cumulative regret in the large-scale problem of this section averaged over 5,000 independent trials. We plot the number of periods, T on a logarithmic scale.

As before, the OGI scheme consistently dominates the other two. What is particularly interesting is that despite going out to a horizon of 10^6 time periods, the relative improvement in regret over these algorithms remains substantial. For instance, going from a horizon length of 2×10^5 (corresponding to a heuristic budget of $T/A = 200$ pulls per arm) to a horizon length of 10^6 (corresponding to a heuristic

T/A	OGI	Thompson	IDS	Rel. improvement (%)	Abs. improvement
20,000	230.5	284.4	297.9	18.9	53.9
40,000	254.7	311.6	333.5	18.3	57.0
60,000	268.6	327.4	354.5	18.0	58.8
80,000	279.1	339.2	369.6	17.7	60.1
100,000	287.1	347.7	380.7	17.4	60.6

Table 2.3: Regret in the large scale experiment from OGI, Thompson Sampling and Bayes UCB. The last two columns show the relative and absolute difference from Thompson Sampling, which is the closest competitor to OGI.

budget of $T/A = 1000$ pulls per arm) resulted in the relative improvement offered by OGI shrining only marginally, from 18.9% to 17.4%.

2.4.3 Bandits with multiple arm pulls

In this section, we consider a somewhat exploratory experiment; we seek to adapt OGI to a more complex bandit problem (here, we allow for multiple simultaneous arm pulls). Again, in the discounted infinite horizon setting, a number of heuristic approaches have been proposed to adapt the Gittins index to more complex settings; a good example is the so-called Whittle relaxation for restless bandits. One might consider applying those same heuristic strategies to the optimistic gittins index.

For this experiment, we consider a more general MAB problem, where the agent is able to pull up to a certain number (say $m < A$) of the arms simultaneously. In order to describe the problem, we recall that A is the total number of arms and define $\mathcal{D}_m := \{d \in \{0,1\}^A : \sum_i d_i \leq m\}$ to be the set of all A-dimensional binary vectors with up to m ones in them, which we take to be the action space. Let $X_t = (X_{1,t}, \ldots, X_{1,A})$ be a tuple of (potential) rewards from the A arms at time t, where the definition of $X_{i,t}$ for any arm i is the same as in Section **??**. Given a decision $d \in \mathcal{D}_m$, which encodes the subset of arms pulled, the reward $d^\top X_t$ is earned and an arm j's reward $X_{j,t}$ is observed if and only if that arm is pulled, i.e. $d_j = 1$. We can then define a policy $(\pi_t, t \in \mathbb{N})$ to be a $\mathcal{D}_m$-valued stochastic process adapted to an information set generated by past actions and observed feedback. A policy π's

regret is given by the equation

$$\text{Regret}\,(\pi, T) = \max_{d \in \mathcal{D}_m} T \cdot \mathbb{E}\left[d^\top X_t\right] - \sum_{t=1}^{T} \mathbb{E}[\pi_t^\top X_t]$$

where the expectation is over both the randomness in the rewards, the prior and the policy's actions.

We propose a heuristic to this problem using our scheme, which is to compute the Optimistic Gittins Index of every arm, at time t, using a discount factor of $1 - 1/t$ (just as before). However, for this problem, we pick m arms with the largest indices. This is essentially Whittle's heuristic [Whittle, 1988], which was originally given for the restless bandit problem but can be described as picking several arms with the largest Gittins indices.

To test our policy, we simulate $A = 6$ binary arms with uniformly distributed biases and fix $m = 3$. We benchmark our heuristic against Thompson Sampling and IDS. Because the arms give independent Bernoulli rewards, we will use a flat Beta prior for all of the algorithms. We implement the version of IDS designed for the linear bandit problem because this experiment is a special case of a linear bandit. Our implementation of IDS also uses 100 Monte Carlo samples per iteration.

The results, produced from 1,000 independent trials, are summarized in Figure 2-2 and Table 2.4. We notice a significant spread in the performance between OGI and both Thompson Sampling and IDS. Just like for our main algorithm, the primary computational bottleneck in using OGI comes from solving the stopping problem and this can be onerous if K is large. However, as the results suggest, the policy works well even for low to moderate look-ahead parameters. The experiment here sets the stage for an exploration of the appropriate extensions to the OGI algorithm for more complex bandit problems (such as contextual bandits) which we leave for future work.

The results, produced from 2,000 independent trials, are summarized in Figure 2-2 and Table 2.4. The horizon is limited to 250 time periods because of the increased computational effort required to execute a single trial of both the IDS algorithm and Whittle's heuristic, when $K > 1$. This extra time is on the order of minutes for these

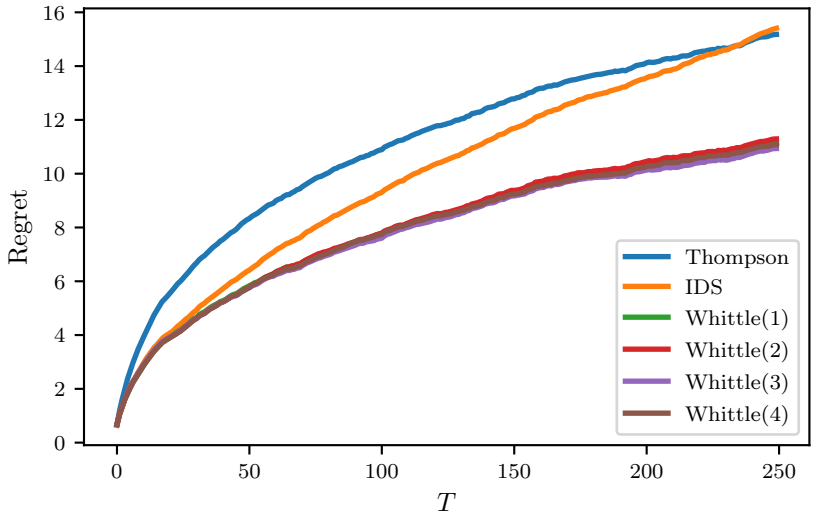

Figure 2-2: Regret for bandits with multiple simultaneous arm pulls

	IDS	Thompson	Whittle(1)	Whittle(3)	Whittle(4)
Mean	15.12	15.23	11.09	11.00	11.11
Standard error	0.21	0.13	0.14	0.15	0.15
25%	1.18	6.60	1.66	1.20	1.39
50%	10.84	14.75	10.34	9.91	9.74
75%	24.60	23.52	19.62	19.27	19.13
CPU time (s)	349.25	2.07	14.20	2196.83	4106.89

Table 2.4: Regret from the multiple arm pulls experiment. "Whittle(K)" refers to the Whittle heuristic policy, where K look-ahead steps are used in computing the Optimistic Gittins index.

algorithms. For the sake of simplicity, we dub this algorithm as exactly 'Whittle's heuristic' for the remainder of this section.

We notice a significant spread in performance between Whittle's heuristic and both Thompson Sampling and IDS. Just like for our main algorithm, the primary computational bottleneck in using Whittle's heuristic comes from solving the stopping problem and this can be onerous if K is large. However, as the results suggest, the policy works well even for low to moderate look-ahead parameters but nonetheless improves slightly when K increases. By contrast, IDS is one of the slowest algorithm because it needs to generate a hundred Monte Carlo samples in every iteration.

Chapter 3

Collateral Management

3.1 Introduction

Collateralized borrowing has become increasingly prevalent in the financial markets after the housing bubble and subsequent crisis between 2007-2009. Major regulatory changes, in addition to market participants' greater preferences for secured borrowing, have reinforced this trend. Driven by the need to mitigate systemic market risks, regulators around the world are pushing for centralized management of collateral-based transactions and demanding more transparency on allocation of assets as collateral. Such changes are putting pressure on investors to manage their collateral more efficiently and have contributed to a significant growth in centralized collateral management services. The increased demand for collateral has also presented itself with new investment opportunities for various types of corporations such as broker-dealers, investment banks and other sell-side firms.

In this chapter, we study the collateral management problem (CMP) faced by a Prime Broker, which provides bespoke securities lending and leveraged trade execution services to buy-side investors. By subscribing to the Prime Broker's bundle of services, its customers, which are typically hedge funds, delegate their day-to-day operational responsibilities around securities clearing and, most importantly, collateral allocation to the Prime Broker. By better understanding the Prime Broker's problem, we seek to draw insights on how a more general centralized collateral management

provider could operate more efficiently.

In addition to focusing on a Prime Broker's business alone, we could also regard the Prime Broker as an individual agent in a multi-agent bilateral securities lending market, specifically an agent who acts as as a lender of assets, or taker of collateral. We will see that even a single lender's problem is a complex, extremely high-dimensional one, which involves a plethora of "moving parts". These include fluctuating asset prices, customer demands and even, as we will discover later, the internal states for each individual customer that the lender interacts with. Arguably the amount of variables (considering we are also taking into account a multi-period version of a problem) involved is greater than traditional financial engineering problems such as portfolio optimization, derivatives pricing, which makes this problem a particularly salient one. By having a grasp on this problem, getting a handle on a game-theoretic version, by conceptually stitching together several single agents' problems, becomes more plausible.

At its core, among other responsibilities, a Prime Broker's main job is to satisfy its clients' borrowing demands. Therefore, we can speak of it as though the Prime Broker maintains an inventory of assets. This inventory depletes with new client demands and gets replenished when the Prime Broker procures securities either by borrowing or purchasing them from external sources in the market. The Prime Broker's revenue is derived from charging fees on the assets lent out, based on a small percentage of their current market price, typically expressed in basis points. This amount also depends on the duration of the loan. In exchange for this, the customer is required to post collateral to cover the economic exposure (i.e. the additional risk to the Prime Broker from counter-party default) from the activity. The notional value of the collateral pledged needs to exceed the value of the loan by a small margin, which is determined usually in terms of a discount applied to each asset pledged as collateral, also known as a *haircut*. The size of the haircut reflects the perceived economic risk in holding a particular asset; a riskier asset (such as a stock) would have a greater haircut than safer, stable securities (e.g. treasury bonds). It is also via the hypothecation of collateral that a Prime Broker is able to refill its inventory and "cross-off" the

flow of assets from different clients, without having to go to the market and pay out fees to other lenders. This practice is known as *internalization* in the industry and is essential for a Prime Broker to remain competitive and improve its business efficiency. We will devote considerable attention to optimizing this process.

One of the questions we consider is how to appropriately set the fees that the Prime Broker charges for lending out assets. If the fees are too low, clearly the Prime Broker will make a loss, otherwise if the fees are too high, or more specifically are above the market rate, the Prime Broker will quickly lose its customers' business. In order to offer the most competitive pricing, we will attempt to both minimize and estimate the costs of satisfying future demand from the Prime Broker's inventory by formulating a multi-period (stochastic) optimization problem. By focusing on the cost-minimization aspect of the problem, and deriving the crucial information of how much it costs to be *lacking* in a certain security, a prime broker can then gauge what borrowing fees it should charge to its clients a priori. We delve into this fundamental idea later on in the chapter.

In contrast to our observations, from what we are aware of, current industry practices use ad-hoc and short-sighted (although somewhat principled) methods of not only setting the Prime Broker's borrowing fees but also managing its day-to-day operations. These are based on an estimate of the *external value* of a given asset, or intuitively the amount of money that can be made if that asset were lent out in future based on its market price and expected future demand. We, on the other hand, formulate a mathematical optimization model that captures the problem a Prime Broker faces. Based on that model, we use an approximate optimal shadow price, as a proxy to estimate an asset's true value to the Prime Broker. This can be seen as the internal value of an asset (as inventory) to the Prime Broker and can be cheaper than an external borrowing fee that a customer might pay out to street lenders. Its lower value would come from the fact that the Prime Broker has access to liquidity from other clients and can take advantage of this, i.e. the internalization it's able to perform as part of its daily operations.

Ultimately, we propose a practical and efficient dual-price based algorithm for

solving the CMP. On top of that, we provide a theoretical guarantee for the algorithm, that is stronger than the guarantees that exist for other dual-price based algorithms applied to inventory-focused revenue-maximization problems that exist in the broader revenue management literature. Moreover, we provide empirical evidence for our algorithm's efficacy by backtesting it on historical data from a Prime Broker and find material cost reductions from borrowing that would lead also to increased annual profits (on the order of millions of dollars) for the Prime Broker. In summary, the main contributions we make from this work are:

- We introduce a novel formulation of a collateral management problem (CMP) of interest to a Prime Broker, and potentially similar securities lenders. The model is based on solving an intractable multi-period, stochastic optimization problem.

- We propose a simple dual-price based algorithm for the CMP and show that is not only asymptotically optimal but also has, in several useful settings, a *constant* additive loss relative to an optimal policy.

- We introduce a new set of simulation benchmarks for the CMP that are both purely data-driven and based on synthetic data. From these benchmarks, we demonstrate empirically that our dual-price based algorithm offers substantially improved revenues for a securities lender than the incumbent approaches that are popular today.

We now proceed to discuss existing ways the collateral management problem has been approached, as well as discussing similar problems from the revenue management literature, in the following review.

3.2 Model

This section describes a multi-period model for a securities lender who needs to loan assets to multiple clients and hypothecate collateral back from them. We refer to this as the Collateral Management Problem (CMP).

Setup We consider a universe of K financial assets indexed by $k \in \{1, \ldots, K\} = [K]$ where we say that the first asset is cash. There is a single securities lender and N borrowers. Throughout this chapter we let $i \in \{1, \ldots, N\} = [N]$ index the borrowers. Initially, the lender is endowed with an inventory $x_k \in \mathbb{R}_+$ for each asset $k \in [K]$, and we denote by $x \triangleq (x_1, \ldots, x_K)$ the tuple of the K inventories. Each of the assets is assigned a 'haircut' $h_k \in [0, 1]$ that gives the fraction of its market value accepted as collateral. Denote by $h \triangleq (h_1, \ldots, k_K) \in [0, 1]^K$ the tuple of haircuts. We will now describe the (random) dynamics of the model.

Let T be a given integer horizon and let us index time periods by $t \in \{1, \ldots, T\} = [T]$. With each client i, we associate a non-negative real-valued (discrete-time) stochastic process $(\delta^i_{k,t}(\omega))_{t\in[T]}$ where $\delta^i_{k,t}(\omega)$ denotes the quantity of the kth asset that client i requests to borrow at time t. We will refer to this random variable as the *demand* from i at t and we call the sequence of such demands as a *demand process* from client i. If we omit the subscript k, $\delta^i_t(\omega)$ denotes the vector of client demands for different assets. Each client also possesses their own inventory of assets that can be utilized for taking collateral. For client i, we model its inventory of asset k as another non-negative stochastic process $(b^i_{k,t}(\omega))_{t\in[T]}$; the vector of these values for the different assets is then written as $b^i_t(\omega)$. The final primitives in our model are asset prices, which are given as $\mathbb{R}^K_+$-valued stochastic processes $(p_t(\omega))_{t\in[T]}$. All random variables are defined on a common probability space $(\Omega, \mathcal{F}, \mathbb{P})$ and we will suppress dependence on ω whenever it's otherwise clear.

Problem Firstly, we denote by $\mathcal{F}_t$ the σ-algebra generated by sample paths of the price, demand and inventory processes from all clients up to time t, i.e.

$$\mathcal{F}_t \triangleq \sigma\left(\left\{(p_s(\omega), \delta^i_s(\omega), b^i_s(\omega)) : s = 1, \ldots, t,\ i \in [N]\right\},\ \omega \in \Omega\right).$$

The lender is obligated to always satisfy a client's demands for an asset by lending it out when requested. We state this notion precisely by defining a policy as

$$v_t(\omega) \triangleq (u^1_t(\omega), \ldots, u^N_t(\omega), z_t(\omega)),$$

namely, a tuple of $N+1$ non-negative K-dimensional processes, each one being adapted to the filtration $\mathcal{F}_t$. A policy v_t is called feasible if it almost surely satisfies all of the constraints,

$$\sum_{s=1}^{t} z_s + \sum_{s=1}^{t}\sum_{i=1}^{N} u_s^i \geq \sum_{s=1}^{t}\sum_{i=1}^{N} \delta_t^i - x, \qquad \forall\, t \in [T] \tag{3.1}$$

$$(h \cdot p_t)^\top u_t^i = p_t^\top \delta_t^i, \qquad \forall\, t \in [T], i \in [N] \tag{3.2}$$

$$u_t^i \leq b_t^i, \qquad \forall\, t \in [T], i \in [N]. \tag{3.3}$$

where '$\cdot$' is a binary operator for the component-wise product of two vectors. We denote the set of feasible policies by Π_T. As a technical assumption to ensure feasibility, we shall also assume that $b_{1,t}^i(\omega)$, namely the supply of cash, is set to an large enough constant and $p_{1,t} = 1$, $h_1 = 1$. This constant could then be the maximum possible exposure, or equivalently the maximum conceivable value for the expression $p_t^\top \delta_t^i$.

Note that (3.1) is a 'covering' constraint which states that all cumulative demands for assets, in excess of x, must be covered by assets borrowed externally or re-used from previously taken collateral. Constraints (3.2) and (3.3) state that the market value of collateral cannot be more than that of the demand (loan) and that it can only be taken from the client's inventory. Finally, we emphasize that due to assumption on the supply of cash, it holds that for any realization of ω, there always exists a feasible policy.

We will assume that re-using assets from collateral incurs no cost to the lender but procuring an asset k from the external marketplace costs c_k for every unit taken in this way. Thus, letting $c \triangleq (c_1, \ldots, c_K)$ be a cost vector, we aim to solve the stochastic problem for minimizing the lender's expected costs:

$$\underset{(z,u^1,\ldots,u^N)\in\Pi_T}{\text{minimize}} \quad \sum_{t=1}^{T} \mathbb{E} c^\top z_t \tag{3.4}$$

whose optimal value is denoted by $J_T^*(x)$. For values of K larger than, say, 3 or 4, the problem suffers from the 'curse of dimensionality'. One of the main goals

in this chapter is to show, how in some general problem settings, we can break the curse of dimensionality by finding provably near-optimal and computationally efficient algorithms.

3.3 Problem Analysis and Collateral Prices

In this section we focus on Problem 3.7, by considering several settings, where it is possible to derive a near-optimal policy in polynomial time. As a conclusion of this work, we derive a key insight about valuation of assets as collateral.

We begin by making some immediate observations from Section 3.2. First of all note that while constraints (3.3) and (3.2) are specific to the main problem in this chapter, we can actually analyze a more general setup, which has dual benefits of simplifying the subsequent analysis and demonstrating that different kinds of constraints could be incorporated to model the lender's requirements on collateral. In particular, if we take the process $\xi_t(\omega) \in \mathbb{R}^d$ to represent exogenous random data and $\delta_t = \delta_t(\omega)$ to be a single demand process, we can consider sets parameterized by ξ_t. Particularly, we can let $U : \mathbb{R}^d \mapsto \mathcal{K}$ be a mapping from parameters ξ to the family of closed, convex sets $\mathcal{K}$, defined according to

$$U(\xi_t) \triangleq \left\{ u_t \in \mathbb{R}^K_+ : g_l(u_t; \xi_t) \leq 0, \quad l = 1, \ldots, L \right\}$$

where the functions g_l are convex and $U(\xi_t) \neq \emptyset$ for any choices of t and ξ_t. Therefore if we redefine the filtration $\mathcal{F}_t$ in terms of the random variables δ_t, ξ_t and consider $\mathcal{F}_t$-measurable functions, $u_t, z_t : \Omega \mapsto \mathbb{R}^K_+$ that satisfy, almost surely, the constraints

$$\sum_{s=1}^{t} z_s + \sum_{s=1}^{t} u_s \geq \sum_{s=1}^{t} \delta_t - x, \qquad \forall\, t \in [T] \tag{3.5}$$

$$u_t \in U(\xi_t) \qquad \forall\, t \in [T] \tag{3.6}$$

this represents a new kind of feasible policy belonging to a class Π^G_T. Now consider

the following problem

$$\underset{(z,u)\in\Pi_G}{\text{minimize}} \quad \sum_{t=1}^{T} \mathbb{E} c^\top z_t. \tag{3.7}$$

We redefine $J_T^*(x)$ to be the optimal value of (3.7) and make the following claim in order to relate it back to Section 3.2.

Lemma 5. *Problem* (3.7) *subsumes Problem* (3.4)

Proof. First, we can see this by re-writing $\sum_i \delta_t^i$ for every t as δ_t in Problem (3.4). Then we take $\xi_t = (b_t^1, \ldots, b_t^N, \delta_t^1, \ldots, \delta_t^N, p_t)$ to be the concatenation of the original data and then let

$$U(b_t^1, \ldots, b_t^N, \delta_t^1, \ldots, \delta_t^N, p_t) = \left\{ u_t \in \mathbb{R}_+^K : u_t = \sum_i u_t^i, \;\; (h \cdot p_t)^\top u_t^i = p_t^\top \delta_t^i, \;\; 0 \le u_i \le b_t^i \right\}. \tag{3.8}$$

□

We have successfully abstracted away, from the original problem, details of clients, prices, haircuts and borrower inventories, which yielded the general formulation of Problem (3.7).

3.3.1 Lower bound on cost

We begin the main body of our analysis by computing a lower bound on $J_T^*(x)$ given in the Lemma below. This will be useful as it will allow us to approximate the optimal value. Throughout, we will use the shorthand $\bar{\delta}_t = \mathbb{E}[\delta_t]$ to denote expected demand.

Lemma 6. *Let* $(\lambda_t \in \mathbb{R}_+^K, \; t = 1, \ldots, T)$ *be any sequence of vectors such that* $\lambda_{t+1} \le \lambda_t \le c$ *for all* $t = 1, \ldots, T-1$. *Consider the function*

$$L(\lambda_1, \ldots, \lambda_T; x) \triangleq \sum_{t=1}^{T} \lambda_t^\top \bar{\delta}_t - \sum_{t=1}^{T} \mathbb{E}\left[\sup_{u_t \in U(\xi_t)} \lambda_t^\top u_t \right] - \lambda_1^\top x \tag{3.9}$$

where the expectation is understood to be over ξ_t. Then we have that

$$L(\lambda_1, \ldots, \lambda_T; x) \leq J_T^*(x).$$

Proof. We will derive the lower bound L by performing a sequence of relaxations on (3.7). First we start by relaxing the first set of constraints to hold in expectation. This gives the following lower bound to $J_T(x)$

$$\begin{aligned} \underset{((u_t,z_t))_{t=1}^T \in \Pi_T^G}{\text{minimize}} \quad & \sum_{t=1}^{T} \mathbb{E}\left[c^\top z_t\right] \\ \text{subject to} \quad & \mathbb{E}\left[\sum_{s=1}^{t} z_s + \sum_{s=1}^{t} u_s + x\right] \geq \sum_{s=1}^{t} \bar{\delta}_t, && \forall t \in [T] \\ & u_t \in U(\xi_t) && \forall t \in [T],\ \mathbb{P}-a.s. \end{aligned} \tag{3.10}$$

Since $\lambda_{t+1} \leq \lambda_t$ for all t, we may consider the defining auxiliary variables $\nu_t \triangleq \lambda_t - \lambda_{t+1}$ for $t \leq T-1$ and $\nu_T \triangleq \lambda_T$. Now because, by construction, $\nu_t \geq 0$, the following Lagrangian relaxation is a a lower bound to (3.10)

$$\begin{aligned} \underset{((u_t,z_t))_{t=1}^T \in \Pi_T^G}{\text{minimize}} \quad & \sum_{t=1}^{T} \mathbb{E}\left[c^\top z_t\right] - \sum_{t=1}^{T} \nu_t^\top \mathbb{E}\left[\sum_{s=1}^{t} z_s + \sum_{s=1}^{t} u_s + x - \sum_{s=1}^{t} \delta_t\right] \\ \text{subject to} \quad & u_t \in U(\xi_t) && \forall t \in [T] \end{aligned} \tag{3.11}$$

Intuitively, a shortfall in the supply of assets will penalize the objective function, while a surplus improves it. Now we let $(z_t^*, u_t^*)_{t=1}^T \in \Pi_T^G$ denote an optimal policy to Problem (3.11) and we re-arrange terms in the optimal objective function of the

relaxed problem as follows:

$$\begin{aligned}
&\sum_{t=1}^{T}\mathbb{E}\left[c^{\top}z_t^*\right]-\sum_{t=1}^{T}\nu_t^{\top}\mathbb{E}\left[\sum_{s=1}^{t}z_s^*+\sum_{s=1}^{t}u_s^*+x-\sum_{s=1}^{t}\delta_s\right]\\
&\quad=\sum_{t=1}^{T}\mathbb{E}\left[\left(c-\sum_{s=t}^{T}\nu_s\right)^{\top}z_t^*\right]-\mathbb{E}\left[\sum_{t=1}^{T}\nu_t^{\top}\sum_{s=1}^{t}\left(u_s^*-\delta_s\right)\right]-\sum_{t=1}^{T}\nu_t^{\top}x \qquad (3.12)\\
&\quad=\sum_{t=1}^{T}\mathbb{E}\left[\left(c-\lambda_t\right)^{\top}z_t^*\right]-\mathbb{E}\left[\sum_{t=1}^{T}\left(\sum_{s=t}^{T}\nu_t\right)^{\top}\left(u_t^*-\delta_t\right)\right]-\lambda_1^{\top}x \qquad (3.13)\\
&\quad=\sum_{t=1}^{T}\mathbb{E}\left[\left(c-\lambda_t\right)^{\top}z_t^*\right]-\mathbb{E}\left[\sum_{t=1}^{T}\lambda_t^{\top}\left(u_t^*-\delta_t\right)\right]-\lambda_1^{\top}x \qquad (3.14)\\
&\quad=\mathbb{E}\left[\sum_{t=1}^{T}\lambda_t^{\top}\left(\delta_t-u_t^*\right)\right]-\lambda_1^{\top}x \qquad (3.15)\\
&=\sum_{t=1}^{T}\lambda_t^{\top}\bar{\delta}_t-\sum_{t=1}^{T}\mathbb{E}\left[\lambda_t^{\top}u_t^*\right]-\lambda_1^{\top}x\\
&=\sum_{t=1}^{T}\lambda_t^{\top}\bar{\delta}_t-\sum_{t=1}^{T}\mathbb{E}\left[\sup_{u_t\in U(\xi_t)}\lambda_t^{\top}u_t\right]-\lambda_1^{\top}x \qquad (3.16)\\
&=L(\lambda_1,\ldots,\lambda_T;x).
\end{aligned}$$

Equations (3.12), (3.13) follow from changing the order of summation and linearity of expectation, (3.13) also follows from the way the variables ν_t were constructed in the first place [so that $\lambda_t=\sum_{s=t}^{T}\nu_s$]. Equation (3.15) is due to z_t^* being an optimal control and $c-\lambda_t\geq 0$ for all t. Finally (3.16) holds since it is optimal, by the exogeneity of ξ_t, for the u_t^* to be chosen in a myopic fashion. □

Since $L(\lambda_1,\ldots,\lambda_T;x)$ is a lower bound on the optimal objective function $J_T^*(x)$, it is fruitful to consider maximizing the aforementioned function, which gives the the dual problem

$$J^{\lambda}(x)\triangleq\sup_{\lambda_1,\ldots,\lambda_T\in\mathbb{R}_+^K}\left\{L(\lambda_1,\ldots,\lambda_T;x):0\leq\lambda_1\leq\ldots\leq\lambda_T\leq c\right\}. \qquad (3.17)$$

As we will demonstrate later, the possibly different values of variables λ_t are not that crucial, so we may consider a (slightly) weaker lower bound, which is easier to

compute and analyze, given as follows. First of all, let us define for $0 \leq \lambda \leq c$, the function $L_T(\lambda; x) \triangleq L(\lambda, \ldots, \lambda; x)$, that is, it's the objective function value of the dual problem when its arguments satisfy $\lambda_1 = \ldots = \lambda_T = \lambda$. Now we introduce the problem

$$\hat{J}_T^{\lambda}(x) \triangleq \sup_{\lambda \in \mathbb{R}_+^K} \{L_T(\lambda; x) : \lambda \leq c\} \tag{3.18}$$

which gives the tightest lower bound to (3.11) among the possible bounds from $L_T(\lambda; x)$. Then we record the following relationships between the optimal value to Problem (3.11) and the bounds considered so far.

Lemma 7. *We have for any initial inventory $x \in \mathbb{R}_+^K$ that*

$$\hat{J}_T^{\lambda}(x) \leq J_T^{\lambda}(x) \leq J_T^*(x)$$

Proof. The right bound is immediate from Lemma 6 in that for any $0 \leq \lambda_1 \leq \ldots \leq \lambda_T \leq c$ we have $L(\lambda_1, \ldots, \lambda_T; x) \leq J_T^*(x)$, so this is also true for the supremum over the λ_t.

For the left bound notice that

$$\hat{J}^{\lambda}(x) \triangleq \sup_{\lambda_1, \ldots, \lambda_T \in \mathbb{R}_+^K} \{L(\lambda_1, \ldots, \lambda_T; x) : 0 \leq \lambda_1 \leq \ldots \leq \lambda_T \leq c,\ \lambda_1 = \ldots = \lambda_T\}. \tag{3.19}$$

which looks identical to problem (3.17) except that the feasible set of (3.19) is a subset of that for (3.17). □

To conclude this preliminary analysis, we are going to derive some properties of the optimal solution λ^* for $\hat{J}^{\lambda}(x)$ that will lead us to develop a simple but asymptotically optimal algorithm. We state these properties in the below Lemma.

Lemma 8. *Firstly, the optimization problem* (3.18) *is convex.*

Secondly, let us denote by $\partial_{\lambda} f(\lambda)$ the sub-differential of a function f at a point λ. Suppose λ^* is an optimal solution to (3.18)*, then there exist vectors $\bar{u}_t \in \mathbb{R}^K$ such that $\bar{u}_t \in \partial_{\lambda^*} \mathbb{E}\left[\sup_{u_t \in U(\xi_t)} (\lambda^*)^{\top} u_t\right]$ for all t, and such that for all assets $k \in [K]$, the*

following statements hold

$$\lambda_k^* < c_k \Longrightarrow \sum_{t=1}^{T} \bar{\delta}_{k,t} \leq \sum_{t=1}^{T} \bar{u}_{k,t} + x_k$$
$$\sum_{t=1}^{T} \bar{\delta}_{k,t} < \sum_{t=1}^{T} \bar{u}_{k,t} + x_k \Longrightarrow \lambda_k^* = 0.$$

Proof. The first property follows from showing that $L(\lambda; x)$ is concave in λ and the second property from finding the relevant KKT conditions. The full proof is in Appendix B.1.1. □

3.3.2 Near-optimal selection policies

For this part of the analysis, we will introduce (collateral) selection rules, which are a key ingredient to developing policies for the main problem in this chapter. These are a class of functions that in each period pick a vector $u_t \in U(\xi_t)$ from the admissible sets. Since we derived, in some sense, optimal dual variables for the assets denoted by λ^*, we will use them in designing selection rules. In fact, the class of functions we consider is specific enough that we make the following definition:

Definition 1 (λ-)selection rule)**.** *For any $\lambda \in \mathbb{R}_+^K$, a function $u^* : \mathbb{R}^d \mapsto \mathbb{R}^K$, parameterized by λ, so that for any ξ*

$$\lambda^\top u^*(\xi; \lambda) = \sup_{u \in U(\xi)} \lambda^\top u$$

and $u^*(\xi; \lambda) \in U(\xi)$ is called a selection rule, or a λ-selection rule to make clear its parameter.

We relate back selection rules to the original problem (3.7) by constructing policies from them. In fact, we define a class of policies as follows:

Definition 2 (λ-selection policy)**.** *Fix $\lambda \in \mathbb{R}^K$ as a parameter. Let $u_t^{S,\lambda} := u_t^*(\xi_t; \lambda)$ and*

$$z_t^{S,\lambda} := \left(\sum_{s=1}^{t} \delta_s - \sum_{s=1}^{t-1} z_s^{S,\lambda} - \sum_{s=1}^{t} u_s^{S,\lambda} - x \right)^+$$

Then the sequence of pairs $((u_t^{S,\lambda}, z_t^{S,\lambda}))_{t=1}^T$ *is called a* λ*-selection policy.*

The next proposition verifies that this does indeed define a feasible policy for our problem.

Proposition 2. *Consider the sequence* $((u_t^{S,\lambda}, z_t^{S,\lambda}))_{t=1}^T$ *in Definition 2. Then this sequence almost surely satisfies the constraints of Problem 3.7 and is therefore a feasible policy.*

Proof. We show claim as follows. We have for any t that

$$\begin{aligned}\sum_{s=1}^{t} z_s^{S,\lambda} + \sum_{s=1}^{t} u_s^{S,\lambda} + x &= \left(\sum_{s=1}^{t} \delta_s - \sum_{s=1}^{t-1} z_s^{S,\lambda} - \sum_{s=1}^{t} u_s^{S,\lambda} - x\right)^+ + \sum_{s=1}^{t-1} z_s^{S,\lambda} + \sum_{s=1}^{t} u_s^{S,\lambda} + x \\ &\geq \sum_{s=1}^{t} \delta_s\end{aligned}$$

and finally $u_t^{S,\lambda} = u_t^*(\xi;\lambda) \in U(\xi_t)$, which completes the proof. □

Since the λ-selection policy is feasible, it will be helpful to define its expected cost so that we can compare it to the optimal one. Therefore, we write

$$J^{S,\lambda}(x) \triangleq \sum_{t=1}^{T} \mathbb{E}\left[z_t^{S,\lambda}\right].$$

The main result of this section will be that the λ^*-selection policy, that is the λ-selection policy parameterized by $\lambda = \lambda^*$, is under fairly mild assumptions asymptotically optimal for Problem 3.7. Furthermore, in several important cases, the optimality gap *is bounded by a constant* that does not grow at all with T. We now make the first of the assumptions used for the result.

Assumption 1. *The mapping* $U : \mathbb{R}^d \mapsto \mathcal{K}$ *is such that for all* $\xi \in \mathbb{R}^d$*, there exists a unique selection rule* $u^*(\xi;\lambda^*)$*. That is,* $u^*(\xi;\lambda^*)$ *is the unique maximizer of* $(\lambda^*)^\top u_t$ *over all* $u_t \in U(\xi_t)$*, which is guaranteed to exist.*

With this assumption in place, we can uniquely identify a selection policy that 'satisfies' the optimality conditions given by λ^* in Lemma 8. The following Proposition states this observation.

Proposition 3. *Let λ^* be an optimal solution to Problem* (3.18) *and suppose that U satisfies Assumption 1. We then have, for every time t, that*

$$\partial_\lambda \mathbb{E}\left[\sup_{u_t \in U(\xi_t)} (\lambda)^\top u_t\right] = \{\mathbb{E}\left[u_t^*(\xi_t;\lambda)\right]\} \tag{3.20}$$

and therefore for every asset k, the following statements hold

$$\lambda_k^* < c_k \Longrightarrow \sum_{t=1}^{T} \bar{\delta}_{k,t} \leq \sum_{t=1}^{T} \mathbb{E}\left[u_{k,t}^*(\xi_t)\right] + x_k \tag{3.21}$$

$$\sum_{t=1}^{T} \bar{\delta}_{k,t} \leq \sum_{t=1}^{T} \mathbb{E}\left[u_{k,t}^*(\xi_t)\right] + x_k \Longrightarrow \lambda_k^* = 0 \tag{3.22}$$

where $u_t^*(\xi_t) \triangleq u_t^*(\xi;\lambda^*)$.

Proof. The proof essentially shows that if the selection rule is unique, then the sub-differential can be "replaced" with a regular gradient. In that case, we would use Lemma 8 and the fact that $\bar{u}_t = \mathbb{E}\left[u_t^*(\xi_t;\lambda)\right]$ always. The full proof is deferred to Appendix B.1.1. □

Before stating our main results, we state key definitions and Lemmas used to describe the state of the system, namely the lender's inventory, which until now we essentially avoided.

Definition 3 (Inventory process)**.** *Let us define the shorthand $\Delta_t \triangleq \delta_t - u_t^{S,\lambda^*}$. For every asset k, consider the discrete time process given recursively as*

$$x_{k,t} = \begin{cases} x_k & t = 1 \\ (x_{k,t-1} - \Delta_{k,t-1})^+ & t > 1 \end{cases}$$

which we will refer to as an inventory process for asset k. The tuple of inventory processes for all K assets is denoted by x_t.

Using this definition, we state two Lemmas used in the proofs of the subsequent main results.

Lemma 9.

$$J_T^{S,\lambda^*}(x) = \sum_{t=1}^{T} \mathbb{E}\left[c^\top (\Delta_t - x_t)^+\right].$$

Proof. The proof relies on induction and appears in Appendix B.1.1. □

Lemma 10. *For any asset k, we have*

$$\sum_{t=1}^{T} \mathbb{E}\left[(\Delta_{k,t} - x_{k,t})^+\right] = \sum_{t=1}^{T} \mathbb{E}\left[\Delta_{k,t}\right] - x_k + \mathbb{E}\left[x_{k,T+1}\right].$$

The proof of this Lemma also appears in the Appendix, in Section B.1.1. We are now ready to state the following general result, which provides a method of bounding the optimality gap for the λ^*-selection policy. The purpose of the following Lemma is to help us bound the gap in different problem settings, and it will be apparent shortly under which conditions the loss against an optimal policy is in fact just a constant.

Lemma 11. *Consider the λ^*-selection policy and the resulting inventory processes $\{x_{k,t}\}_{t=1}^{\infty}$ of every asset $k \in [K]$ induced by the policy. Let $h(T)$ be a non-negative, non-decreasing function of the horizon. Provided every asset k satisfies at least one of the following two conditions:*

1. $\sup_{t\geq 1} \mathbb{E}\left[x_{k,t}\right] < \infty$
2. $\sum_{t=1}^{T} \mathbb{E}\left[(\Delta_{k,t} - x_{k,t})^+\right] \leq h(T)$

and Assumption 1 holds, we have that

$$J_T^*(x) \leq J_T^{S,\lambda^*}(x) \leq J_T^*(x) + C(K) + Kh(T)$$

for any $T \geq 1$, where C is a constant that does not depend on the horizon T, but does on K.

Proof. Suppose, without loss of generality, that only the first $B \leq K$ assets satisfy condition (1), while the remaining assets indexed by $k' = B+1, \ldots, K$ do not satisfy condition (1) but do satisfy (2).

We begin by breaking up the costs from the two sets of assets

$$J_T^{S,\lambda^*}(x) = \sum_{t=1}^{T} \mathbb{E}\left[c^\top(\Delta_t - x_t)^+\right] \tag{3.23}$$

$$= \sum_{k=1}^{B}\sum_{t=1}^{T} \mathbb{E}\left[c_k(\Delta_{k,t} - x_{k,t})^+\right] + \sum_{k'=B+1}^{K}\sum_{t=1}^{T} \mathbb{E}\left[c_{k'}(\Delta_{k',t} - x_{k',t})^+\right]$$

$$\leq \sum_{k=1}^{B}\sum_{t=1}^{T} \mathbb{E}\left[c_k(\Delta_{k,t} - x_{k,t})^+\right] + Kh(T) \tag{3.24}$$

where (3.23) follows Lemma 9 and above the inequality is implied by the assumptions made on the last $K - B$ assets. All that remains is to bound the second term.

To that end, for each asset $k \leq B$ we will define the constant $M_k \triangleq \sup_{t\geq 1} \mathbb{E}\left[x_{k,t}\right] < \infty$ and bound the left hand term of (3.24) as follows:

$$\sum_{k=1}^{B}\sum_{t=1}^{T} \mathbb{E}\left[c_k(\Delta_{k,t} - x_{k,t})^+\right] = \sum_{k=1}^{B} c_k \left(\sum_{t=1}^{T} \mathbb{E}\left[\Delta_{k,t}\right] + \mathbb{E}\left[x_{k,T+1}\right] - x_k\right) \tag{3.25}$$

$$\leq \sum_{k=1}^{B}\left(c_k\mathbb{E}\left[\sum_{t=1}^{T}\left(\delta_{k,t} - u^*_{k,t}(\xi_t;\lambda^*)\right) - x_k\right] + c_k M_k\right)$$

$$\leq \sum_{k=1}^{B}\left(\lambda^*_k\mathbb{E}\left[\sum_{t=1}^{T}\left(\delta_{k,t} - u^*_{k,t}(\xi_t;\lambda^*)\right) - x_k\right]\right) + \sum_{k=1}^{B} c_k M_k \tag{3.26}$$

$$\leq \sum_{k=1}^{K}\left(\lambda^*_k\mathbb{E}\left[\sum_{t=1}^{T}\left(\delta_{k,t} - u^*_{k,t}(\xi_t;\lambda^*)\right) - x_k\right]\right) + \sum_{k=1}^{B} c_k M_k \tag{3.27}$$

$$= L_T(\lambda^*; x) + \sum_{k=1}^{B} c_k M_k$$

$$\leq J^*_T(x) + \sum_{k=1}^{B} c_k M_k \tag{3.28}$$

where Equation (3.25) uses Lemma 10. The second last inequality, (3.26), is due to Proposition 3 and the fact that λ^* is non negative. The inequality in (3.27) follows from Proposition 3 and, in particular from statement (3.22). Finally, (3.28) follows from the result in Lemma 6. □

It's immediately clear from the above Lemma that, as long as the function $h(T)$ that we find is bounded from above as $T \to \infty$, the policy suffers a constant additive loss.

Theorem 2. *Suppose that Assumption 1 holds and the demand process $\{\delta_t\}_{t=1}^{\infty}$ is an irreducible finite-state Markov chain in steady state (the distribution of the starting state is the same as the steady state). Suppose also that δ_t and the control $u_t^*(\xi_t; \lambda^*)$ are almost surely bounded at every time t. That is, there exists a constant $D \in \mathbb{R}_+$ such that*

$$|\delta_t - u| \leq D, \quad \forall u \in U(\xi_t) \ \ a.s.$$

for all $t = 1, 2, \ldots$.

Suppose further that the exogenous process, ξ_t, is drawn from an i.i.d sequence. Under these assumptions, the λ^* selection policy is asymptotically optimal up to a constant, in the sense that

$$J_T^*(x) \leq J_T^{S,\lambda^*}(x) \leq J_T^*(x) + C(K),$$

for all integers $T \geq 1$ and some constant $C(K)$ that depends on K.

Proof. We prove properties about the inventory processes $\{x_{k,t}\}_{t=1}^{\infty}$ corresponding to every asset, which will allow us to apply Lemma 11 and conclude the result. Specifically, we will show that each asset k satisfies at least one of conditions (1) or (2) in Lemma 11.

It is simple to see that for every asset, the process $\{\Delta_{k,t}\}_{t=1}^{\infty}$ satisfies the Markov property. This is because the random variable ξ_t, which determines the feasible set $U(\xi_t)$, is drawn from i.i.d process. Moreover, $\delta_{k,t}$ was assumed to be a Markov chain.

So let us suppose condition (1) does not hold for an arbitrary asset k and we will show that (2) must then hold. The proof consists of two parts.

Part 1: Showing that $\mu_k \triangleq \mathbb{E}[\Delta_{k,t}] < 0$. We will assume for contradiction that $\mu_k \geq 0$. Then we will prove by induction that, under this assumption, $\mathbb{E}[X_{k,t}] \leq R \triangleq x_k + 3D$ for all times t (which would exactly give us the contradiction).

Proving the base case for the first time period is trivial. Now let us fix an arbitrary time $s > 1$, and assume the induction hypothesis that $\mathbb{E}[x_{k,s}] \leq x_k + 3D$. First of all, we note that

$$\begin{aligned}
x_{k,s+1} &= \mathbb{1}(x_{k,s} \geq \Delta_{k,s})\, x_{k,s+1} + \mathbb{1}(x_{k,s} < \Delta_{k,s})\, x_{k,s+1} \\
&= \mathbb{1}(x_{k,s} \geq \Delta_{k,s})\, x_{k,s+1} \\
&= \mathbb{1}(x_{k,s} \geq \Delta_{k,s})\,(x_{k,s} - \Delta_{k,s}) \\
&= \mathbb{1}(x_{k,s} \geq \Delta_{k,s})\, x_{k,s} - \mathbb{1}(x_{k,s} \geq \Delta_{k,s})\, \Delta_{k,s} \\
&= \mathbb{1}(x_{k,s} \geq \Delta_{k,s})\, x_{k,s} - (1 - \mathbb{1}(x_{k,s} < \Delta_{k,s}))\Delta_{k,s} \\
&= \mathbb{1}(x_{k,s} \geq \Delta_{k,s})\, x_{k,s} - \Delta_{k,s} + \mathbb{1}(x_{k,s} < \Delta_{k,s})\, \Delta_{k,s}.
\end{aligned}$$

Then, using the above equation, multiplying both sides by the indicator random variable $\mathbb{1}(x_{k,s-1} > x_k + D)$ and taking expectations, we find that

$$\begin{aligned}
\mathbb{E}[x_{k,s+1}\mathbb{1}(x_{k,s-1} > x_k + D)] &= \mathbb{E}[\mathbb{1}(x_{k,s-1} > x_k + D)\, x_{k,s}] \\
&\quad - \mathbb{E}[\mathbb{1}(x_{k,s-1} > x_k + D)\, \Delta_{k,s}] \qquad (3.29) \\
&\leq \mathbb{E}[x_{k,s}] - \mathbb{E}[\mathbb{1}(x_{k,s-1} > x_k + D)\, \Delta_{k,s}] \\
&= \mathbb{E}[x_{k,s}] - \mathbb{E}[\mathbb{1}(x_{k,s-1} > x_k + D)]\, \mathbb{E}[\Delta_{k,s}] \qquad (3.30) \\
&= \mathbb{E}[x_{k,s}] - \mathbb{P}(x_{k,s-1} > x_k + D)\, \mu \\
&\leq R \qquad (3.31)
\end{aligned}$$

where equation (3.29) follows from the fact that the events $\{x_{k,s-1} > x_k + D\}$ and $\{x_{k,s} \leq \Delta_{k,s}\} \subset \{x_{k,s} \leq D\}$ are mutually exclusive. Equation (3.30) follows from the fact that $\{\Delta_{k,t}\}_{t=1}^{\infty}$ satisfies the Markov property, thereby making $\Delta_{k,s}$ independent of the random variable $x_{k,s-1}$, which is $\mathcal{F}_{s-2}$-measurable. In addition to the above, it also clearly holds that

$$\begin{aligned}
\mathbb{E}[x_{k,s+1}\mathbb{1}(x_{k,s-1} \leq x_k + D)] &\leq x_k + D + \mathbb{E}\left[\Delta_{k,s-1}^{+}\right] + \mathbb{E}\left[\Delta_{k,s}^{+}\right] \\
&\leq x_k + 3D = R. \qquad (3.32)
\end{aligned}$$

Thus using bounds (3.31), (3.32) and linearity of expectation, it follows that

$$\mathbb{E}\left[x_{k,s+1}\right] \leq R,$$

and therefore we get the contradiction from having just shown that $\sup_{t\geq 1}\mathbb{E}\left[x_{k,t}\right] \leq R < \infty$. From this, it follows that $\mu_k < 0$.

Part 2: Concluding that $\sum_{t=1}^{\infty}\mathbb{E}\left[(\Delta_{k,t} - x_{k,t})^+\right] < \infty$. This part of the proof relies on the first part and the use of concentration inequalities. First, we observe the following trivial lower bound on the inventory $x_{k,t}$:

$$x_{k,t} \geq -\sum_{s=1}^{t-1}\Delta_{k,s} + x_k.$$

We also note that for any constant $\alpha > 0$ and because $\{\delta_{k,t}\}_{t=1}^{\infty}$ is a Markov chain in steady state that

$$\mathbb{P}\left(\frac{1}{t}\sum_{s=1}^{t}(\delta_{k,s} - \bar{\delta}_k) > \alpha\right) \leq C_1 e^{-C_2 t} \tag{3.33}$$

where C_1 and C_2 refer to constants and $\bar{\delta}_k \triangleq \mathbb{E}\left[\delta_{k,1}\right] = \mathbb{E}\left[\delta_{k,2}\right] = \ldots$. The above bound follows from the main result in [?] that proves concentration inequalities for finite-state Markov chains. By using the above inequalities, we can bound the expression

of interest as follows. Letting $\bar{u}_k = \mathbb{E}\left[u^*_{k,1}(\xi_t;\lambda^*)\right]$, we find that

$$\begin{aligned}
\sum_{t=1}^{\infty}\mathbb{E}\left[(\Delta_{k,t}-x_{k,t})^+\right] &\leq \sum_{t=1}^{\infty}\mathbb{E}\left[\left(\Delta_{k,t}+\sum_{s=1}^{t-1}\Delta_{k,s}\right)^+\right] \\
&= \sum_{t=1}^{\infty}\mathbb{E}\left[\left(\sum_{s=1}^{t}\Delta_{k,s}\right)^+\right] \\
&= \sum_{t=1}^{\infty}\mathbb{E}\left[tD\mathbb{1}\left(\sum_{s=1}^{t}\Delta_{k,s}>0\right)\right] \\
&= D\sum_{t=1}^{\infty}t\mathbb{P}\left(\sum_{s=1}^{t}\Delta_{k,s}>0\right) \\
&= D\sum_{t=1}^{\infty}t\mathbb{P}\left(\frac{1}{t}\sum_{s=1}^{t}\Delta_{k,s}-\mu_k>-\mu_k\right) \\
&\leq D\sum_{t=1}^{\infty}t\left[\mathbb{P}\left(\frac{1}{t}\sum_{s=1}^{t}(\delta_{k,s}-\bar{\delta}_k)>-\frac{\mu_k}{2}\right)+\right. \\
&\qquad \left.\mathbb{P}\left(\frac{1}{t}\sum_{s=1}^{t}(-u^*_{k,s}(\xi_s;\lambda^*)+\bar{u}_k)>-\frac{\mu_k}{2}\right)\right] \\
&\leq D\sum_{t=1}^{\infty}t\left(C_1e^{-C_2t\mu_k^2/4}+C_3e^{-C_4t\mu_k^2/4}\right) \qquad (3.34) \\
&< \infty,
\end{aligned}$$

where (**??**) is due to $\mu_k = \bar{\delta}_k - \bar{u}_k$ and (3.34) follows from (3.33), the Chernoff-Hoeffding bound, and the $\mu_k < 0$ as was shown in Part 1. □

The previous Theorem was somewhat restrictive in that it only applies when the demand process is a finite-state Markov chain (so that any given δ_t can only take on a finite number of possible values) but it demonstrated that achieving the constant loss is possible in a non-i.i.d regime. The next Theorem allows an arbitrary support for the random variables $\{\delta_t\}_{t=1}^{\infty}$ but imposes the i.i.d assumption on both the demand process and the $\{\xi_t\}_{t=1}^{\infty}$ process. It turns out that in this setting, the dual price algorithm also achieves, in some sense, the best possible asymptotic rate of optimality.

Theorem 3. *Suppose that Assumption 1 holds and the demand process $\{\delta_t\}_{t=1}^{\infty}$ is bounded, that is $|\delta_t| \leq D$, a.s. for all $t = 1, 2, \ldots$. Suppose further that the random*

vectors in the sequence $\{(\delta_t, \xi_t)\}_{t=1}^{\infty}$ *are i.i.d (however, each vector* δ_t *may depend on* ξ_t *in the same time period). Under such a setting, the* λ^* *selection policy is also asymptotically optimal up to a constant, in the sense that*

$$J_T^*(x) \leq J_T^{S,\lambda^*}(x) \leq J_T^*(x) + C(K),$$

for all integers $T \geq 1$ *and some constant* $C(K)$ *that depends on* K*.*

The proof of this result is similar to Theorem 2 but arguably simpler and, for this reason, is deferred to the Appendix.

3.3.3 Computational methods and interpretation

Up until now, in this section of the chapter, we (implicitly) proposed an algorithm for solving the CMP that is often asymptotically or nearly optimal. We now state a concrete computational method for the problem given in Section 3.2, which we recall is a special case of the general model just analyzed.

First, we will assume that it's possible to *simulate* sample paths of δ_t^i, b_t^i and p_t either through knowing the distributions in question, or being able to find enough relevant historical sample paths. For fixed horizon T, we will say we are given a finite set of S samples whose elements we index by $q \in \{1, \ldots, S\} = [S]$. We write with $(\delta_{k,t,q}^i)_{t=1}^T$, $(b_{k,t,q}^i)_{t=1}^T$ and $(p_{k,t,q})_{t=1}^T$ the qth set of sample paths of the stochastic processes in the CMP. With this notation in hand, we formulate the SAA (sample-average approximation) version of (3.18),

$$\begin{aligned} \text{maximize} \quad & \frac{1}{S}\sum_{q=1}^{S}\sum_{t,k}\left(\sum_i \lambda_k \delta_{k,t,q}^i - \sup_{u_{q,t}\in U_{q,t}} \lambda^\top u_{t,q}\right) - \lambda^\top x \\ \text{subject to} \quad & 0 \leq \lambda_k \leq c_k \qquad\qquad k \in [K] \end{aligned} \tag{3.35}$$

where $U_{t,q}$ is the qth sample of an admissible collateral set at time t, that is

$$U_{t,q} \triangleq \left\{ u_{t,q} \in \mathbb{R}_+^K : \begin{array}{ll} u_{t,q} = \sum_{i=1}^{n} u_{t,q}^i & \\ (h \cdot p_{q,t})^\top u_{t,q}^i \leq p_{q,t}^\top \delta_{t,q}^i, & i \in [N] \\ u_{t,q}^i \leq b_{t,q}^i, & i \in [N] \end{array} \right\}.$$

Using the next trick, it is possible to re-formulate (3.35) as a single linear program (LP) whose number of variables and constraints scales polynomially in S. Now, using strong duality and substituting $u_{t,q} = \sum_{i=1}^{n} u_{t,q}^i$, we notice that

$$\begin{aligned} &\sup_{u_{t,q} \in U_{t,q}} \lambda^\top u_{t,q} \\ &\quad = \inf_{\mu,\nu} \left\{ \sum_i \left(p_{q,t}^\top \delta_{t,q}^i \mu_{t,q}^i + (b_{t,q}^i)^\top \nu_{t,q}^i \right) : \begin{array}{ll} \mu_{t,q}^i \in \mathbb{R}_+, \quad \nu_{t,q}^i \in \mathbb{R}_+^K, & i \in [N] \\ \mu_{t,q}^i (h \cdot p_{q,t})^\top + (\nu_{t,q}^i)^\top \geq \lambda^\top, & i \in [N] \end{array} \right\} \end{aligned} \tag{3.36}$$

where we used $\mu_{t,q}^i$ and $\nu_{t,q}^i$ to denote the dual variables. Thus substituting the above equation into (3.35), we obtain the LP

$$\begin{aligned} \max_{\lambda,\mu,\nu} \quad & \frac{1}{S} \sum_{q=1}^{S} \sum_{t,i} \left(\lambda^\top \delta_{t,q}^i - p_{q,t}^\top \delta_{t,q}^i \mu_{t,q}^i - (b_{t,q}^i)^\top \nu_{t,q}^i \right) - \lambda^\top x \\ \text{s.t.} \quad & 0 \leq \lambda_k \leq c_k && k \in [K] \\ & \mu_{t,q}^i \in \mathbb{R}_+, \quad \nu_{t,q}^i \in \mathbb{R}_+^K, && i \in [N], t \in [T], q \in [S] \\ & \mu_{t,q}^i (h \cdot p_{q,t})^\top + (\nu_{t,q}^i)^\top \geq \lambda^\top, && i \in [N], t \in [T], q \in [S]. \end{aligned} \tag{3.37}$$

We write the optimal solution of (3.37) as $\hat{\lambda}(x; S)$. If we assume the random samples are drawn from their true distribution, then we can show that $\hat{\lambda}$ is a consistent estimator for λ^*.

Proposition 4. *Suppose $\hat{\lambda}(S)$ is an optimal solution to* (3.37) *and λ^* is the unique optimal solution to* (3.18) *(assumed to exist), then for any $\epsilon > 0$ there exists a large*

enough S such that almost surely,

$$\left\| \hat{\lambda}(S) - \lambda^* \right\|_2 \leq \epsilon$$

The proof of this omitted but can be shown using, for example, the law of large numbers. We have just developed a numerical approach for estimating λ^*, we conclude this section by giving a complete algorithm. First we start with a general but technical definition.

Definition 4 (v-greedy algorithm)**.** *Let $v \in \mathbb{R}_+^K$ be a vector of 'asset values' and M a large constant. Any algorithm which at time t selects a control from the set*

$$U_t^*(v) = \underset{u_t^i}{\operatorname{argmax}} \left\{ \sum_i v^\top u_t^i - M \sum_{i,k} c_k (\delta_{k,t}^i - u_{k,t}^i)^+ : \begin{array}{ll} (h \cdot p_t)^\top u_t^i \leq p_t^\top \delta_t^i & i \in [N] \\ 0 \leq u_t^i \leq b_t^i & i \in [N] \end{array} \right\}$$

is called v-greedy.

The M term in Definition 4 ensures that the policy, as far as possible, matches current demand with available collateral. In other words, the greedy policy prioritizes assets, which reduce costs in the current period *with certainty*, over those which are forecast to be valuable by the vector v.

There are two distinct phases to the main algorithm we present:

1. **Offline phase:** Using S Monte Carlo (or historical) samples solve (3.37) and store an optimal solution $\hat{\lambda} := \hat{\lambda}(S)$.

2. **Online phase:** at each period $t \in [T]$, employ a $\hat{\lambda}$-greedy policy by solving

$$\sup_{(u_t^1, \ldots, u_t^N) \in U_t^*(\hat{\lambda})} \sum_i v_{\text{final}}^\top u_t^i \tag{3.38}$$

   where v_{final} is a tie-breaker between elements of $U_t^*(\hat{\lambda})$.

The need for a tie-breaker in (3.38) arises due to the fact that there might be more than one optimal solution, and that $\hat{\lambda}$ is merely an approximation. One choice for

Algorithm	Internal prices	Description
Vol (Volume)	$\mathbb{1}$	Assigns the same value to all assets
DD (Demand-driven)	$c \cdot \bar{\delta}$	Prefers larger expected costs
DPP (Dual-price policy)	$\hat{\lambda}$	Uses approximate dual prices

Table 3.1: Summary of heuristic algorithms in numerical experiment where $\bar{\delta}$ denotes a sample average estimate for demand

v_{final} could be the vector of ones $\mathbb{1}$, i.e. we choose the collateral with maximal value and total volume.

We give our algorithm, which mimics a $\hat{\lambda}$-selection policy, the moniker Dual Price Policy (DPP) and the next section evaluates it on real and synthetic data. In practice, we would run the offline phase every so often to update $\hat{\lambda}$ in case the demand distributions are non-stationary, and this could occur once a few days or weeks. The online phase would occur on a frequent, e.g. daily basis.

3.4 Experiments

We present numerical experiments that benchmark the Dual Price Policy (DPP) against either the optimal offline solution, which can be found 'in hindsight', or similar competing algorithms. The heuristics are all v-greedy policies that differ in their choices of the parameter v. Table 3.1 presents a summary of the algorithms we test. As an example, there is one algorithm, which we call "Vol", that assigns equal weight to all assets by having $v = \mathbb{1}$. In effect, "Vol" is a heuristic that at every time step maximizes the total *volume* of collateral that is taken subject to constraints given by availability, prices and haircuts in each period.

Our experiment consists of backtesting our algorithm on real-world data from a Prime Broker. We also produce at the end, a set of examples demonstrating the sensitivity of dual prices to changes in certain parameters such as the Prime Broker's inventory. By doing this, we gain some intuitive understanding of what determines dual prices and makes them differ from the current collateral pricing methods used in industry.

3.4.1 Empirical Analysis

We gauge the practical value of our algorithm by backtesting it on historical data from Credit Suisse over the sample period February 2nd 2016 to June 1st 2017. This comprises a total of 300 business days and records the growth of a *new business* for the firm. For this reason, the data contains a dynamic set of clients and securities, which grows in size from the beginning of the sample period to the end. For any algorithm to perform well on this data, it needs to frequently update its valuation of asset prices as collateral.

The data we are given to simulate the problem is a daily log of investment positions belonging to different trading desks. Along with this log, we are also provided with relevant historical market prices, borrowing fees and haircuts of all the securities that appear in the data. On aggregate, the data contains information about 201 clients of Credit Suisse who trade in 5896 securities[1]. In the simulation that follows, we will compare our algorithm's performance against the set of benchmark policies given in Table 3.1 but before doing so, we will describe how we processed the data in order to simulate the problem.

Fitting the data to the model

The raw data is given to us as a time series of positions for $N+1$ trading desks. For convenience in explaining the setup, we will index them with $i = 0, 1, \ldots, N$ where the zeroth index refers to the Prime Broker's desk and the remaining indices $1, \ldots, N$ correspond to the clients' desks. As before we index securities with $k = 1, \ldots, K$ and each day in the data with $t = 1, \ldots, T$ (where obviously $K = 5896$ and $T = 300$). With this familiar notation in hand, we will denote with $w^i_{k,t} \in \mathbb{R}$ the position of the ith trading desk in security k on day t. If this value is positive, that means that the desk has a long position, conversely, a negative value indicates a short position.

For the purposes of simulating the real problem, we will assume that Credit Suisse's desk always lends securities to cover 100% of client shorts and finances (with

[1]For confidentiality reasons, both the identity of clients and securities have been obscured in the data-set

cash) 30% of a client's purchases. As a result of this, the sample borrowing demand from the ith desk, on day t in a *security* k (that is an asset with $k > 1$) is succinctly given by the following equation:

$$\hat{\delta}^i_{k,t} := ((w^i_{k,t})^- - (w^i_{k,t-1})^-)^+$$

The above simply means that demand for borrowing either equals the increase, if any, in the client's short position and is zero otherwise.

Cash is treated differently and its demand is given by 30% of the total notional value of client purchases on any given day:

$$\hat{\delta}^i_{1,t} := \frac{3}{10} \sum_{k>1} p_{k,t}((w^i_{k,t})^+ - (w^i_{k,t-1})^+)^+.$$

Next, we assume the client's inventory on any given day is simply equal to their long position. For this reason, we set $\hat{b}^i_{k,t} = (w^i_{k,t})^+$ to be the simulated client inventory, which we will assume is exogenous to our model. For the sake of consistency, in all simulation runs we will set the Prime Broker's initial inventory in asset k to be $(w^0_{k,1})^+$, which means that we always take the broker's initial inventory to be their long position on the first day in the raw data (one may think of this as a random snapshot in time of the broker's inventory). We assume the borrowing costs are given by the notional value of the loan, on the day it was borrowed, multiplied by an interest rate that varies between 1 and 3%. The haircuts on individual securities range from 90% to 97.5%.

Data-driven simulation results

Given the procedure described above, we produced a particular instance of the problem that resembles what happens in reality, especially in this case of a new and growing business. In line with the theoretical prescriptions of this chapter and as described earlier, each algorithm in the simulation maintains a set of internal security prices, v, which is periodically updated (this is the offline phase of an algorithm) and

Data	Update delay	DPP cost ($M)	DD cost ($M)	Volume cost ($M)
20	5	467,608,285.65	468,546,680.08	556,423,670.60
	10	470,111,119.00	471,251,487.79	556,423,670.60
30	5	472,852,176.49	473,705,227.91	556,423,670.60
	10	472,142,653.04	473,095,290.01	556,423,670.60
40	5	474,524,923.12	473,141,339.98	556,423,670.60
	10	474,524,923.12	475,807,040.23	556,423,670.60

Table 3.2: Final costs to the Prime Broker after 300 days.

Data	Update delay	DPP cost ($M)	DD cost ($M)	Vol cost ($M)
20	5	82,511,407.21	82,897,541.31	85,279,283.95
	10	82,773,949.82	83,089,246.13	85,279,283.95
30	5	82,598,667.55	83,007,101.17	85,279,283.95
	10	82,882,992.23	83,253,324.74	85,279,283.95
40	5	83,140,234.28	83,284,857.92	85,279,283.95
	10	83,140,234.28	83,551,965.06	85,279,283.95

Table 3.3: Costs to the Prime Broker after 150 days.

is precisely the cost vector in Definition 4. For running the experiment, we make it so that the internal prices of each algorithm are updated either every 5 or 10 days. We also vary the amount of historical 'training' data used in the offline phase. We remark that this experiment serves a secondary purpose, besides simply being a way evaluating algorithms, which is to figure out practical recommendations on how often to run the offline phase of an algorithm, and how much of the historical data at any given time is relevant, because it may be unnecessarily inefficient, and perhaps even detrimental for performance, to run it every day and use as much past data as possible. It is worth stressing and keeping in mind that some historical data could become outdated due to regime changes or variations in trading patterns. Finally, in implementing the DPP, we treat the entire sequence of historical data as a single scenario so that $S = 1$ in Problem (3.35). The final costs to the Prime Broker after T or $T/2$ days, generated by each algorithm, are shown in Tables 3.2 and 3.3. We also present a plot of how the differences in cost from the current heuristics and our algorithm change over time in Figure 3-1.

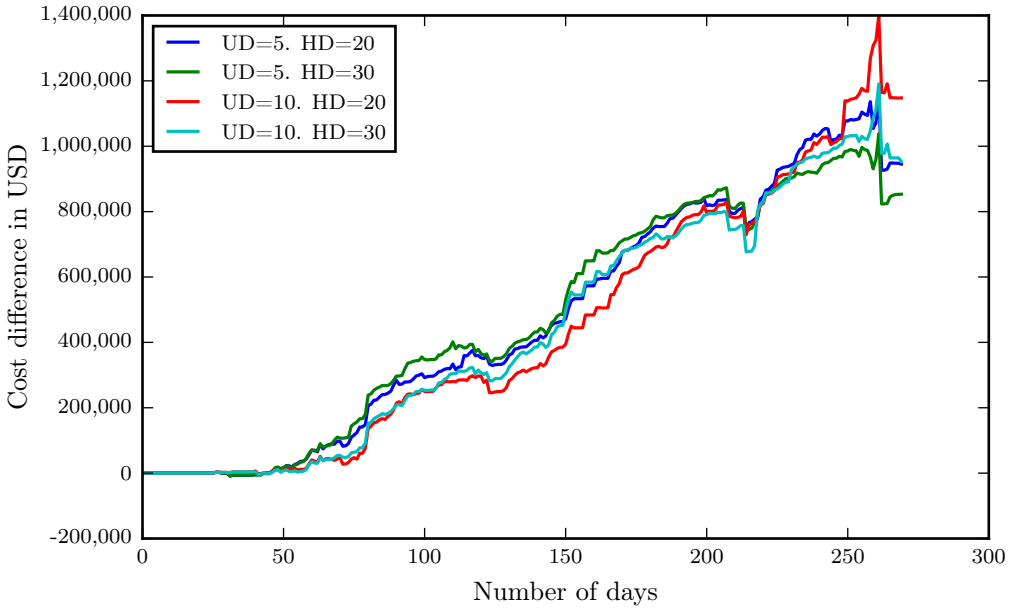

Figure 3-1: The difference in cost between the demand-driven policy and the dual price policy under different algorithm configurations. In the legend, UD is the update delay and HD is the amount of historical data used.

We first observe that the costs generated by the DPP, after 300 days, are consistently lower than the demand-driven policy by up to 1.3M USD, after 1.5 years, in just one (newly established) market for the Prime Broker. Secondly, the spread between the two policies grows nearly linearly over time as noted by the fact that the difference in cost after 150 days was up to 0.5M USD but that figure roughly doubles after 300 days. What Figure 3-1 also shows is that the spread in cost is robust to the configuration of the UD and HD parameters. This suggests that the DD policy increasingly deteriorates relative to the DPP one and it would be fruitful to find out what the difference would be after several years. In any case, this experiment shows that there is enormous practical value to hypothecating collateral based on appropriately chosen internal prices. In fact, the Volume policy which disregards any meaningful pricing scheme, by simply taking the maximum volume of collateral, suffered costs that were on the order of 100M USD higher than the others over the 300 day period.

Finally, another phenomenon we observe is that with more data, the performance

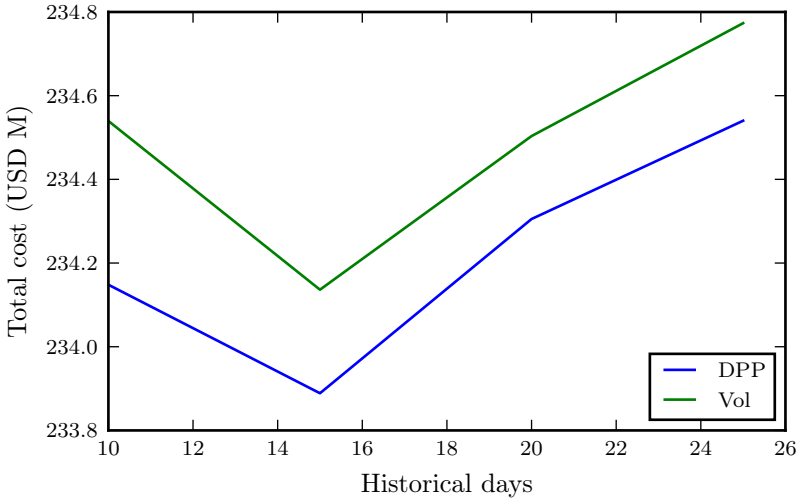

Figure 3-2: Final cost difference between algorithms and Volume-driven baseline after 300 days, as a function of the historical data size used to update internal prices. The update delay used to generate this was 10 days, while the amount of historical days' worth of data was fixed at 20 days.

of both heuristic policies drops slightly (obviously this does not apply to the volume-based policy), as shown in Figure 3-2. What this suggests is that data older than about 15-20 days becomes irrelevant at any given point in time. This is a feature of the dataset, as the market represented by the data is quickly evolving over time given that it was new. However, what this confirms to us is that our policy is robust to the changing conditions in the Prime Broker's 'environment'.

3.4.2 Examining Collateral Prices

Finally to conclude our numerical experiments, we will look at specific examples of $\hat{\lambda}$ under different problem settings and conditions in order to understand what influences collateral prices and how they differ from estimates used in current industry practice. To make the examples more interpret-able, we will assume that all data is deterministic, i.e. that decision maker will keep on seeing a constant stream of demand, exposures and so on for an arbitrary length of time. We identified two main areas where we see a notable difference between our valuation method and others and these are: (i) interdependence between assets and (ii) sensitivity to the Prime

Broker's state. These will now be discussed individually in the following two concrete examples.

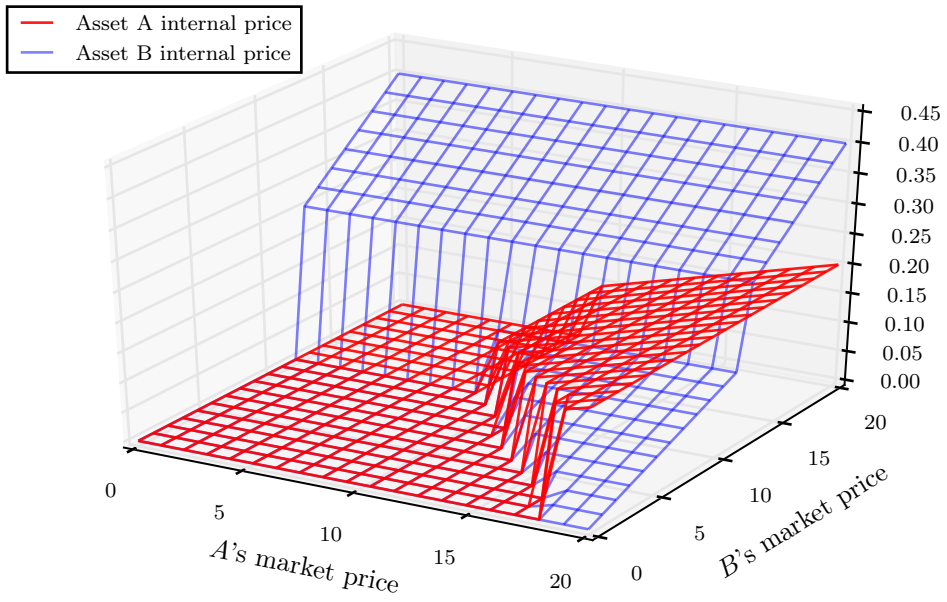

Figure 3-3: Interdependence between two assets' *market* prices and their resulting values in the $\hat{\lambda}$ vector.

Example 1. (Market prices of two assets) This example illustrates how data for one asset can affect the internal price of another asset. For this we assume that there are three assets, and we will refer to them as A, B and C. To keep this as simple as possible, we will make it so that the haircut on all assets is 100%, i.e. no haircut, while their borrowing fees are 10, 20, and 30 basis points, respectively, of their market price. We assume that in the future there is a single client who presents herself with a constant exposure of 13k USD. Moreover, there is a constant supply, in the form of client inventory of the three assets of 100, 1000 and 5000 units. We vary the market price of only assets A and B between 0 and 20 USD, while keeping C's market price fixed at 10, and see what the resulting values for $\hat{\lambda}$ are of those assets.

The values of the resulting internal price for assets A and B, plotted in Figure 3-3, show that merely changing the market price of one asset while holding the other one fixed does perturb the fixed asset's internal price. In the region where both assets have a low market price (roughly $p_A < 10,\ p_B < 15$) both their internal prices get set

to zero. This is because there is additional slack in the exposure constraint when the two assets have a low market price, which makes it easier to re-hypothecate them in future time periods, and reduces their current value. In essence, this reflects the low perceived urgency of re-hypothecating them now.

Of course, what this example highlights are fundamental differences between our pricing scheme and incumbent ones. With the latter approaches, an asset's internal value would be independent of other assets. Current methods only predict the value of an asset in terms of the individual future profit it is expected to bring. This is merely a function of expected borrowing demand for that asset alone, its specific future price and the borrowing fee and is independent of information about other assets. Also, when the market price of both assets are low, our algorithm would prefer to only re-hypothecate asset C, rather than a mixture of it and the other two assets. By contrast, incumbent schemes would pick a mixture of all the three assets.

Example 2. (Inventory and haircut of a single asset) The second example also points out a difference between our method of pricing collateral and current ones. For this we picked a random day from Credit Suisse's data and "re-played" the pricing algorithm on that day taking the previous 20 days as historical data. We focused on one random asset and had its inventory varied between 0 and 4,000 units and its haircut varied between 90% and 100%. The resulting internal price for that asset, from training the DPP algorithm, is plotted in Figure 3-4. A smaller haircut meant that more of that asset could be claimed as collateral in any given period, while a larger inventory also reduced the likelihood of depleting the asset in future periods, thus making it less valuable as collateral. As we see in the plot, the influence of inventory on the asset's internal price's is much greater compared to the haircut. Finally what we learn from this example, is that our pricing scheme crucially accounts for current inventory levels, which is a part of the Prime Broker's state. Incumbent pricing methods ignore this element of the decision and this explains why they demonstrate worse performance.

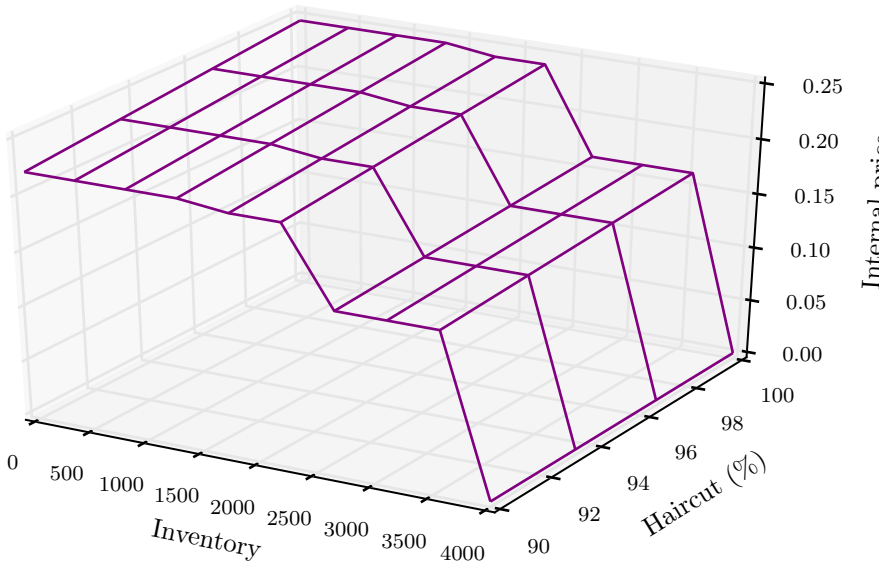

Figure 3-4: Effect of inventory quantity and haircut on the internal price of a single random asset, picked from a random day, in Credit Suisse's data

Chapter 4

Deep Reinforcement Learning in Financial Engineering

4.1 Introduction

Reinforcement Learning (RL) is a general methodology for addressing intractable sequential decision problems that has gained much attention in recent years due to the big data revolution. RL has been extensively applied to domains such as robotics, engineering, business and artificial intelligence [Mao et al., 2016, Evans and Gao, 2016]. One of the greatest successes of RL was demonstrated in the game of Go [Silver et al., 2016], where an AI Go player trained with deep neural networks beat Lee Sedol, the then world champion four games to one in 2016. This groundbreaking and world-famous event brought RL technology to mainstream attention.

Not too long before AlphaGo was developed, researchers demonstrated that a generic reinforcement learning algorithm, called Deep Q-Learning, could be applied to a diverse set of Atari arcade games [Mnih et al., 2013]. The algorithm was so general that it could learn to play seven completely different games well, from scratch, using only screen pixels as input and no other prior knowledge about any of the games, except for the available actions and what the score is in every frame. The remarkable aspect to all this was the complete generality of the Deep Q-learning algorithm, which raised the question if we were truly on our way to genuine AI wherein agents can learn

how to perform tasks by themselves by purely learning from experience.

What enabled the advent of Deep Reinforcement Learning was recent advances in the area of deep neural networks [LeCun et al., 1995, Goodfellow et al., 2016]. In fact, a core component of the Deep Q-learning algorithm is a convolutional neural network, not radically different from those used in computer vision problems, except for being somewhat smaller. The fundamental idea is that, instead of running machine learning on state inputs directly, a deep neural network is utilized as a feature extractor, where the intermediate (usually referred to as 'hidden') layers produce a set of summary features that can be used to better explain raw input data. In the case of Atari games, the hidden layers learn relevant features of frames during gameplay.

Of course, the study optimal control goes back decades. There is an especially rich literature on the general question of how to approximately solve MDPs, unsurprisingly called Approximate Dynamic Programming (ADP). The terms ADP and RL are basically synonymous, but the former term captures methods within the OR and decision science community (see for example [Bertsekas and Tsitsiklis, 1995, Bertsekas and Ioffe, 1996, De Farias and Van Roy, 2003, 2004]).

There is an interesting connection between current, state-of-the-art RL algorithms and its ADP forefathers. With the latter, a common approach to solve intractable MDPs is to approximate the value function using a *weighted, linear combination* of *basis functions*. Perhaps an oversimplified way of looking at these algorithms is they perform a type of linear regression to tune the weights so that the value function approximation is close to the optimal value function. The key differences between ADP methods lie in how exactly they perform this, roughly-speaking, "linear regression" and in how they collect the data for training (i.e. which states and rewards are sampled to reduce correlation in the training data). In the case of TD Learning [Bertsekas and Ioffe, 1996], and Q-learning [Watkins and Dayan, 1992], this regression is performed more or less directly. That is, we update the current estimate of the value function using the outcome observed of taking a certain action in a state. More precisely, we perform a type of gradient update on the basis functions weights to minimize the TD error. It is natural to see how one could then extend

such an approach to more complex approximation architectures represented by deep convolutional networks via backpropagation.

One immediate question that arises is how easily we can avoid the need to use hand-crafted basis functions, and opt instead for deep architectures that learn representations, essentially, by themselves. This would address a key drawback with ADP methods that the practitioner needs to design, in a an ad-hoc fashion, a good set of basis functions a priori without knowing if they can potentially even capture the optimal value function. What's missing though when we try to combine ADP methods work with Deep Learning is that sometimes the implementation becomes less clear cut. For example, with ALP (Approximate Linear Programming, De Farias and Van Roy [2003]), we rely on having a linear architecture so as to be able to solve a LP. In this chapter we will find ways to address this issue in the special problem class of optimal stopping.

In spite of the promising progress seen in RL and ADP, it is still not completely understood why some algorithms work well and others don't. Moreover, it is generally hard to derive useful theoretical results. For example, Gu et al. [2016] show that the vanilla DQN algorithm can be beaten by a simpler Monte Carlo tree search. In practice, applying RL technology is a tricky business since every practitioner would need a good understanding of both statistics and optimal control, and have a reliable set of benchmark results from which to know what methods work well and when. In practice it's hard to ensure even the last point [Henderson et al., 2017].

In the this chapter, we have in mind the following two goals to address several difficulties encountered in RL. We wish to import some state of the art algorithms in ADP and combine them with deep architectures, to see how effective such an approach could be and whether we can really do away with hand-crafted basis functions, while achieving improved performance. Secondly, we focus on model-based reinforcement learning algorithms (where we know the MDP dynamics) in the context of problems with special structure. We want to see how popular approaches such as Policy Gradient can be tailored to problems with partly linear dynamics or where the reward function is known to satisfy special properties such as convexity.

Applications The applications considered in this chapter are in financial engineering because, for those types of problems, MDP dynamics are usually specified ahead of time and are known. We will study the problem of option pricing, which is a type of optimal stopping problem. By leveraging the special structure inherent in stopping problems, we are able to design neural network architectures to approximate the option's price, as well as tailor other RL algorithms for the task. We will also study a new framework for portfolio optimization problems (which can also work in other domains such as inventory management) and see that, in that case, specially designed RL algorithms can achieve state of the art performance. We provide a few early theoretical results to support the encouraging experiments.

In greater detail, we make the following contributions

1. **New Option Pricing Method:** we demonstrate a practical method for computing a tight dual upper bound on an option's price. In order to make the method scalable, we find alternative martingale representations that are cheaper to compute, and can thus handle a deep neural network as a core component.

2. **Portfolio Optimization with RL:** we analyze the application of Policy Gradient methods to a general class of portfolio and inventory problems. We show that for this class of problems, neural nets can both in theory and practice learn *near-optimal* policies and outperform other heuristics, including those based on convex optimization.

3. **Application to Optimal Execution and Algorithmic Trading:** we demonstrate the applicability of our RL framework to a few real, practical applications in optimal execution and automated trading.

In the following two main sections, we will describe each of the two problem classes, the methodology we develop/analyze to them and results from numerical experiments.

4.2 Option Pricing

In this section, we study option pricing, or more generally optimal stopping problems. These problems exhibit a special structure in that they rely almost entirely on *exogenous* state. In other words, our decision in every period is either to stop observing the process and collect a terminal reward, or to continue waiting and defer the opportunity to stop to a later time. In this way, there is a rather simple trade-off inherent to the decisions, and thus this type of problem represents one of the most basic multi-period, stochastic control problems. As such, this forms a natural starting point for us to attempt to combine Deep Learning with existing ADP methods, and see what is possible achieve. For ease of exposition, we will refer to 'option pricing' and 'optimal stopping' interchangeably, as well as the phrases 'exercising the option' and 'stopping the process'.

We will focus on two famous classes of ADP methods for solving stopping problems for this research. These are:

1. *Approximate Value Iteration:* Famous algorithms for this include ones in Longstaff and Schwartz [2001], Tsitsiklis and Van Roy [1999]. The fundamental idea is we carry out a backward induction to estimate the 'value-to-go' from not stopping based on the current state at every point in time. After fitting such "continuation-value" functions, a lower bound on option's price can be estimated by simulating a greedy algorithm that stops as soon as the current payoff exceeds estimated value from continuing.

2. *Dual Martingale Methods:* By contrast these methods approximate the option's price by computing an unbiased estimate of an upper bound. The bound is gotten through information relaxation, i.e. allowing the policy to look ahead when it should not be able to do so, but penalize for these violations in expectation (much like a duality approach). This method has widely been discussed in Haugh and Kogan [2004], Rogers [2002].

Of course, our goal will be to make progress on both these fronts, so that practically speaking we will end up with tighter confidence intervals on an option's price. This

would have significant practical value in the financial markets given that an accuracy improvement in valuing an option, even of a few basis points, can make a big difference to the bottom line considering that large financial institutions trade millions of dollars' worth of options in a day.

4.2.1 Introduction and problem setup

We present here a general formulation of the optimal stopping problem, under a Markov process, over a finite time horizon T. Let $\{x_t : 0 \leq t \leq T\}$ be an $\mathbb{R}^n$-valued continuous-time Markov process, representing for example a vector of asset prices. We denote with $\mathbb{F} = \{\mathcal{F}_t : 0 \leq t \leq T\}$ the natural filtration generated by the process, that is $\mathcal{F}_t = \sigma\left(x_s : 0 \leq s \leq t\right)$ and is the σ-field generated by the process's trajectory up to time t. We are given a payoff function $g : \mathbb{R}^m \mapsto \mathbb{R}$ that maps each state to a reward. That is, if we were to stop the process at time t in the state x_t, we would earn an undiscounted reward of $g(x_t)$.

Our goal is to stop the process at the most profitable time, that is to solve the optimization problem

$$J_0^\star(x) \triangleq \sup_{\tau \in \mathcal{T}} \mathbb{E}\left[e^{-r\tau} g(x_\tau) \mid x_0 = x\right],$$

where the optimization is over $\mathcal{F}_t$-stopping times τ, taking values in the set $\mathcal{T} \subset [0, T]$ and r is a continuous discount rate. For simplicity, we'll assume that $T \in \mathcal{T}$ always. In studying this problem it will be useful to define a general value function of time and state, namely,

$$J_t^\star(x_t) \triangleq \sup_{\tau \in [t,T] \cap \mathcal{T}} \mathbb{E}\left[e^{-r(\tau - t)} g(x_\tau) \mid x_t\right].$$

Intuitively, this measurable function gives the remaining optimal expected *value* to an agent given that she has not stopped prior to time t and the current state is x_t. We refer to the sequence $J_t^\star$ as the optimal value function. It can be shown that the optimal value function is a supermartingale and, moreover, is the Snell envelope of the payoff process $e^{-rt} g(x_t)$.

If $\mathcal{T} = [0, T]$, namely the set of possible times to stop is a continuous interval, the option is called *American*, whereas if $\mathcal{T}$ is finite, the option is called *Bermudan* – we will focus mainly on the latter type.

For Bermudan options, we will now introduce some additional notation. Since we are working with discrete time periods, we can see that the stopping time τ in our optimization problem can be expressed as a policy $\pi \triangleq (\pi_t : \ t \in \mathcal{T})$, namely a sequence of $\mathcal{F}_t$-measurable functions. Each function $\pi_t : \mathbb{R}^n \mapsto \{0, 1\}$ determines an action at time t, which is either to stop (1) or continue (0). Without loss of generality, we will require that $\pi_T(x) = 1$ for all $x \in \mathbb{R}^n$. We define the class of all such admissable policies as Π.

For a given policy π, if we were to start in a state $x \in \mathbb{R}^n$ at time t, the expected reward we would earn is

$$J_t^\pi(x) \triangleq \mathbb{E}\left[e^{-r(\tau_\pi(t)-t)} g(x_{\tau_\pi(t)}) \mid x_t = x\right],$$

where $\tau_\pi(t) \triangleq \min\{s \in \mathcal{T} : s \geq t, \ \pi_s(x_s) = 1\}$. Our goal will be to find a policy which maximizes $J_t^\pi(x)$ for all periods t and states x, denoted by $\pi^\star$. The maximum value achieved by the optimal policy is then precisely $J_t^\star(x)$ defined earlier.

Since $\mathcal{T}$ is assumed to be finite for a Bermudan option, we can express it as the sequence $\mathcal{T} = \{t_1, t_2, \ldots, t_K\}$ for some $K \in \mathbb{N}$ (corresponding to the number of exercise opportunities) where $t_K = T$. Because the optimal policy solves Bellman's equation, we know that it has a simple characterization, for every $k \in [K-1]$ in terms of the equation:

$$\pi_{t_k}^\star(x) = \begin{cases} 1, & g(x_{t_k}) \geq e^{-r(t_{k+1}-t_k)} \mathbb{E}\left[J_{t_{k+1}}^\star(x_{t_{k+1}}) \mid x_{t_k} = x\right] \\ 0, & \text{otherwise,} \end{cases}$$

most of the heuristic policies we consider in this chapter are also greedy, but only with respect to an estimate of $J^\star$. This leads us naturally to the following section on finding lower bounds.

4.2.2 Lower Bounds via Approximate Value Iteration

We start by illustrating one of the most basic examples of ADP, as well as the most widely applied in practice. We will find a sub-optimal exercise policy for Bermudan options by approximating the optimal value function at every time period. The idea for this is proposed in [Tsitsiklis and Van Roy, 1999, Longstaff and Schwartz, 2001]. The main purpose of this section is to demonstrate the value that Deep Learning can bring, if we are ready to accept longer computation times.

Generally speaking, the advantage that optimal stopping buys us over other problems, in the context of RL, is that the distribution of states visited at a any time is independent of the policy. In other words, the state observed is from an exogenous process. From a machine learning standpoint, it is therefore easy enough to obtain i.i.d samples of states with which to fit a value function approximation. As such we can rely on having better estimates of the value function compared to what we might achieve in other problems, which makes this method particularly attractive.

We will first assume a parameterization for a family of *continuation value* function approximations

$$\mathcal{C} = \{C^\theta : [K-1] \times \mathbb{R}^n \mapsto \mathbb{R} : \theta \in \mathbb{R}^d\},$$

where we recall that K is the number of exercise opportunities, so that we have a continuation value estimate at every possible exercise time. The actual continuation value is defined to be $C(k,x) = \mathbb{E}\left[J^\star_{t_{k+1}}(x_{t_{k+1}}) \mid x_{t_k} = x\right]$, which is what we aim to approximate.

In the existing work, the following architecture is used: $C^\theta(k,x) = \sum_{i=1}^B \theta_{i,k}\varphi_i(x)$ where $\varphi_1(.), \ldots, \varphi_B()$ are a finite set of *basis* functions. In other words, the approximation is given as a linear combination of basis functions. In general, the basis functions are chosen manually and arguably in an ad-hoc manner. For example, if the state is a vector of real numbers, x, a possible choice of basis functions would be all possible monomials $x_1^{q_1} \ldots x_n^{q_n}$ up to a certain degree p, so that $0 \leq q_i \leq p$ for all i. For this section we will focus specifically on the following $n+2$ basis functions: $\varphi_{n+1}(x) = g(x)$, $\varphi_{n+2}(x) = 1$ and then $\varphi_i(x) = x_i$ for $i = 1, \ldots, n$. In other words, we

approximate the continuation value using a linear combination of the current payoff, the prices of underlying assets and a constant.

Simulate M independent paths $\{x^j_{t_1}, \ldots, x^j_{t_K}\}$ for $j = 1, \ldots, B$ of the Markov process ;

Set the terminal value function estimate $\hat{J}_K(x^j_{t_K}) \leftarrow g(x^j_{t_K})$ for $j = 1, \ldots, B$;

for *each exercise opportunity* $k = K - 1$ *down to* 1 **do**

Fit parameters θ_k by solving

$$\theta_k \leftarrow \underset{\theta \in \mathbb{R}^M}{\operatorname{argmin}} \ \frac{1}{B} \sum_{j=1}^{B} \left(C^\theta(k, x^j_t) - \hat{J}_{k+1}(x^j_{t_{k+1}}) \right)^2 . \tag{4.1}$$

for *each sample path* $j = 1, \ldots, B$ **do**

if $g(x^j_{t_k}) \geq e^{-r(t_{k+1} - t_k)} C^{\theta_k}(k, x^j_{t_k})$ **then**

$J_k(x^j_{t_k}) \leftarrow g(x^j_{t_k})$

else

$J_k(x^j_{t_k}) \leftarrow J_{k+1}(x^j_{t_{k+1}})$

end

end

end

Output the continuation value approximations $C^{\theta_k}(k, .)$, that will be employed by a greedy algorithm.

Algorithm 1: Approximate value iteration for optimal stopping via backward induction.

Now that we have defined the basis functions, it is possible to tune the weights $\theta_{i,k}$ to fit the continue value function approximations. Algorithm 1 shows how this done exactly using backward induction. The output of the algorithm is a sequence of approximations $C^{\theta_1}(1, .), \ldots, C^{\theta_K}(K, .)$, which are then queried by a simple greedy policy defined according to:

$$\hat{\pi}_{t_k}(x) = \begin{cases} 1, & g(x_{t_k}) \geq e^{-r(t_{k+1} - t_k)} C(k, x_{t_k}) \\ 0, & \text{otherwise.} \end{cases}$$

Simulating the above sub-optimal policy, and calculating its average payoff, would then give an unbiased estimate for a lower bound on the option's price. In essence, Algorithm 1 solves a sequence of regression problems, where we regress the current state (or features of the state) against an estimate of the continuation value. If we use the linear architecture, described at the beginning, namely $C^\theta(k,x) = \sum_{i=1}^{B} \theta_{i,k}\varphi_i(x)$ then these regression problems can be solved efficiently with OLS.

A natural question at this point is if we can move beyond OLS and consider more complex architectures using neural networks. In the following numerical experiment, we are going to compare the standard least-squares value iteration (LSVI) approach from Longstaff and Schwartz [2001] with a neural network extension. Simply put, in the second method we will take $C^\theta(k,x)$ to be a neural network whose input layer is precisely the tuple of basis functions used in LSVI, and where the output layer is a scalar value representing the continuation value. We then minimize the loss function in Problem 4.1 via some form of stochastic gradient descent and backpropagation. The full details will be given shortly.

Clearly an advantage with the new approach is that we can potentially fit the value function more closely, and thus we should achieve tighter lower bounds on the option price. The potential downsides are that we now are faced with a non-convex problem and thus might reach 'bad' local minima when solving (4.1), among other issues during optimization. Finally, if we don't use enough sample trajectories in Algorithm 1 there is a potential for overfitting, which is less of an issue with simpler models such as least squares. In the experiment that follows, we will see that in practice there is a tangible benefit to using a neural network if we are willing to accept longer computation times for Algorithm 1.

Numerical Experiment

For the remainder of this chapter, we will study the following canonical option pricing problem and compute bounds on its value. This particular optimal stopping problem will reappear in later sections. We consider a Bermudan option over a calendar time horizon of $T = 3$ years, defined on n assets. There are a total of $K = 54$ exercise

opportunities at calendar times $\delta, 2\delta, \ldots, \delta K$ where $\delta \triangleq T/K$. The payoff is that of a call option on the maximum of n non-dividend-paying assets with an *up-and-out* barrier. For modeling asset prices, we will use the Black-Scholes framework. We shall assume that the risk-neutral asset price dynamics for each asset i are given by a Geometric Brownian motion. That is the price process $p_{i,t}$ of the ith asset satisfies the stochastic differential equation,

$$dp_{i,t} = rp_{i,t}dt + \sigma_i p_{i,t} dB_{i,t}$$

with some initial value $p_{i,0}$, where $B_{i,t}$ is a standard Brownian motion and $r = 0.05$ is the continuously-compounded risk-free interest rate. We let ρ_{ij} be the correlation between the price movements of the ith and jth assets. To keep this experiment simple and in line with Desai et al. [2012] we will assume that $\rho_{ij} = 0$ for $i \neq j$ and that $\sigma_i = 0.2$ for all i. In other words, the volatility of every asset's returns are 20% and the returns are independent between assets. Furthermore, every has the same initial price $p_{i,0} = p_0$.

This Bermudan option has a barrier feature, which means that if the maximum of the asset prices exceeds some pre-specified threshold B, the option gets *knocked-out* and becomes worthless thereafter. For our experiment, we set $B = 170$ and the initial price p_0 will take on values much lower than this. In order for us to formally define the payoff function, we will need to fully describe the state in terms of a knock-out indicator y_k, which equals one if the option has been knocked out and is zero otherwise. In particular the discrete-time binary-valued stochastic process y_k evolves according to the recursion

$$y_k = \begin{cases} \mathbb{1}\left(\max_i p_{i,0} \geq B\right) & k = 1 \\ \mathbb{1}\left(\max_i p_{i,\delta k} \geq B\right) \vee y_{k-1} & k > 1. \end{cases}$$

A state during the kth exercise period is defined as the tuple $x_k \triangleq (p_{k\delta}, y_k)$, and the

corresponding value of the payoff function is given by

$$g(p_{k\delta}, y_k) = \left(\max_{i=1,\ldots,n} p_{i,\delta k} - S \right)^+ (1 - y_k).$$

Given this particular payoff function it's possible to use the same basis function architecture described earlier in implementing both LSVI and its neural network extension. To derive the lower bounds, we sampled 200,000 independent trajectories of the price processes. Using this same set of sample paths and a common set of basis functions (alluded to earlier), we computed the following heuristic policies on a machine with a single CPU and 32 GB of RAM:

- **LS:** The standard Longstaff-Schwartz method as described in Algorithm 1 where the approximation architecture is a linear combination of the basis functions.
- **NN:** An extension of the previous method, where the approximation architecture is a 4 layer feedforward neural network with sigmoid activations between the hidden layers. Batch normalization (explained later) is also applied to the output of the activations. There are 20 hidden neurons in each layer and the input to the network consists of the basis function values. We train a separate network in each iteration of Algorithm 1 using the Adam Kingma and Ba [2014], with a minibatch size of 32. We run the optimization for 50 epochs over the training data.
- **PO:** Another heuristic policy derived from the value function approximation given by the pathwise optimization method in Desai et al. [2012]. This is supposed to be an improvement over Longstaff-Schwartz so we include it for comparison.

After training each policy, we evaluate it ten times on a different set of 10,000 independent sample paths. The mean payoff from each policy, as well as the standard error over the 10 evaluations, are shown in Table 4.1. In addition, we show the absolute and relative improvement over the two benchmarks in Table 4.2. We also report the time taken to train in each policy in minutes. More precisely, in the case of LS

$n = 4$ assets

p_0	90			100			110		
Method	LS	NN	PO	LS	PO	NN	LS	NN	PO
Mean	32.73	34.06	33.01	40.74	42.72	41.54	46.87	49.18	48.16
S.E.	0.07	0.08	0.06	0.05	0.07	0.05	0.06	0.04	0.04 0.06
Time (min)	6.67	452.22	136.08	6.53	437.87	142.18	6.15	66.98	125.77

$n = 8$ assets

p_0	90			100			110		
Method	LS	NN	PO	LS	PO	NN	LS	NN	PO
Mean	43.10	45.08	44.11	49.00	51.24	50.25	52.43	54.16	53.49
S.E.	0.04	0.05	0.04	0.02	0.03	0.02	0.04	0.05	0.01
Time (min)	6.91	480.61	160.53	6.90	603.86	154.23	5.96	50.82	155.72

$n = 16$ assets

p_0	90			100			110		
Method	LS	NN	PO	LS	PO	NN	LS	NN	PO
Mean	49.83	51.57	50.87	52.79	54.55	53.62	54.54	55.81	55.14
S.E.	0.02	0.04	0.02	0.03	0.02	0.02	0.02	0.03	0.04
Time (min)	7.89	477.70	236.08	7.05	432.57	212.08	6.41	50.33	207.47

Table 4.1: Lower bound estimates on option price from the heuristic policies as a function of the initial price p_0 and number of assets n.

n	p_0	(NN)-(PO)	(%)	(NN)-(LS)	(%)
4	90	1.05	3.19	1.33	4.07
	100	1.18	2.83	1.98	4.85
	110	1.01	2.10	2.30	4.92
8	90	0.96	2.18	1.98	4.59
	100	0.99	1.97	2.24	4.57
	110	0.67	1.25	1.73	3.31
16	90	0.68	1.34	1.74	3.49
	100	0.92	1.71	1.76	3.33
	110	0.67	1.21	1.27	2.34

Table 4.2: Relative value of the NN algorithm as a function of the initial price p_0 and number of assets n.

this is the time taken to solve the least squares problems and for the NN extension, this is the time taken for Adam to converge after the 50 epochs over the training data.

As we can see, there is a significant (greater than 200 basis point) improvement on the lower bound from fitting the continuation value function using a more complex neural network architecture as opposed to OLS with basis functions. We also see an improvement (albeit a more modest one) on the PO method, which is based on a different way of regressing the value function [see Desai et al. [2012] for details]. When implementing our algorithm, we avoided potential issues such as overfitting by using enough sample paths, improved optimization with batch normalization and through the use of the state-of-the-art Adam algorithm.

While this experiment is conceptually simple, it does demonstrate tangible value from deep learning. By a slight enough modification to a least squares based algorithm, we obtain encouraging results and outperform two well-established exercise policies. Of course, all this is at the expense of longer computation times, which may be managed by utilizing more hardware (more CPUs or GPUs), but this is beyond the scope of the experiment. In the next section, motivated by these encouraging results, we will tackle duality methods for computing upper bounds and see how we could leverage deep learning there.

4.2.3 Upper Bounds via Martingale Duality

In this section, we will focus again on approximating the price of an option, but only this time through unbiased estimates of an upper bound. We will again explore how deep learning can be exploited to improve existing approximations. Recall that computing a lower bound on an option price is, in principle, straightforward as all that is required is to estimate the mean payoff from a (sub-optimal) policy. On the other hand, getting an upper bound boils down to solving a new, tractable stopping problem, in which the non-anticipativity constraint is relaxed and the payoff function is modified in a carefully chosen way. The optimal value of the new stopping problem is an upper bound to the original one.

To be more precise, let us assume that the set of exercise times $\mathcal{T}$ is finite, so we are again dealing with Bermudan options. We will denote the individual exercise times as $0 = t_0 < t_1, \ldots, t_K \leq T$. We will fix an arbitrary $\mathcal{F}_t$-measurable martingale $M = \{M_t,\ t \geq 0\}$ and define the tractable approximation problem

$$U(x, M) \triangleq \mathbb{E}\left[\max_{s \in \mathcal{T}} \left(e^{-rs} g(x_s) - M_s\right) \;\middle|\; x_0 = x\right] \tag{4.2}$$

which is computationally easy to estimate since s is *not* a stopping time but rather the maximal index over a sample trajectory of values. When the set $\mathcal{T}$ is finite, it's clear how to estimate the above quantity with a simple Monte Carlo algorithm, where each iteration's time complexity is linear in $K = |\mathcal{T}|$.

The following well-known weak duality result is crucial in what will follow, and we show the proof because it's short but instructive.

Lemma 12 (Weak duality)**.** *For any martingale M adapted to the filtration $\mathbb{F}$ and any starting state $x \in \mathbb{R}^n$,*

$$J_0^\star(x) \leq U(x, M).$$

Proof. Let $\tau^\star$ be an optimal stopping time taking values in $\mathcal{T}$. We then have

$$\begin{aligned} J_0^\star(x) &= \mathbb{E}\left[e^{-r\tau^\star} g(x_{\tau^\star}) \mid x_0 = x\right] \\ &= \mathbb{E}\left[e^{-r\tau^\star} g(x_{\tau^\star}) - M_{\tau^\star} \mid x_0 = x\right] \\ &\leq \mathbb{E}\left[\max_{s \in \mathcal{T}} e^{-rs} g(x_s) - M_s \mid x_0 = x\right], \end{aligned}$$

where the second equality follows from the optional stopping theorem and the fact that $\tau^\star$ is bounded. The inequality above follows from relaxing the non-anticipativity constraint and allowing exercise policies access to the entire sample trajectory. □

The result above is analogous to duality in optimization (hence the name), where we remove difficult constraints and replace them with penalty terms in the objective for penalizing them. For example, we could choose an obvious martingale such as $M_t \equiv 0$, and this will give a loose, yet valid bound. In fact, the above upper bound

is tight, which follows from the following strong duality result:

Theorem 4.2.1 (Strong duality)**.** *For any starting state $x \in \mathbb{R}^n$, there exists a zero-mean martingale $M^\star$ such that*

$$J_0^\star(x) = U(x, M^\star).$$

Moreover, $M^\star$ is the martingale part of the Doob-Meyer decomposition of the supermatingale $\{J_{t_k}^\star(x_{t_k}) : k \in [K]\}$, that is

$$J_{t_k}^\star(x_{t_k}) = J_0^\star(x) + M_{t_k}^\star - A_{t_k}^\star, \quad k = 1, \ldots, K$$

where $A^\star$ is a previsible increasing discrete process.

The work in Rogers [2002] includes a proof of strong duality and we omit it here. The fact that we have a potentially tight upper bound given in terms of any martingale M, motivates us to consider the intractable *dual problem*:

$$\inf_{M \in \mathcal{M}} U(x, M) \tag{4.3}$$

which is a search over the space of martingales adapted to $\mathbb{F}$, denoted by $\mathcal{M}$. As is apparent from Lemmas 12 and 4.2.1, the optimal value to (4.3) is the option's price $J_0^\star(x)$. Our focus will be on approximately solving this dual problem by restricting our search to a more manageable space of martingales.

Pathwise Optimization

In order to see how deep learning could be leveraged to approximately solve the dual problem in Equation (4.3), we describe the pathwise optimization method [Desai et al., 2012]. Notice that, given knowledge of the optimal value function $J_{t_k}^\star(.)$, we can determine the optimal martingale in Theorem 4.2.1 as follows:

$$M_{t_k}^\star = \sum_{p=1}^{k} e^{-rt_p} \left(J_{t_p}^\star(x_{t_p}) - \mathbb{E}\left[J_{t_p}^\star(x_{t_p}) \mid x_{t_{p-1}} \right] \right). \tag{4.4}$$

The above equation follows from Doob's decomposition theorem. In practice, we might need to estimate the conditional expectation above via Monte Carlo simulation.

For each exercise period k, let $\mathcal{J}_k$ be the space of $\mathcal{F}_{t_k}$-measurable functions $J : \mathbb{R}^n \mapsto \mathbb{R}$, which we assume includes the payoff function $g(.)$. We define $\mathcal{P}$ as the set of functions $J : [K] \times \mathbb{R}^n \mapsto \mathbb{R}$, such that for each $k \in [K]$, the function $J_{t_k}(.) \triangleq J(k, .)$ belongs to $\mathcal{J}_k$. Thus, it is clear that if by an abuse of notation, we wrote the value function $J^\star_{t_k}(.)$ as $J^\star_k$ (namely we only defined the optimal value function during discrete exercise times), it would follow that $J^\star \in \mathcal{P}$. The idea of pathwise optimization is to parameterize martingales in terms of an arbitrary function $J \in \mathcal{P}$, by defining the process

$$(MJ)_{t_k} = \sum_{p=1}^{k} e^{-rt_p} \left(J_{t_p}(x_{t_p}) - \mathbb{E}\left[J_{t_p}(x_{t_p}) \mid x_{t_{p-1}} \right] \right). \tag{4.5}$$

Subsequently, we shall refer to this as a *discrete representation* because we express the process as a finite sum of random variables. Since the above is indeed a martingale, we can formally define the following upper bound operator, given in terms of a value function $J \in \mathcal{P}$ and the starting state x:

$$(F_0^D J)(x) \triangleq \mathbb{E}\left[\max_{s \in \mathcal{T}} \left\{ e^{-rs} g(x_s) - (MJ)_s \right\} \;\middle|\; x_0 = x \right].$$

Thus we can focus our attention on finding the minimum upper bound over all value functions in the space $\mathcal{P}$, namely the following dual problem:

$$\inf_{J \in \mathcal{P}} (F_0^D J)(x) \tag{4.6}$$

whose optimal solution is $J^\star$, as noted from Equation 4.4 and Theorem 4.2.1, and where the optimal objective value is $J^\star_0(x)$. Unfortunately, (4.6) remains an infinite-dimensional optimization problem for which there aren't any solution methods, and we discuss ways to address that.

In order to find a tractable approximation to Problem (4.6), we will optimize over a smaller space of functions $\hat{\mathcal{P}} \subset \mathcal{P}$ which is compactly parameterized by some real

vector $\theta \in \mathbb{R}^p$. That is we consider a restricted family

$$\hat{\mathcal{P}} = \left\{ J^\theta \in \mathcal{P} : \theta \in \mathbb{R}^p \right\},$$

where, for the purposes of this chapter, J^θ could denote the output layer of a deep feedfoward neural network whose input is the state vector $x_t \in \mathbb{R}^n$ at time t. In that case, θ denotes the sequence of all parameters in the network (including all hidden layers). Alternatively, and in Desai et al. [2012], J^θ is assumed to be a linear combination of p basis functions $\Phi = \{\phi_1, \ldots, \phi_p\} \subset \mathcal{P}$, that describe features of the state. In other words, they would define a candidate value function as

$$J^\theta(x) = \sum_{k=1}^{p} \theta_k \phi_k(x), \tag{4.7}$$

which describes the same method of approximating the continuation function in the Longstaff-Schwartz algorithm of the previous section. In fact, later on, we will use the same set of basis functions as we did in Section 4.2.2.

In any case, we refer to both ways of approximating the value function as an *approximation architecture*. The advantage of using the neural network, as we saw last time, is that we do not need to design basis functions by hand, so that the same architecture can (hopefully) be reused in a variety of stopping problems. At the same time, a major drawback of the neural network is that we might need many more parameters, and the function $J^\theta(.)$ becomes non-linear and non-convex in θ.

Once we have decided upon a parameterization J^θ and thus fixed a family $\hat{\mathcal{P}} \subset \mathcal{P}$, we may consider the problem

$$\inf_{\theta \in \mathbb{R}^p} (F_0^D J^\theta)(x) \tag{4.8}$$

which is an unconstrained optimization over p real numbers and thus can be tackled with numerical techniques. This is what we call the *pathwise optimization* problem. We can see that solving this problem provides a (somewhat) practical way of finding an upper bound to Problem (4.6). Now if J^θ was given in terms of a neural network, the previous optimization problem would be non-linear and non-convex, however we

could find local minima using methods such as stochastic gradient descent. On the other hand, were we to opt for the linear basis function architecture, the same problem would be both practically and theoretically tractable, and this observation is stated in the following Lemma:

Lemma 13. *Suppose that J^θ is a linear combination of p basis functions. That is, there exist $\{\phi_1, \ldots, \phi_p\} \subset \mathcal{P}$ such Equation (4.7) holds, then Problem (4.8) is convex and therefore practically tractable.*

We omit the simple proof of this lemma, which appears in the original pathwise optimization paper. The bottom line is that as long as the function J^θ is affine in the parameters θ, the pathwise optimization problem is convex.

Before making a final remark about the convex variant of the pathwise optimization problem just discussed, we briefly outline how we might actually solve (4.8). In practice, we would approach this problem by formulating either its SAA (sample-average approximation) version, or using stochastic gradient descent. More precisely, for the SAA method, we would generate S *outer sample paths* of the underlying Markov process $\{x_t^j,\ 0 \leq t \leq T, j = 1, \ldots, S\}$. Then for every exercise opportunity $k = 1, \ldots, K$ and outer sample path $j = 1, \ldots, S$, we generate I *inner samples* of the process conditioned on its value in the previous exercise time, that is $\{x_{t_k}^{j,i}, i = 1, \ldots, I\}$, where each random variable in that sequence is an i.i.d draw from the conditional distribution $\mathbb{P}\left(x_{t_k} \in . \mid x_{t_{k-1}} = x_{t_{k-1}}^j\right)$. With these outer and inner samples, we approximate the upper bound operator $(F_0^D J^\theta)(x)$ with its sample-average approximation

$$(\hat{F}_0^D J^\theta)(x) \triangleq \frac{1}{S} \sum_{j=1}^{S} \left(\max_{k=1,\ldots,K} \left\{ e^{-t_k r} g(x_{t_k}^j) - (\hat{M}^D J)_{t_k}^j \right\} \right). \tag{4.9}$$

where we let $(\hat{M}^D J)^j$ denote the jth sample from a martingale, defined as:

$$(\hat{M}^D J)_{t_k}^j \triangleq \sum_{l=1}^{k} \left(J_{t_l}^\theta(x_{t_l}^j) - \frac{1}{I} \sum_{i=1}^{I} J_{t_l}^\theta(x_{t_l}^{j,i}) \right). \tag{4.10}$$

With that, we can now solve the deterministic problem

$$\inf_{\theta \in \mathbb{R}^p} (\hat{F}_0^D J^\theta)(x) \tag{4.11}$$

whose optimal objective value gives a biased estimate to the value of Equation (4.8). In order to obtain an unbiased upper bound to our original stopping problem, we take the optimal solution to the aforementioned problem, say $\theta^\star$, and evaluate the same function $(\hat{F}_0^D J^{\theta^\star})(x)$ with a new batch of independent outer & inner samples.

One final fact worth noting is that if J^θ is affine in its parameters, Problem 4.11 can be reformulated as a deterministic LP. For convenience, we will refer to this specific algorithm as the *pathwise optimization* (PO) method. We can then solve large-scale instances of the problem with off-the-shelf LP solvers. The pathwise optimization method is a key benchmark that we will consider in this chapter when we develop new upper bounding methods.

If however, we are to parameterize J^θ as a neural network, we will refer to the algorithm (again for convenience) as the *deep pathwise optimization* (DPO) method. However, instead of solving the problem via the complete SAA version in Equation (4.11), we will instead apply stochastic gradient descent to (4.8) with mini-batches of outer and inner samples. Rather than simply settling for DPO as an "application of deep learning" to this problem, we will see if alternative ways of characterizing a martingale are more computationally efficient in practice, in order to offset the extra computational burden of using neural networks.

Martingale Duality under Brownian motion

In this section we will consider alternative martingale representations to the one in Equation (4.5). Recall that in this expression, J^θ can represent a neural network, which typically contains more parameters than the linear basis function architecture. As such, the need for inner sampling can make evaluating the gradient of Equation (4.9) computationally onerous. For this reason, we will explore alternative martingale representations that don't require nested Monte Carlo and thus make deep

learning techniques possible.

For the remainder of this section, we will need to assume that x_t is a geometric Brownian motion process, and therefore satisfies the SDE

$$\begin{aligned} dx_{i,t} &= rx_{i,t}dt + \sigma x_{i,t}dB_{i,t}, \quad i = 1, \ldots, n \\ x_0 &= x \end{aligned} \tag{4.12}$$

where $B_t = (B_{1,t}, \ldots, B_{n,t})$ is a vector of n independent Brownian motions and a_i and b_i are certain functions of the current time and state. As before, we let $\mathbb{F}$ denote the natural filtration generated by B_t. Since the main application of optimal stopping is option pricing, and asset price dynamics are typically assumed to follow a geometric Brownian motion, this is assumption is reasonable.

The advantage in this setting is that we can rewrite Equation (4.5) in terms of an Itô integral. The reason why such representation turns out to be useful are twofold. First of all, any Itô integral is automatically guaranteed to be martingale and hence can be used to evaluate objective the function in Problem 4.3. Second of all, an Itô integral can be approximated without the use of nested Monte Carlo unlike the martingale from pathwise optimization method. While this seems attractive at first, we point out that approximating an Itô integral would require us to use a finer mesh than the discretization of $[0, T]$ given by $\mathcal{T}$. Nonetheless, the absence of inner sampling might make it cheaper to evaluate an approximation to the martingale upper bound and we'll investigate if this is true. In any case, for us to express (4.5) as an integral, we need the following Lemma:

Lemma 14. *Let x_t be a geometric brownian motion with n components satisfying (4.12). Then if $J : \mathbb{R}^n \mapsto \mathbb{R}$ is a twice-continuously differentiable function, we have that for any $t > 0$*

$$\begin{aligned} J(x_t) - \mathbb{E}\left[J(x_t) \mid x_0\right] &= \sum_{i=1}^{d} \int_0^t \sigma x_{i,s} \psi_i^J(t-s, x_s)\, dB_{i,s} \\ &= \int_0^t \sigma(x_s \cdot \Psi^J(t-s, x_s))^\top dB_s \end{aligned} \tag{4.13}$$

where we define for any $u > 0$ the function

$$\psi_i^J(u,x) := \frac{\partial}{\partial x_i}\mathbb{E}\left[J(x_u) \mid x_0 = x\right], \quad i = 1,\ldots,n$$

and function $\Psi^J(u,x)$ denotes the vector of the above n partial derivatives.

The proof of this Lemma follows from the Clark-Ocone theorem and fact that the stochastic process on the left-hand side of the equation (4.13) is a martingale.

We are now motivated to define a new martingale representation, similar to that of (4.5), but one which does not include conditional expectations in its definition. Let $\mathcal{Q}$ be the set of measurable functions $\Psi : [0,T] \times \mathbb{R}^n \mapsto \mathbb{R}^n$. For any function $\Psi \in \mathcal{Q}$, in this family, we define a martingale in terms of it

$$(M^C\Psi)_{t_k} = \sum_{p=1}^{k} e^{-rt_p} \int_{t_{p-1}}^{t_p} \sigma(x_s \cdot \Psi(t_p - s, x_s))^\top dB_s, \tag{4.14}$$

which we dub the *continuous representation*. In particular, it follows directly from Lemma 14 that for any value function $J \in \mathcal{P}$, that is moreover twice-continuously differentiable, that $M^C\Psi^J = M^D J$, where the left hand side is defined in the Lemma. Put differently, the two martingale representations are equivalent when J is a sufficiently smooth function.

Given the new representation, we are ready to define the *continuous* martingale duality operator F_0^C, on functions in $\mathcal{Q}$, as

$$(F_0^C\Psi)(x) \triangleq \mathbb{E}\left[\max_{k\in[K]}\left\{e^{-rt_k}g(x_{t_k}) - (M^C\Psi)_{t_k}\right\} \;\middle|\; x_0 = x\right], \tag{4.15}$$

and as we would expect it shares some of the weak and strong duality properties of the discrete version. We state these facts in the following Theorem.

Theorem 4.2.2 (Weak and strong duality)**.** *For any starting state x, we have that*

1. $J_0^\star(x) \le (F_0^C\Psi)(x)$ *for any function* $\Psi \in \mathcal{Q}$
2. $J_0^\star(x) = \inf_{\Psi\in\mathcal{Q}} (F_0^C\Psi)(x)$.

Proof. The first property (weak duality) follows from the fact that $M^C\Psi$ is a martingale, and hence the steps same as in proof of Theorem 4.2.1 apply here. □

Since Equation 4.17 defines a tight upper bound, we will compactly parameterize a subset of the family $\mathcal{Q}$ just like in the PO method. To this end, we let Ψ^θ denote a function in $\mathcal{Q}$ parameterized by the vector θ. For example, this could describe a feedforward neural network with n inputs, n outputs and where parameter $\theta \in \mathbb{R}^p$ denotes all the p weights in the network (from all the hidden layers). Now that we defined a family of all such parameterized functions, which we may call $\hat{\mathcal{Q}} \subset \mathcal{Q}$, we optimize the upper bound given in Equation (4.15) over it by solving

$$\inf_{\theta \in \mathbb{R}^p} (F_0^C \Psi^\theta)(x) \tag{4.16}$$

which is a relatively low-dimensional optimization problem in p parameters. If Ψ^θ were a neural network, we could interpret the above objective function as a rather complicated loss function given in terms of the outputs of the network Ψ^θ.

We are faced with one remaining difficulty, which is evaluating the expectation inside objective function. To deal with this, as expected, we use a sampling approach and introduce the following new notation. Let us define the mesh $\mathcal{U} = \{u_0, u_1, \ldots, u_L\} \subset [0, T]$, where $u_0 = 0$ and $u_L = T$ and we assume that $\mathcal{T} \subset \mathcal{U}$, i.e. this mesh is finer than the grid of exercise opportunities.

Over this new mesh, we sample N trajectories $\{x^j_{u_l},\ t = 0, \ldots, L,\ j = 1, \ldots, N\}$ of our Markov process. We also denote with $\{B^j_{u_l},\ t = 0, \ldots, L,\ j = 1, \ldots, N\}$ the corresponding set of paths from the brownian motion. Thus we can compute an unbiased estimate of (4.15) with the following expression

$$(\hat{F}_0^C \Psi)(x) \triangleq \frac{1}{N} \sum_{j=1}^{N} \left(\max_{k \in [K]} \left\{ e^{-rt_k} g(x^j_{t_k}) - (\hat{M}^C \Psi)^j_{t_k} \right\} \right) \tag{4.17}$$

where $\hat{M}^C\Psi$ is an approximation of the continuous martingale and is defined according

to

$$(\hat{M}^C\Psi^\theta)^j_{t_k} \triangleq \sum_{p=1}^{k} e^{-rt_p} \sum_{t_{p-1}\le u_l<t_p} \sigma(x^j_{u_l}\cdot\Psi^\theta(t_p-u_l,x^j_{u_l}))^\top(B^j_{u_{l+1}}-B^j_{u_l}). \tag{4.18}$$

In order to tune the network weights, we would then solve the following deterministic problem with N sample trajectories:

$$\inf_{\theta\in\mathbb{R}^p} (\hat{F}^C_0\Psi^\theta)(x) \tag{4.19}$$

where we defined $\hat{F}^C_0\Psi^\theta$ with respect to the N sample paths. Evidently, the problem is analogous to the one in pathwise optimization. In fact, as is explained in the following Proposition, with the right architecture, Problem (4.19) is convex.

Proposition 5. *Suppose that there exist a set of basis function $\phi_1,\dots,\phi_m$, each one mapping $\mathbb{R}^n$ to $\mathbb{R}$, such that*

$$\Psi^\theta(x)=\Theta\Phi(x)$$

where, with a slight abuse of notation, Θ denotes the components of θ arranged into a $n\times m$ matrix and $\Phi(x)=\Big[\phi_1,\dots,\phi_m(x)\Big]$. In that case, Problem 4.19 is convex and can be formulated as a linear program.

Proof. By inspecting Equation (4.17) we see that the objective function in the problem is a sum of maxima over different affine functions in θ. In other words, the objective is convex piecewise linear in θ and such a problem can be solved with linear programming. □

The above result simly tells us that as long as Ψ^θ is affine in its parameter, the dual problem for finding an upper bound is convex. Moreover, this suggests to us another efficient way of computing an upper bound through linear programming, that is different from the PO method. We will dub this particular algorithm as the Continuous Pathwise Optimization (CPO) method and we will also benchmark it later on.

At this point it is natural to wonder if we can readily just use a neural network as Ψ^θ for Problem (4.19). In theory, that's possible and we would then minimize such a function with backpropagation to find a local minimum. Since this mimics the typical way neural networks are optimized, we describe the step of solving this optimization problem the *training* phase. A general issue is that, if N is relatively small, we might 'overfit' θ to the sample trajectories during training. This means that when we finally compute an unbiased estimate of upper bound (4.15), with the trained weights, the actual upper estimate on the option price would be too loose. On the other hand, using a too large value of N would make computing gradients in Problem (4.19) computationally onerous. For these reasons, we will instead consider an alternative, discretized stochastic problem

$$\inf_{\theta\in\mathbb{R}^p} \mathbb{E}\left[\max_{k\in[K]}\left\{e^{-rt_k}g(x_{t_k}) - (\tilde{M}^C\Psi)_{t_k}\right\} \;\middle|\; x_0 = x\right] \tag{4.20}$$

where the martingale part is approximated using the following sequence of sums of random variables sampled at points in the mesh $\mathcal{U}$:

$$(\tilde{M}^C\Psi^\theta)^j_{t_k} \triangleq \sum_{p=1}^{k} e^{-rt_p} \sum_{t_{p-1}\le u_l < t_p} \sigma(x_{u_l}\cdot\Psi^\theta(t_p-u_l,x_{u_l}))^\top(B_{u_{l+1}}-B_{u_l}). \tag{4.21}$$

We would then train our network by minimizing (4.20) via stochastic gradient descent, with independent mini-batches of sample trajectories. We give this algorithm the nickname, Deep Continuous Pathwise Optimization (DCPO). A nice feature of our problem is that because we can simulate trajectories, the data we work with is effectively unlimited and every mini-batch is contains new and (with high probability) unseen samples. It's plausible that this would prevent overfitting. It is also worth noting that the discretized approximation in (4.21) is by construction always a martingale, no matter how coarse $\mathcal{U}$ is. Thus we know that (4.20) is a always a valid upper bound, even when $\mathcal{U} = \mathcal{T}$. In practice, with stochastic gradient descent and a fine enough mesh $\mathcal{U}$, we find good local minima and avoid overfitting. For this reason, we do not need to resort to regularization techniques such as dropout [Srivastava et al.,

2014] when optimizing these networks.

Numerical Experiments

In this section we will compute upper bound estimates on option prices using the three methods described in the previous section, namely PO, CPO and DCPO. Recall that the first two algorithms rely on us providing basis functions to describe state features, while the third doesn't. Whenever they're needed, we will use a common set of basis functions, which are the same as what we saw earlier in this chapter. In other words, we will specifically fix $n+2$ basis functions, $\phi_1(x), \ldots, \phi_{n+2}(x)$ evaluated at a state $x \in \mathbb{R}^n$, where $\phi_{n+1}(x) = g(x)$ (the problem-specific option payoff), $\phi_{n+2} = 1$ (a constant) and $\phi_i(x) = x_i,\ 1 \leq i \leq n$ (the individual asset prices).

We will consider a max-call option on n assets whose common initial price is denoted by p_0. The assets will follow the same price dynamics, x_t, as in the previous experiment of this chapter and so will be modeled with a multi-dimensional brownian motion. The payoff of the option in a state $x \in \mathbb{R}^n$ is

$$g(x) = \left(\max_{1 \leq i \leq n} x_i - S \right)^+$$

where $S = 100$ is the strike price. We will use two different problem horizons, $T = 1, 3$ years and suppose that there are $d = 54$ evenly spaced exercised opportunities, so that $\mathcal{T} = \{0, T/d, 2T/d, \ldots, T\}$. The remainder of the parameters will be reused from the previous experiment, so that the risk free rate is $r = 0.05$, the annualized volatility is 20%, and so on. For the continuous representation, we use a finer mesh with $d' > d$ evenly spaced points, namely $\mathcal{U} = \{0, T/d', 2T/d', \ldots, T\}$ (note that d has to divide into d'). We will leave d' as a parameter that we'll tune later.

In order to help us describe the computational setup for all algorithms and report results, we define a new *complexity parameter* (CP) as roughly the total number of calls to the value function (whether it's J^θ or Ψ^θ) in Equations (4.10) or (4.18), within each term of the summation. More precisely, for the DCPO and CPO methods, CP would be d'/d, i.e. the relative increase in mesh granularity, while for the PO and DPO

methods the CP parameter is roughly I, the number of inner samples. The reason we define this parameter, is in order to establish a common metric for the computational complexity of all the algorithms. Generally, with larger values of this parameter, the bounds from all the algorithms improve, as we will demonstrate later. Employing a larger complexity parameter however requires longer runtimes and higher memory requirements due to the increased number of terms in (4.10) and (4.18).

The parameter settings and implementation details for the three algorithms, in all the experiments that follow, are summarized below:

- **PO:** In the notation of Section 4.2.3, we solve the LP in (4.11) with $N = 10,000$ outer samples and using the basis functions mentioned earlier. For each outer sample, the number of next state inner samples for each period is equal to CP, the complexity parameter. Given a solution θ^{PO} we evaluate $\hat{F}_0^D J^{\theta^{\mathrm{PO}}}$ using a distinct set of $N = 10,000$ outer samples and $I = CP$ inner samples for one-step conditional expectations, where CP is the complexity parameter.

- **CPO:** We solve a LP derived from equation (4.19) with $N = 10,000$ outer samples. These are the same trajectories as the ones we use for the PO method, only sampled at a finer granularity given by the mesh $\mathcal{U}$, where $d' = CP \times d$ and CP is the complexity parameter. We let Ψ^θ, in the definition of (4.19), denote the linear basis function architecture, so that the parameter θ is a $n+2$-dimensional vector. Given an optimal solution θ^{CPO} to this LP, we evaluate $\hat{F}_0^C \Psi^{\theta^{CPO}}$ on a distinct batch of $N = 10,000$ sample trajectories. These are the same paths used to evaluate the PO method.

- **DCPO:** Following the notation of Section 4.2.3, we let $\Psi^\theta = \tilde{\Psi}^\theta$ where $\tilde{\Psi}^\theta : \mathbb{R}^n \mapsto \mathbb{R}^n$ denotes a feedforward neural network with some number of hidden layers H and width W (the number of hidden units in every layer). We apply batch-normalization [Ioffe and Szegedy, 2015] and ReLU activations to the output of every hidden layer. Here θ represents all the weights, biases and batch-normalization scaling parameters in this network. All neuron weights are initialized at random from a zero-mean Gaussian distribution with variance

$1/(HW)$.

We train this network by minimizing the corresponding loss function given in (4.20) using the Adam algorithm [Kingma and Ba, 2014] with different learning rate parameters that we will vary between 0.001 and 0.1. Every update step to θ is performed with a fresh mini-batch of 1,000 independent sample trajectories. We use the early stopping framework from [Goodfellow et al., 2016, p. 240] with 100 steps between evaluations and the 'patience parameter' set to 10, and where the validation data consists of a separate batch of 10,000 independent sample trajectories.

Letting θ^{DCPO} denote the best parameters after training the network, we evaluate $\hat{F}_0^C \tilde{\Psi}^{\theta^{DCPO}}$ on the same batch of $N = 10,000$ sample trajectories used for the PO and CPO methods. We keep this evaluation data the same to reduce variance when comparing algorithms.

- **DPO:** For this algorithm, we let $J^\theta = \tilde{J}^\theta$ where $\tilde{J}^\theta : \mathbb{R}^n \mapsto \mathbb{R}$ is a feedforward neural network with some number of hidden layers H and W hidden units in every layer (also known as width). This is the same type of neural network as for the DCPO method, where batch-normalization and ReLU activations are applied to the output of every hidden layer. We train this network, in a similar way to DCPO, with the Adam algorithm. The main difference is that we minimize the loss function from (4.8).

  Like in the PO algorithm, we take for each outer sample, and at every exercise time, $I = CP$ inner samples. Here CP denotes the complexity parameter. We evaluate this algorithm in an analogous manner to the PO method.

The goal of the following experiment is to find the tightest bounds possible with each algorithm, given the available computational resources. For this reason, we will use a relatively large number of outer/training samples (mentioned above) and fix CP to a large value, specifically 100, that allowed us to compute all bounds within 24 hours.

We tested both methods involving neural networks, namely DPO and DCPO, using the Adam algorithm and tried four learning rate parameters: $0.001, 0.005, 0.01$

n	4			8		
Algo	CPO	DCPO	PO	CPO	DCPO	PO
Mean	55.626	52.841	52.998	76.063	72.081	72.206
S.E	0.017	0.064	0.041	0.038	0.055	0.069
Time (min)	0.000	1032.732	144.633	149.337	2150.381	151.043
Best LR	0.010	0.010	N/A	0.010	0.005	N/A
Best Depth	0	4	N/A	0	4	N/A

Table 4.3: Best bounds from all methods with different problem sizes. The last two rows show the corresponding parameter values for the best solutions.

and 0.1. All code was implemented using the Python interface for TensorFlow 1.3.1 [Abadi et al., 2015] and performed on an Intel Xeon E5-2690 2.60GHz CPU with 32 GB of RAM. Since training a neural network is a non-convex problem, we repeated the training procedure 30 times with random initial weights and with the four different settings of Adam algorithm's learning rate parameter. We picked the best solution from all $30 \times 4 = 120$ initializations.

The resulting bounds generated by all four methods are shown in Table 4.3, where we vary the number of assets n in the problem and fix $p_0 = 100$. In the table, we report average upper bounds on option price over 10 trials, along with the corresponding standard errors and computation times in minutes.

Broadly speaking, we make the following observations and conclusions. The bounds generated from the DCPO methods are always at least as strong as the PO ones. At best, we find a 30 basis point improvement. Bounds from DPO and CPO are typically the weakest. From this, we can see the DCPO method offers us two benefits: (i) potentially better quality bounds over alternatives (ii) the ability to estimate the option price quite well, *without* the need for basis functions to be provided as inputs. We emphasize this point and recall that the only input to the network in DCPO were merely the asset prices. Nonetheless, the approximation architecture we used for it (such as a 4-layer neural network) was able to model a rich enough class of functions. As long as the optimization is properly run, we are able ensure good quality bounds.

Unfortunately, the benefits just mentioned came at the expense of much greater computational cost. As shown in Table 4.3, it typically took at least 16 hours for

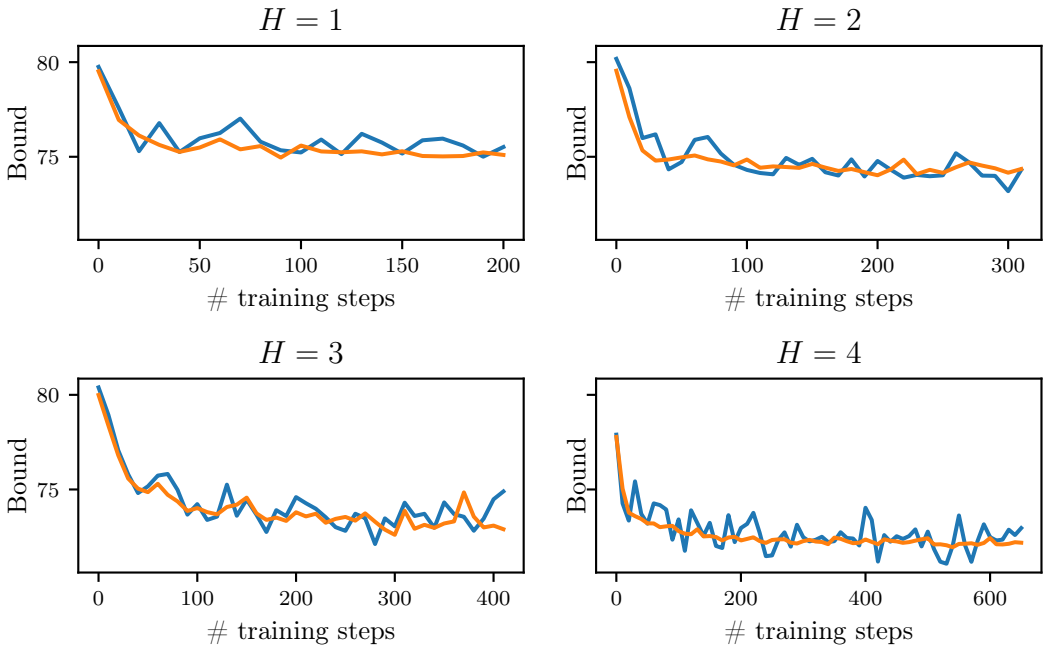

Figure 4-1: Objective value as a function of the number of training steps. This corresponds to the same problem as in Table 4.3 with $n = 8$.

the stopping criterion, associated with training the network, to be met. Figure 4-1 shows how the objective function on both training and validation data changed with the number of gradient update steps for networks of varying depth. We see the algorithm begins to converge about a quarter of the way through until it hits the early stopping criterion. Therefore, for purely practical purposes, we could settle for slightly looser bounds if computation times were limited to roughly 4 hours and we were forced to terminate the optimization earlier. Another key observation is that the number of training steps until termination increases proportionally with network depth. The figure also suggests that, because validation and training set objective values are close, we are indeed mitigating the effects of overfitting thanks to using mini-batches of sample trajectories and the Adam algorithm.

We end this section by exploring the effects of setting different parameters on bounds generated by the DCPO method, as well as runtimes. Just as in the main experiment of this section, we train the network 30 times using the DCPO method from different random starting weights and pick the best solution. To start with,

n	Depth	Mean	S.E	Time (min)
4	1.0	53.885	0.037	365.576
	2.0	53.225	0.055	423.834
	3.0	53.010	0.056	906.412
	4.0	52.965	0.047	1,375.504
8	1.0	74.930	0.034	566.103
	2.0	73.451	0.043	832.401
	3.0	72.860	0.055	1,103.703
	4.0	72.626	0.059	1,411.745

Table 4.4: Best bound with DCPO for networks of varying depths.

Depth	LR	Mean	Time (min)	S.E
3.0	0.001	72.40	2720.68	0.06
	0.005	72.23	1787.76	0.05
	0.010	72.30	1106.52	0.05
	0.100	72.86	1103.70	0.06
4.0	0.001	72.24	2268.59	0.06
	0.005	72.08	2150.38	0.06
	0.010	72.21	1265.81	0.04
	0.100	72.63	1411.74	0.06

Table 4.5: Effect of depth & learning rate on DCPO bounds.

we explore the effects of network depth by keeping all other parameters fixed. In particular, we set the width to 30 neurons and keep the learning rate for Adam to 0.1; all other parameters are set the same way as before. For the results that follow, we report the best mean upper bound estimate, its standard error, the time taken to train the network In table 4.4, we see that in almost all cases, deeper networks result in tighter bounds but this improvement diminishes with more layers. This is promising as this implies there is a real gain to be had from deeper architectures, as we might expect. We also remark that the batch-normalization was crucial for allowing us to optimize deeper networks. Without that, we found that the objective value, unlike what was shown in Figure 4-1, would zig-zag over time and generally fail to converge to good local minima.

Next, we explore the effect of varying the learning rate with deep networks. In

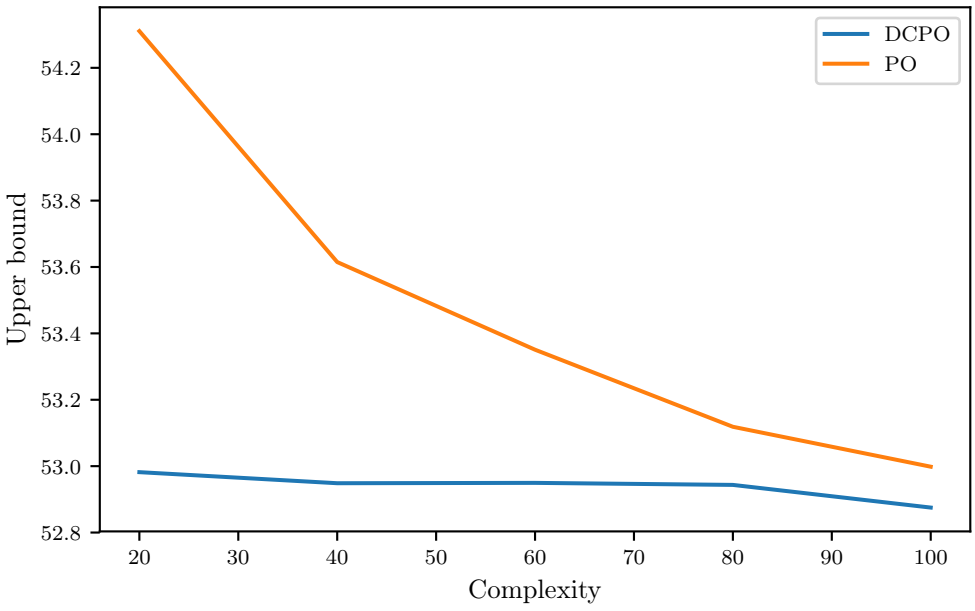

Figure 4-2: Complexity

this experiment, we fix the learning rate and depth, then take the minimum solution over 30 random instances. Table 4.5 shows that with lower learning rates, convergence generally takes much longer but the bounds improve slightly. As expected, this shows that with a smaller learning rate parameter, the Adam algorithm progresses more slowly but settles into local minima better. Again, this demonstrates an important trade-off between computational efficiency and solution quality.

Finally, we provide some experimental evidence that suggests the DCPO method gives tighter bounds than PO, with smaller complexity parameter values. This can be seen from Figure 4-2, where we see that the DCPO bound deteriorates much more gracefully even with a complexity parameter (i.e. number of additional calls to a neural network in between exercise opportunities) of around 20. With such a modest runtime overhead, the DCPO bound is tighter by around 2 %. This may suggest that the continuous representation is in some sense better for approximating a good martingale.

The main conclusion we reach from these experiments is that deep learning provides a tangible edge to option pricing. Moreover, the neural network architecture

used in the novel DCPO algorithm leverages no knowledge about the underlying problem (such as the payoff function) because it only takes as input the asset prices. It seems likely therefore that the method can readily generalize to different kinds of stopping problems with new payoff structures. Unfortunately, the nature of training neural networks can make computing bounds using this new method computationally intensive in terms of both time and memory. The way to combat this might be to use GPUs for training the network, or possibly multiple cores. It could also be fruitful to train multiple networks in parallel, as we have done here, in order to find good local minima quickly.

In the next section, we will study another key problem in financial engineering, namely, portfolio optimization and see how a broad class of such problems could be tackled with different RL methodology.

4.3 Portfolio Optimization and Quasi-Linear Convex Control Problems

In this section, we will study a broad class of control problems that have several important, practical applications in financial management, but moreover extend to other domains such as engineering and supply chain management. What control problems in these areas have in common is that they deal in discete time, the decisions are modeled with continuous variables and the endogenous state dynamics are, not only known to the decision maker, but are also linear. Even though this seems restrictive, many systems are linear over the range we'd like to operate them, or can be approximated as such via linearization and/or discretization of time. Moreover, in many control problems of significant practical interest, such as in business and economics, state dynamics do follow arguably simple, linear update rules. For instance, if the state is inventory, like the problem we studied in Chapter 3 or in supply chain applications, the update rule is a simple addition or subtration of the control applied (such as quanitity ordered or sold) to the state variable. There are other particular,

salient features of the problem we will study and we list them below:

1. **General Markovian dynamics of exogenous state:** While we mention earlier that the endogenous component of state (i.e. that part affected by the agent's controls) follows linear dynamics, in the framework we develop and analyze, it is allowed for the exogenous state to follow arbitrary stochastic dynamics so long as they are Markovian. This is useful in financial applications, since it is often assumed that asset prices evolve independently of the agents actions and do follow a Markov process, such as a Geometric Brownian motion.

2. **Concave reward functions:** The class of problems we are about to study include as a special case the Linear Quadratic Regulator (LQR) problem, which we will expound upon later. One of the key features of LQR is the fact that the reward function is concave quadratic (assuming that we're maximizing the overall objective function) in the controls and state variables. For the type of problems we are about to test out and analyze, the reward function could be any general concave function of state and control.

3. **Convex constraints on control & state:** Our framework has the flexibility to model convex constraints on both control on state. This is a feature that in cannot be handled in the stadard LQR framework. For example, in finance this allows us to model no-short constraints on trading decisions or to limit the maximum position size in a portfolio.

We give this class of problems the name 'quasi-linear-convex control'. In what follows, we will formulate this framework as a certain type of dynamic programming problem. If it were not for the non-anticipativity constraints imposed on a policy, the problem could be solved through convex optimization. Unfortunately, the fact that this is a stochastic dynamic problem, makes it suffer from the curse of dimensionality shared with MDPs.

On the surface, what could make this problem harder to tackle than many MDPs is the fact that the control is continuous. This means that were we to resort to

general ADP methods, like approximate value iteration or Q-learning, it would be unclear how we might even determine actions for a greedy policy due to the non-linear, and non-convex form of the value function approximation. In other words, we would be forced to solve intractable (possibly constrained) non-linear optimization problem during the course of running a policy. For this reason, we attempted to implement the continuous Q-learning technique from Gu et al. [2016] and the policy gradient method [Williams, 1992, Sutton et al., 2000, Kakade, 2002]. Ultimately, we found surprisingly that a certain policy gradient method, adapted to constrained QLCC problems, yielded state-of-the-art performance in portfolio optimization. This finding, along with some preliminary theoretical analysis, will be the focus of this section.

4.3.1 Problem formulation

We start by formulating a general discrete-time stochastic control problem with quasi-linear dynamics, which captures a range of applications in finance, business and engineering.

Consider a $T+1$ period dynamic problem whose state at each period $t \in \{0, 1, \ldots, T\}$ is given by the vector $y_t \in \mathbb{R}^{n+k}$. We express the state as a pair $y_t = (x_t, f_t)$, where $x_t \in \mathbb{R}^n$ and $f_t \in \mathbb{R}^k$. As well see shortly, in the context of portfolio optimization, we can think of x_t is being a state variable representing the current portfolio and f_t being the current set of factors for predicting asset price movements. Suppose further that the agent is given fixed, known values $x \in \mathbb{R}^n$ and $f \in \mathbb{R}^k$ for the initial state, such that $x_0 \equiv x$ and $f_0 \equiv f$ with probability 1. For consistency, we denote the pair of initial states as the vector $y = (x, f)$. Let $\{\eta_t,\ t = 1, \ldots, T\}$ and $\{\epsilon_t,\ t = 1, \ldots, T\}$ be two zero-mean, exogenous i.i.d noise processes (possibly correlated with one another) that generate the uncertainty in our problem. Assume that ϵ_t and η_t are of dimensions n and p, respectively. We denote with $\mathbb{F} = \{\mathcal{F}_t,\ t = 0, \ldots, T\}$ to be the natural filtration generated by these processes. We will assume that the dynamics for f_t satisfies the equation

$$f_t = h_t(f_{t-1}, \eta_t)$$

for all $1 \leq t \leq T$, where $h_t : \mathbb{R}^k \times \mathbb{R}^p \mapsto$ is an arbitrary measurable function. This means that the process $\{f_t\}$ evolves in a Markovian fashion and is only affected by i.i.d process $\{\eta_t\}$. Thus, we will say that it's *exogenous*, i.e. not affected by the agent's actions. At the same time, the dynamics of the remaining components of state are defined according to the linear equations

$$x_t = A_t x_{t-1} + B_t \mu_t + C_t f_{t-1} + \epsilon_t \tag{4.22}$$

for all $1 \leq t \leq T$. Here the vector $\mu_t \in \mathbb{R}^\ell$ is the control applied in period t, and we assume that this is an $\mathcal{F}_{t-1}$-measurable function, i.e. the control is non-anticipative and depends on all the randomness observed up to and including period $t-1$. Thus the dynamics of x_t are linear and are determined by matrices $A_t \in \mathbb{R}^{m \times m}$, $B_t \in \mathbb{R}^{m \times \ell}$ and $C_t \in \mathbb{R}^{m \times k}$, as well as the current exogenous state, f_{t-1}. Since the above component of state does depend on the agent's actions, we will refer to this as the *endogenous* state.

Now we will define a per-period reward function $r_t : \mathbb{R}^{n+k} \times \mathbb{R}^\ell \mapsto \mathbb{R}$ for every $1 \leq t \leq T$. This is a function that takes as input the entire state from $t-1$, as well as the control applied at period t, and outputs a reward to the agent. We define it as

$$r_t(y_{t-1}, \mu_t) = r_t(x_{t-1}, f_{t-1}, \mu_t) = f_{t-1}^\top H_t \mu_t + g_t(x_{t-1}, \mu_t),$$

where g_t is some strictly, jointly concave function in y_t and μ_t and $H_t \in \mathbb{R}^{k \times \ell}$. In addition to defining rewards, we impose polyhedral constraints on the allowed state at every time period:

$$F_t \mu_t + G_t x_{t-1} \leq b_t, \ t = 1, \ldots, T \tag{4.23}$$

where F_t, G_t, b_t are real matrices/vectors of appropriate dimensions. In other words, the control at any period has to satisfy some set of linear constraints that depend on the current value of the endogenous state. We let $\mathcal{U}$ denote the set of all non-anticipative policies that satisfy all the above linear inequalities with probability 1.

Our goal is to find a policy in $\mathcal{U}$ which maximizes the total sum of rewards across all time periods from 1 to T. Therefore, in summary, the problem we face is the following stochastic dynamic one:

$$\begin{aligned}
\underset{\mu\in\mathcal{U}}{\text{maximize}} \quad & \mathbb{E}\left[\sum_{t=1}^{T} r_t(y_{t-1},\mu_t)\right] \\
\text{subject to} \quad & x_t = A_t x_{t-1} + B_t\mu_t + C f_{t-1} + \epsilon_t, \quad t=1,\dots,T \\
& f_t = h_t(f_{t-1},\eta_t). \qquad\qquad\qquad\quad t=1,\dots,T \\
& F_t\mu_t + G_t x_{t-1} \le b_t \qquad\qquad\quad t=1,\dots,T \\
& y_0 = y \qquad\qquad\qquad\qquad\qquad\quad .
\end{aligned} \tag{4.24}$$

We denote the optimal value of this problem as $V_0^\star(y) = V_0^\star(x,f)$, which is a function of the initial state $y = (x,f)$. We generalize the definition of the optimal value function to arbitrary time periods $1 \le t < T$, and write it as $V_t^\star(y_t) = V_t^\star(x_t,f_t)$, which is a measurable function of x_t and f_t. In other words, this is the value-to-go from time $t+1$ and onwards, given that we are currently in a state $y_t = (x_t,f_t)$. Finally, we will assume that under any random realization, there is a feasible control μ_t for every time period.

Notice that the above framework includes, as a special case, the LQR problem. In particular, it can be shown that Problem (4.24) is LQR if h_t is an affine function, there are no constraints on state, and r_t is concave quadratic in x_{t-1} and μ_t.

4.3.2 Policy Gradient with Penalization

Unfortunately, solving the problem given in (4.24) is intractable and so we resort to RL methods. A popular approach in RL is to treat the policy we're searching for as a parameterized function of the relevant state information. We would then tune the parameters of this function to maximize some metric, usally just the average total reward earned by the corresponding policy, in order to find an effective (yet likely sub-optimal) policy. This is similar in a lot of ways to Q-learning, TD-Learning or approximate value iteration, in the sense that we use Monte-Carlo simulation to

generate sample trajectories from our system in order to fit some function of interest. The difference this time is we would be 'fitting' a heuristic policy directly, as opposed to an approximation to the optimal value function and defining a heuristic greedy policy in terms of it.

Policy gradient (PG) is a mature RL technique that has been expounded on and developed in Williams [1992], Kakade [2002], Sutton et al. [2000]. Traditionally, PG deals with MDPs that have finite action spaces, and typically represents a policy as a probability distribution over actions in a given state. This means that the policy, in practice, is randomized. Moreover, PG has typically been used as a method for model-free reinforcement learning because all that is needed for it to work is simulation of sample trajectories, given the current parameterization of a policy, and observations of the rewards earned. In particular, there is no need for knowing model dynamics in implementing PG. To keep this section shorter, we will not delve into details of how vanilla PG works exactly but it suffices to say, that periodically after running the system for some time, the policy's parameters are updated given obsevations of past actions, visited states and rewards.

Our application of PG differs from standard implementations in two key ways. First of all, we model a policy as a real, vector-valued function of state. In other words, we do not impose the restriction that a policy should be represented by a probability distribution. Secondly, the policy is modeled as a deterministic time-inhomogenous function of state.

Thus let us parameterize a policy in terms of a feedfoward neural network. In each period t, we define the *proposed control* as the deterministic function $u_t^\theta : \mathbb{R}^{n+k} \mapsto \mathbb{R}^m$. This could be modeled as neural network with weights $\theta \in \mathbb{R}^P$. Due to the constraints (4.23), we cannot in general use the control u_t^θ in period t as-is because it isn't guaranteed to be feasible. For this reason, we will *project* the proposed control to the feasible set. To this end, let us define the polyhedron

$$C_t = C_t(x_{t-1}) \triangleq \{u \in \mathbb{R}^\ell : F_t u + G_t x_{t-1} \leq b_t\}.$$

Assume we are given a distance metric $D : \mathbb{R}^\ell \times \mathbb{R}^\ell \mapsto \mathbb{R}_+$, which must be convex. Let us denote the distance of a vector $u \in \mathbb{R}^\ell$ to the feasible set, for the given metric D, as the function $D_{C_t}(u)$. Formally, this is given by the optimal cost of the convex minimization problem,

$$\begin{aligned} \underset{z \in \mathbb{R}^\ell}{\text{minimize}} \quad & D(z, u) \\ \text{subject to} \quad & z \in C_t, \end{aligned} \tag{4.25}$$

where we let $\Pi_{C_t}(u)$ denote an optimal solution to the above optimization problem, which we also call the *projection* of u onto C_t. Now that we defined a projection operation and a notion of distance to the feasible set, we may consider applying a control of the form

$$\pi_t^\theta(y_{t-1}) \triangleq \Pi_{C_t}(u_t^\theta(y_{t-1})),$$

which by construction is guaranteed to be feasible. The above should simply be interpreted as the projection of the, generally infeasible, proposed control to the nearest feasible one. Thus, it's clear how to implement a policy in practice. The only remaining question is how to find a proposal function $\mu_t^\theta(.)$ that yields good performance in terms from the resulting policy $\pi^\theta = \{\Pi_{C_t} \circ u_t^\theta : \quad t = 1, \ldots, T\} \in \mathcal{U}$.

Training algorithms We list four ways of searching for proposal functions μ_t^θ. Recall that the above simply denotes a function parameterized by vector $\theta \in \mathbb{R}^P$. All of the algorithms involve maximizing some objective function in terms of θ. We can intuitively guess that the first of these won't perform well (as we will see later) but we describe the algorithm nonetheless for pedagogical reasons. As we will demonstrate later, the remaining two algorithms yield near-optimal policies.

All the algorithms essentially work by fixing an objective function in terms of the initial state y and parameters θ, say $\nu(\theta, y)$, and the maximizing this objective over θ. The function ν is always an expectation over sums of random variables corresponding to rewards and penalties. In other words, we would solve:

$$\underset{\theta \in \mathbb{R}^P}{\text{maximize}} \; \nu(\theta, y)$$

via algorithms such as Adam, and performing parameter updates using mini-batches of sample trajectories. Below we list the four possible choices for ν, which would give us a complete algorithm:

1. **Policy gradient:** To guide us in our search for μ^θ, we can ignore the constraints and calculate the hypothetical total expected reward *without* projection as:

$$\hat{V}(\theta, y) = \mathbb{E}_{\mu^\theta}\left[\sum_{t=1}^{T} r_t(y_{t-1}, \mu_t^\theta(y_{t-1})) \,\middle|\, y_0 = y\right],$$

   where the expectation is over sample paths produced by applying the infeasible proposed controls μ_t^θ, with the dynamics in (4.22).

   If the original problem was merely constrained LQR (a problem with linear dynamics, convex quadratic costs and polyhedral constraints), we could find a globally optimal value for $\hat{V}$, which would correspond to the projected optimal LQR policy (see Moallemi and Sağlam [2015]). As we might expect this would produce highly sub-optimal policies.

2. **Projected policy gradient:** Because our goal is to solve (4.24), we simply aim to calculate the value of a policy π^θ via the equation

$$\hat{V}^\Pi(\theta, y) = \mathbb{E}_{\pi^\theta}\left[\sum_{t=1}^{T} r_t(y_{t-1}, \pi_t^\theta(y_{t-1})) \,\middle|\, y_0 = y\right],$$

   where the expectation is over sample paths produced by running the policy π^θ.

   Recall again that π^θ describes the projected values of μ_t^θ. For reasons that we will discuss later, the trained policy here typically performs poorly.

3. **Penalized policy gradient:** This is an extension of the first method where we allow infeasible controls. The only difference here is we penalize the objective every time that the constraints get violated. Precisely, the penalty at every period t is $\lambda D_{C_t}(\mu_t^\theta(y_{t-1}))$ where $\lambda > 0$ is some scalar. In other words, the penalty is proportional to the distance from the nearest feasible control. Thus,

the objective we would maximize is

$$V(\theta, y) = \mathbb{E}_{\mu^\theta}\left[\sum_{t=1}^{T}\left\{r_t(y_{t-1}, \mu_t^\theta(y_{t-1})) - \lambda D_{C_t}(\mu_t^\theta(y_{t-1}))\right\} \,\middle|\, y_0 = y\right],$$

where the expectation is over trajectories generated by executing the infeasible policy μ^θ given the dynamics in (4.22).

4. **Projected and Penalized policy gradient:** Finally, this method is an extension of the second where we follow sample paths of the projected policy, namely use the controls according to $\pi_t^\theta = \Pi_{C_t}(\mu_t^\theta)$, but penalize the underlying proposed control μ_t^θ in every period. In other words, we use the objective function

$$V^{\Pi}(\theta, y) = \mathbb{E}_{\pi^\theta}\left[\sum_{t=1}^{T}\left\{r_t(y_{t-1}, \pi_t^\theta(y_{t-1})) - \lambda D_{C_t}(\mu_t^\theta(y_{t-1}))\right\} \,\middle|\, y_0 = y\right],$$

where the expectation, in analogy with the second method, is with respect to trajectories seen by following the projected policy.

The advantage of the last two methods is that the additional penalty term provides "gradient information" that encourages the algorithm to search for policies that satisfy the constraints in the problem in addition to earning large rewards. Without it, the training procedure might find and settle for policies that seem effective but, when implemented in practice via projection, yield lower rewards. We make the intuition clearer with the following concrete example of where projected gradient descent fails to find a globally optimal policy given a particular parameterization for μ^θ.

Example 1. *Suppose we have a one period problem (i.e. $T = 1$) with initial state $y = (x, f)$ and that we parameterize a policy as a constant vector $\mu^\theta(y) = \theta$ with $\theta \in \mathbb{R}^\ell$. Assume that the constraint set, at time period 1, is the following polyhedron*

$$C_1 = \{u \in \mathbb{R}^\ell : u_i \leq 1,\ \forall i \in [\ell]\}.$$

Then we have that

$$\begin{aligned}\hat{V}^{\Pi}(\theta, y) &= \mathbb{E}\left[r_1(y, \Pi_{C_1}(\mu^{\theta}(y))\right] \\ &= \mathbb{E}\left[r_1(y, \Pi_{C_1}(\theta))\right] \\ &= r_1(y, \theta \wedge e)\end{aligned}$$

where r_1 is some strictly concave function in its second argument, $e = (1, \ldots, 1)$ is an ℓ-dimensional vector of ones and $\wedge$ denotes the componentwise minimum. Assume that there exists $\theta^{\star} \in \mathbb{R}^{\ell}$ such that $\theta^{*} \leq e$ and is the global minimum of $r_1(y, \theta)$ over θ. It is not hard to see that θ^{*} is the global minimum for $\hat{V}^{\Pi}(\theta, y)$ as well.

Suppose that we start gradient descent with an initial value $\theta^0 > e$, then the gradient of the objective function evaluated at this point is

$$\nabla V^{\Pi}(\theta_0, y) = \nabla(\theta_0 \wedge e)\nabla V^{\Pi}(y, \theta_0 \wedge e) = 0,$$

and so gradient descent doesn't make any progress, and we never recover the optimal solution $\theta^{\star}$.

As the above example shows, projected policy gradient can easily get stuck at poor stationary points. Applying penalization to the above example would prevent this problem and our intuition at the moment is that this is the reason why algorithms 3 and 4, mentioned previously, work so much better in practice. We will demonstrate this phenomenon shortly with the following two case studies.

4.3.3 Numerical Case Studies

To gauge the efficacy of policy gradient algorithms just discussed, we will test out two practical trading problems. The first is a relatively small-scale problem that involves liquidating a long position on a single stock. Meanwhile, the second case-study involves a much larger problem, which is about managing a portfolio of 15 commodity futures contracts over a period of several months. The two problems, while distinct from each other, have a common underlying model based off our quasi-

linear, convex one, that we will outline here.

Portfolio optimization formulation

We consider an economy with n different assets. The agent is given an initial portfolio $x_0 \in \mathbb{R}^n_+$ in these n assets and needs to trade into and out of it over T periods. Here, the control $\mu_t \in \mathbb{R}^n$ represents a trade, where $\mu_{i,t} > 0$ if asset i is being bought in period t, $\mu_{i,t} < 0$, if it's being sold and $\mu_i = 0$ represents the absence of a trade in that asset i during t. Our positions in the n assets are given by vector x_t, which evolves according to the linear equations

$$x_t = x_{t-1} + \mu_t.$$

Let $p_t \in \mathbb{R}^n$ denote the price of the assets in period t. Returns earned by holding a unit of each of the assets over time period $(t-1, t]$ is given by $r_t = p_t - p_{t-1}$. We assume a factor pricing model, where the conditional expectation of returns at time $t-1$ is an affine function of L factors in the economy denoted by $f_t \in \mathbb{R}^L$, that is

$$r_t = \bar{r} + Bf_{t-1} + z_t$$

where z_t denotes a zero-mean i.i.d noise process, matrix $B \in \mathbb{R}^{n \times L}$ are the factor loadings, and f_{t-1} is the collection of L factors during period $t-1$. Finally $\bar{r}$ is the intercept term. We write the covariance matrix for z_t as Σ_z. The factors themselves follow a mean reverting process according to

$$f_t = (I - \Phi)f_{t-1} + \eta_t$$

where $\Phi \in \mathbb{R}^{L \times L}$ is a matrix of mean-reversion coefficients for the factors and η_t is an i.id. zero-mean noise process having covariance matrix Σ_η, that is $\mathrm{Var}(\eta_t) = \Sigma_\eta$ for all t.

In every period t, given a discount factor $\rho \in [0, 1)$ we take the reward function

to be

$$
\begin{aligned}
r_t(x_{t-1}, f_{t-1}, \mu_t) &= (1-\rho)^{t+1}(x_{t-1}+\mu_t)^\top \mathbb{E}\left[r_t \mid f_{t-1}\right] \\
&\quad - (1-\rho)^{t+1}\frac{\gamma}{2}(x_{t-1}+\mu_t)^\top \Sigma_z (x_{t-1}+\mu_t) - \frac{(1-\rho)^t}{2}\mu_t^\top \Lambda \mu_t \\
&= (1-\rho)^{t+1}\left(x_t^\top(\bar{r}+Bf_{t-1}) - \frac{\gamma}{2}x_t^\top \Sigma_z x_t\right) - \frac{(1-\rho)^t}{2}\mu_t^\top \Lambda \mu_t
\end{aligned}
$$

where the first term represents the total expected return on our portfolio over the period $(t, t+1]$, the second term is the risk, the third term are transaction costs. Here, γ is a risk aversion parameter. Note that the above reward is the in the objective function of Gârleanu and Pedersen [2013]. Essentially, the dynamic trading strategies we develop trade off total expected portfolio returns with risk and transaction costs.

To highlight that we are being consistent with our earlier framework in section 4.3.2, we remark that the function $h_t(f_{t-1}, \eta_t) = (I-\Phi)f_{t-1} + \eta_t$ in this setting and that the linear dynamics, in (4.22) are such that $A_t \equiv I$, $B_t \equiv I$, $C_t \equiv 0$ and $\epsilon_t \equiv 0$. For the reward function, we have $H_t \equiv B^\top$ and $g_t(x_{t-1}, \mu_t) = (1-\rho)^{t+1}x_{t-1}Bf_{t-1} - (1-\rho)^{t+1}\frac{\gamma}{2}(x_{t-1}+\mu_t)^\top \Sigma_z (x_{t-1}+\mu_t) - \frac{(1-\rho)^t}{2}\mu_t^\top \Lambda \mu_t$.

Finally, now that we stated the overall model, we will now describe all policies that we will evaluate in our case studies, as well as their implementation:

- **Policy Gradient:** Broadly speaking, this is the name we give to all parameterized, projected policies, denoted by π^θ, that we train via maximizing one of the four objective functions given in section 4.3.2. We will now describe exactly how we configure a certain set of these policies, in addition to describing their implementation.

  The general parameterization we use for μ_t^θ is that of a feedfoward neural network with D hidden layers (the depth) and W neurons in each hidden layer (this number is also referred to as the width). We assume that we use different, non-overlapping sets of weights in θ, at each time t.

  As mentioned, we implement a policy gradient algorithm with each of the four objectives listed in section 4.3.2, and we will refer to them as PolicyGradient,

Project, Penalize, and ProjectAndPenalize. To optimize a given objective function, we run the Adam algorithm with a learning rate parameter of 0.01 and update θ using mini-batches of 100 independent sample trajectories. We perform 10,000 training steps and stop the optimization, on the kth iteration early if $\Delta\hat{\nu}(y, \theta^{(k)}) < 10^{-7}$, where $\Delta\hat{\nu}(y, \theta^{(k)})$ is the empirical change in objective value on an independent validation set of 1,000 sample trajectories. For the Penalize and ProjectAndPenalize versions of this algorithm, we use a range of values for λ, namely $\{0.05, 0.01, 0.1, 1.0, 5.0, 10.0\}$.

- **Projected LQR:** This is equivalent to the first policy gradient method discussed in this chapter, as shown in Fazel et al. [2018]. Because the unconstrained version of the problem is an LQR one [Gârleanu and Pedersen, 2013], we can find the optimal LQR policy $\mu^\star$ and then project it as described earlier in this chapter. We will dicuss how the projection is carried out in each of the case studies separately due to the difference in constraints.

- **Deterministic:** By replacing all random variables with their expected values in Problem (4.24), we could efficently solve a static convex optimization problem. In particular, let us define the following general optimization problem with parameters $s \in [T]$, $\mathbf{f} = (f_0, \ldots, f_{T-s+1})$ and $x \in \mathbb{R}^n$:

$$\begin{array}{lll} \underset{\substack{u_1,\ldots,u_{T-s+1}\in\mathbb{R}^n \\ x_0,x_1,\ldots,x_{T-s+1}\in\mathbb{R}^n}}{\text{maximize}} & \displaystyle\sum_{t=1}^{T-s+1} r_{t+s-1}(x_{t-1}, f_{t-1}, u_t) & \\ \text{subject to} & x_t = x_{t-1} + u_t, & t = 1, \ldots, T-s+1 \\ & F_{t+s-1}u_t + G_{t+s-1}x_{t-1} \leq b_t, & t = 1, \ldots, T-s+1 \\ & x_0 = x. & \end{array} \tag{4.26}$$

We let $V_s^D(x, \mathbf{f})$ be function that maps problem parameters x and bf to the optimal objective value for (4.26). Here, s describes the 'starting time period', x is the initial portfolio and $\mathbf{f}$ are some fixed values for the factors. In the above problem, we essentially find an optimal series of trades given we are solving a

determinsitic problem with $T-s+1$ periods, where we know ahead of time the sequence of factors are $\mathbf{f}$ and our initial positions are x.

For the deterministic policy we would compute $V_1^D(x,\ \mathbf{f}^E)$ where $\mathbf{f}^E = (f_0, (I-\Phi)f_0, \ldots, (I-\Phi)^T f_0)$. Given an optimal solution $\mu^\star = (\mu_1^\star, \ldots, \mu_T^\star)$ to the above, we would implement a static policy with these controls from the optimal solution.

- **Deterministic MPC:** This is based on the previous idea except that we would re-solve a new deterministic problem every period s to find the current control. Specifically, at every period s and given we're in state $y_{s-1} = (x_{s-1}, f_{s-1})$, let us define the expected remaining factor sequence as $\mathbf{f}_s^E \triangleq (f_{s-1}, (I-\Phi)f_{s-1}, \ldots, (I-\Phi)^{T-s+1} f_{s-1})$. In period s, would solve the convex problem for $V_s^D(x_{s-1}, \mathbf{f}_s^E)$ defined in (4.26), whose optimal solution we denote by $\mathbf{u}_s^\star = (\mu_{1,s}^\star \ldots, \mu_{T-s+1,s}^\star)$. By taking as the control $\mu_{1,s}^\star$ in every period $s = 1, \ldots, T$ this would exactly define the MPC policy.

In addition to benchmark policies, we also compute upper bounds on the optimal objective value to (4.24) in order to estimate the optimality gap from all the heuristics. The two methods we use for this are:

- **LQR Bound:** By computing the same Riccati equations as in Gârleanu and Pedersen [2013], we calculate the optimal value for the unconstrained LQR version of this problem. Of course, this give us a (in practice, a very loose) upper bound.

- **Perfect-hindsight Bound:** This upper bound is gotten by relaxing all of the non-anticipativity constraints on policies and finding the mean optimal value of the relaxed version of our problem over N independently sampled scenarios. In other words, letting $\mathbf{f}^i = (f_0^i, f_1^i, \ldots, f_T^i)$ denote the ith independent sampled trajectory of the factors, we would compute

$$\frac{1}{N}\sum_{i=1}^{N} V_1^D(x_0, \mathbf{f}^i)$$

where the function V_1^D is defined in (4.26). This is one of the simplest ways to get a (loose) upper bound on the optimal expected value of problems of the form in (4.24).

In the two case studies that follow, we will set the parameters $\Phi, B, \Sigma_z, \Sigma_\eta, \rho, \gamma, \bar{r}, x_0$ and f_0 in different ways. We will also use different choices for the constraint sets C_t, which will illustrate the range of problems our framework is able to capture.

Optimal Execution

For an application of our policy gradient method, we consider the problem of optimal exection, originally explored in Bertsimas and Lo [1998]. The following concrete instance is given in Moallemi and Sağlam [2015].

We consider the problem of liquidating $x_0 = 100,000$ shares of AAPL stock over a trading horizon of 1 hour. There is a trading opportunity every 5 minutes so that the horizon is $T = 12$. Every trade has to be a sell so that $C_t(x_{t-1}) = \{u \in \mathbb{R} : u \leq 0\}$ for all $1 \leq t < T$, moreover x_T must equal zero (the portfolio must be fully liquidated at the end) so that $C_T(x_{T-1}) = \{u \in \mathbb{R} : u \geq 0,\ x_{T-1} + u = 0\}$. With these constraints, the projection operation used in our algorithms is simple enough to carry out with respect to the L_2 norm:

$$\Pi_{C_t(x_{t-1})}(u) = 0 \vee (x_{t-1} \wedge u)$$

where $\vee$ and $\wedge$ denote the componentwise maximum and minimum operators, respectively.

Using historical AAPL prices from January 4, 2010 and January 5, 2010, Moallemi and Sağlam [2015] estimated the following parameters via regressions, which will use in our experiment:

- Factor loadings $B = [0.3375, -0.072]$.
- Intercept term $\bar{r} = 0.0726$.
- Variance in returns $\Sigma_z = 0.0428$.

- Mean reversion coefficients:

$$\Phi = \begin{pmatrix} 0.0353 & 0 \\ 0 & 0.7146 \end{pmatrix}$$

- Transaction cost matrix is assumed proportional to Σ_z, i.e. $\Lambda = \frac{1}{2}\Sigma_z$.

- Covariance matrix of factor changes is

$$\Sigma_\eta = \begin{pmatrix} 0.0378 & 0 \\ 0 & 0.0947 \end{pmatrix}$$

- No discount factor, i.e. $\rho = 0$.

- No risk aversion added in so that $\gamma = 0$.

The factor parameters were taken to be value and momentum signals. Finally to add more randomness into the problem, we will not take the intial factor f_0 to be a deterministic quantity but rather sample it from a multivariate Gaussian with zero mean and covaraince matrix $\sum_{t=0}^{\infty}(I-\Phi)^\top \Sigma_\eta (I-\Phi)$.

Results and discussion We trained several policies using the PG method with different settings of the objective function, the λ-parameter and the general configuration described earlier. We took the best performing policy out of these and call it 'BestPG', which in this case turned out to use $\lambda = 0.01$, utilized neural networks with depth of 3, width of 10 and used the ProjectAndPenalize objective. In addition, we trained a few benchmark policies that also appear in Moallemi and Sağlam [2015] and we give the nicknames TWAP, LQRProject, DetMPC and 'Opt Linear'. The first of these is a simple sell strategy that divides up the trades into equal amounts, i.e. $u_t = -x_0/T$. The second is the projected LQR algorithm described before. The third is the MPC policy also mentioned earlier, and the fourth is the optimal linear policy deribed by Moallemi and Sağlam [2015].

Method	BestPG	TWAP	LQRProject	DetMPC	Opt Linear
Mean	6.43	-8.92	5.70	5.90	6.19
S.E	0.22	0.21	0.22	0.23	0.22
Sim (min)	3.90	0.20	0.47	251.13	3025.11
Train(min)	20.12	0.00	0.00	0.00	0.00

Method	Pathwise	PH-UB	LQR
Mean	6.46	8.43	12.59
S.E	0.22	0.31	N/A

Table 4.6: Upper and lower bounds with existing approaches

We simulate the total wealth from every policy over a common set of 5,000 independent sample problems. Table 4.6 shows the mean objective value, the standard error and the time taken to both 'train' the policy as well as the time taken to actually run it with the simulated problem data. By train, we mean to tune a policy's parameters via Monte Carlo before running it. In the results table we see that the best policy gradient algorithm outperforms the state-of-the-art optimal linear policy by about %4. Moreover, its gap from the pathwise upper bound (the best known upper bound for this problem) is about is 0.5% meaning that the trained policy is nearly optimal. Finally, the training time for policy gradient is modest compared to the time taken solve the optimization problem for the optimal linear policy.

Large-Scale Algorithmic Trading

In this next case study we consider a constrained variation on the benchmark problem from Gârleanu and Pedersen [2013], Glasserman and Xu [2013], which also fits the general model outlined in section 4.3.3 but is much higher dimensional than the execution problem. Here, we develop a *constrained* long term dynamic trading strategy for 15 commodity futures contracts. These are: Aluminum, Copper, Nickel, Zinc, Lead, and Tin from the London Metal Exchange (LME), Gas Oil from the Intercontinental Exchange (ICE), WTI Crude, RBOB Unleaded Gasoline, and Natural Gas

from the New York Mercantile Exchange (NYMEX), Gold and Silver from the New York Commodities Exchange (COMEX), and Coffee, Cocoa, and Sugar from the New York Board of Trade (NYBOT). Our goal is to manage a portfolio of these contracts under a *maximum book size* constraint.

To estimate parameters for this problem, Gârleanu and Pedersen [2013] used the historical prices of all 15 contracts in a ~ 13 year sample period from January 1996 to 2013. Using the historical returns, and three rolling rolling Sharpe Ratios of the aforementioend returns as factors, they estimated the following parameters (some of which were not explicitly provided in the paper and we needed to calculate):

- There are $n = 15$ commodity futures contracts.

- $L = 45$ factors corresponding to the 5-day, 1-year and 5-year rolling average Sharpe ratios for each contract $i \in [15]$ denoted by $f_t^{5D,i}$, $f_t^{1Y,i}$ and $f_t^{5Y,i}$.

- Factor loadings $B \in \mathbb{R}^{45\times 45}$ and offset vector $\bar{r} \in \mathbb{R}^{15}$ defined such that

$$\mathbb{E}\left[r_t^i \mid f_{t-1}^{5D,i}, f_{t-1}^{1Y,i}, f_{t-1}^{5Y,i}\right] = 0.001 + 10.32 f_{t-1}^{5D,i} + 122.34 f_{t-1}^{1Y,i} - 205.59 f_{t-1}^{5Y,i} \quad (4.27)$$

  where the left-hand side is the predicted daily commodity price changes and the right-hand side contains the return predictors corresponding to the 5-day , 1-year and 5-year rolling average sharpe ratios.

- Mean reversion coefficients $\Phi \in \mathbb{R}^{45\times 45}$ defined such that

$$\mathbb{E}\left[f_t^{5D,i} \mid f_{t-1}^{5D,i}\right] = 0.7481 \times f_{t-1}^{5D,i} \quad (4.28)$$

$$\mathbb{E}\left[f_t^{1Y,i} \mid f_{t-1}^{1Y,i}\right] = 0.9966 \times f_{t-1}^{1Y,i} \quad (4.29)$$

$$\mathbb{E}\left[f_t^{5Y,i} \mid f_{t-1}^{5Y,i}\right] = 0.999 \times f_{t-1}^{5Y,i} \quad (4.30)$$

- Return covariance matrix Σ_z estimated from the residuals corresponding to (4.27).

- Factor change covariance matrix Σ_η estimated from the residuals corresponding to eqs. (4.28) to (4.30).
- Transcation cost matrix $\Lambda = (5 \times 10^{-7})\Sigma_z$.
- Discount factor $\rho = 1 - \exp(-0.02/260)$, corresponding to a 2% annualized rate.
- Risk-aversion parameter $\gamma = 10^{-9}$, which Gârleanu and Pedersen [2013] interpret as a relative risk aversion of an agent with $1 billion under management.
- We sampled the initial factor f_0 using a multivariate Gaussian calculated the same way as in section 4.3.3.
- In all our simulations, we fix the initial prices $p_0 \in \mathbb{R}^n_+$ from the prices on January 1, 1996.

Finally, we impose a constraint that the total value of the invested portfolio cannot exceed a limit $W^{max} = \$2\text{M}$ at any given time period. That is,

$$C_t(x_{t-1}) = \left\{ u \in \mathbb{R}^n : \sum_{i=1}^{n} |p_{i,t}^\top (x_{i,t-1} + u_i)| \leq W^{max} \right\}.$$

In order to project proposed controls onto this polyhedron, we need to use as the distance matric the following weighted one:

$$D_{p_t}(u, v) = \sum_{i=1}^{n} |p_{i,t}(u_i - v_i)|.$$

Duchi et al. [2008] provide an $O(n)$ algorithm for a projecting arbtirary vectors in $\mathbb{R}^n$ to an L_1 ball and we leverage it in the following sub-routine to carry out our projection, namely $\Pi_{C_t}(.)$[1]:

1. At any period t, suppose we are given a proposed control $u_t \in \mathbb{R}^n$ by some heuristic algorithm.

[1]This is assuming p_t is positive, we handle the edge case when it might be negative in our code. Notice that under the factor pricing model a simulated price can be negative, which is not the case in the geometric borwnian motion model.

T	Method	Objective	Objective S.E.	Train (m)	Sim (m)
30	BestPG	262,793.4	2,310.4	84.14	1.18
	Det	50,264.8	421.6	0.00	52.59
	DetMPC	120,846.4	481.7	0.00	701.26
	ProjLQR	122,138.2	1,215.3	0.00	2.76
60	BestPG	377,792.1	2,791.2	149.96	2.33
	Det	75,189.7	790.3	0.00	124.53
	ProjLQR	220,693.3	1,999.3	0.00	4.83
90	BestPG	547,058.5	4,629.1	225.58	3.93
	Det	86,384.4	1,080.1	0.00	220.68
	ProjLQR	301,833.5	2,604.6	0.00	6.17

Table 4.7: Overall lower bounds

2. Project the notional values $p_t \cdot u_t$ onto the L_1 ball, where $\cdot$ denotes the componentwise multiplication of two vectors, and call the result $\hat{w}_t$.

3. Return the vector $(\hat{w}_{1,t}/p_{1,t}, \ldots, \hat{w}_{n,t}/p_{n,t})$.

One can show that the above routine is an $O(n)$ algorithm for solving the problem $D_{C_t}(u) = \min_{v \in C_t} D_{p_t}(u, v)$, but we omit the proof.

Results and discussion As before we trained different versions of the policy gradient algorithm by varying the objective function used in training, the parameter λ and neural network topolgy. Unfortunately, due to the significantly larger problem size, we were not always able to simulate the deterministic MPC policy – but whenever we did, we reported its performance. Similarly, we could not simulate the optimal linear policy due to problem size, nor calculate the pathwise upper bounds or perfect hindsight policy. This was all because we were limited to 64GB of RAM and 48 hours of computation time. For this reason this experiment focuses more on practical performance against other implementable algorithms and how effectively our method scales to large instances. We also investigate more closely the effect of setting parameters, such as the objective function, λ-parameter and others, on our algorithm's performance.

Table 4.7 shows the mean objective values and standard errors of all heuristic

T	Method	Objective	Objective S.E.
30	LQCUB	15,527,106.4	0.0
	Perfect Hindsight	634,977.2	2,767.1
60	LQCUB	41,745,614.8	0.0
	Perfect Hindsight	1,107,307.4	6,454.4
90	LQCUB	64,503,457.0	0.0
	Perfect Hindsight	1,364,200.7	10,326.7

Table 4.8: Overall upper bounds

T	PGType	Objective	Objective S.E.
30	Penalize	206,891.6	1,516.1
	PolicyGradient	78,676.6	568.8
	Project	186,322.8	2,748.4
	ProjectPenalize	262,793.4	2,310.4
60	Penalize	368,619.8	2,515.9
	PolicyGradient	122,318.0	1,494.1
	Project	297,030.1	4,050.7
	ProjectPenalize	377,792.1	2,791.2
90	Penalize	410,563.8	1,966.0
	PolicyGradient	162,260.8	981.7
	Project	324,883.0	7,497.0
	ProjectPenalize	547,058.5	4,629.1

Table 4.9: Breakdown by PG type

policies that we are able to implement. The corresponding LQR lower bounds, which are quite weak, are shown in Table 4.8. The best policy gradient algorithm used the ProjectAndPenalize objective, used neural networks with 30 hidden neurons in a single layer, and set $\lambda = 0.05$. We see that policy gradient sometimes beats the closest competitor 'Projected LQR' by more than 100%, and in all other cases by at least 50%. Policy gradient turns out to be a strong algorithm here and with manageable training times that extend to just a few hours even with large horizons such as $T = 90$.

In addition to showing the overall mean wealth generated by the different strategies, we also break down the performance of policy gradient by objective function. This parameter turns out to make a huge difference as demonstrated in Table 4.9. We see that the most effective objective function is ProjectAndPenalize, followed usually

by Penalize. Our explanation for this is that penalization and projection both work in tandem to guide gradient descent methods into policies that both satisfy constraints and earn large rewards. Penalization encourages constraint satisfaction and projection ensures that we don't explore sample paths that would not be generated by a feasible and optimal policy.

Given these impressive numerical results, we attempt to analyze the policy gradient algorithm and give some conjectures in the following section.

4.3.4 Theory and Conjectures

In this section our goal is to show that the general problem in (4.24) has a special enough structure that we can guarantee that the optimal policy is continuous in state, or that the value function is concave. The former is a useful property because it shows that a neural net can, in principle, fit an optimal policy arbitrarily well with enough width and depth. Moreover, the fact that an optimal value function is concave can, in principle, reduce the search space for good approximations. We start with proving the following Lemma which contains most of the technical groundwork needed to prove our main result.

Lemma 15. *Let $\eta \in \mathbb{R}^n$ and $\theta \in \mathbb{R}^k$ be parameter vectors and $g : \mathbb{R}^{2n+k} \mapsto \mathbb{R}$ be a twice-continuously differentiable function, which is strictly, jointly convex in its first n and last k components. That is, for any η, $g(x, \eta, \theta)$ is jointly and strictly concave in x and θ.*

Fix any matrices $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m, F \in \mathbb{R}^{m\times k}$ and consider the convex optimization problem

$$\begin{aligned} V(\eta, \theta) := \underset{x\in\mathbb{R}^n}{\text{maximize}} \quad & \eta^\top x - g(x, \eta, \theta) \\ \text{subject to} \quad & Ax \leq b + F\theta. \end{aligned} \tag{4.31}$$

Let $\mathcal{G} \subset \mathbb{R}^n \times \mathbb{R}^k$ denote the set of possible parameter values η, θ that result in a feasible, bounded problem. Define the function $x^\star(\eta, \theta)$, where $x^\star : \mathcal{G} \mapsto \mathbb{R}^n$, to be the unique optimal solution with the given parameters. Then, we have that

1. *The optimal solution $x^\star(\eta, \theta)$ is a continuous function of both η and θ.*

2. For any $\eta \in \mathbb{R}^n$, the value function $V(\eta, \theta)$ is a concave function of θ.

Proof. To prove the first part, let us first write out the KKT conditions by introducing dual variables $\mu \in \mathbb{R}^m$

$$\nabla_x g(x, \eta, \theta) = \eta - \mu^\top A \tag{4.32}$$

$$Ax \leq b + F\theta \tag{4.33}$$

$$\mu \geq 0 \tag{4.34}$$

$$\mu_i \left(b_i + F_i^\top \theta - A_i^\top x\right) = 0,\ i = 1, \ldots, m. \tag{4.35}$$

Since g is strictly concave in x, we have that the Jacobian of $\nabla_x g(x, \eta, \theta)$ is positive definite. Thus the determinant of the Jacobian is always non-zero. Therefore, by the inverse function theorem, there exists an inverse function $h : \mathbb{R}^{2n+k} \mapsto \mathbb{R}^n$ such that $h(y, \eta, \theta) = x \Leftrightarrow y = \nabla g_x(x, \eta, \theta)$. Now let $\tilde{\mu}(\eta, \theta)$, $\hat{\mu}(\eta, \theta)$ denote vectors of optimal dual variables corresponding to active and inactive constraints, respectively. Let, similarly, $\tilde{A}$, $\tilde{F}$ and $\tilde{b}$ denote the parts of matrices A, F and b whose rows correspond to the active constraints. For the inactive constraints, we know that $\hat{\mu}(\eta, \theta) = 0$.

Then, by substituting in (4.32) into (4.35), we have that

$$\begin{aligned} \tilde{b} + \tilde{F}\theta = \tilde{A} h(\tilde{\mu}(\eta, \theta)^\top \tilde{A} - \eta, \eta, \theta) \Leftrightarrow \\ \tilde{\mu}(\eta, \theta)^\top = \nabla_x g((\tilde{A})^{-1}(\tilde{b} + \tilde{F}\theta) + \eta, \eta, \theta)(\tilde{A})^{-1}. \end{aligned} \tag{4.36}$$

By doing further substitution, we derive that

$$x^\star(\eta, \theta) = h\left(\nabla_x g((\tilde{A})^{-1}(\tilde{b} + \tilde{F}\theta) + \eta, \eta, \theta) - \eta, \eta, \theta\right).$$

The largest set of parameters that produce an optimal solution with the active con-

straints, shown above, is defined to be a critical region

$$\mathcal{R}_{\tilde{A}} \triangleq$$
$$\left\{(\eta,\theta) \in \mathcal{G} : \tilde{\mu}(\eta,\theta) \geq 0,\ Ah\left(\nabla_x g((\tilde{A})^{-1}(\tilde{b}+\tilde{F}\theta)+\eta,\eta,\theta)-\eta,\eta,\theta\right) \leq b+F\theta\right\}. \tag{4.37}$$

Within the critical region, it's clear since g is twice-continuously differentiable, that $x^\star(\eta,\theta)$ is continuous. At the boundary between two or more regions, the optimal solution can be expressed in terms of different sets of active constraints. However, because the optimal solution is always unique (due to strict concavity of the function g), the optimal solution must be continuous across the boundary between regions.

The proof for the second statement is immediate from Lemma 2.1 in Bemporad and Filippi [2006]. □

The above immediately enables us to prove the following key proposition. Essentially, it shows that because of the structure in our problem, and the fact that our controls are continuous as opposed to discrete, means that the optimal policy is a continuous function of x_{t-1}. One could easily come up with examples of control problems (with finite action spaces), where even if we expressed a policy as a probability distribution over actions (i.e. the policy takes values in a simplex), the policy would not be a continuous function of state. Thus in principle a neural network might not be able to represent an optimal policy in our problem fairly accurately. In any case, we state the result in question below:

Proposition 6. *Consider Problem 4.24 and assume that $g_t(x_{t-1},u)$ is strictly concave. Let $\mu_t^\star(x_{t-1},f_{t-1})$ denote the optimal control at time t, then $\mu_t^\star(x_{t-1},f_{t-1})$ is a continuous function of its inputs for all times t. Moreover, the optimal value function $V_t^\star(x_t,f_t)$ is concave in x_t for all $0 \leq t < T$.*

Proof. Let's start with the base case. When $t = T$, we're solving

$$\begin{aligned} V^{\star}_{T-1}(y_{T-1}, w_{T_1}) = \underset{u}{\text{maximize}} \quad & f_{T-1}^{\top} u - g_T(x_{T-1}, u) \\ \text{subject to} \quad & F_T u + G_t x_{T-1} \leq b_T \end{aligned} \tag{4.38}$$

and the result is immediate from Lemma 15 once we notice that we can substitute in x_{T-1} for θ (where θ is the variable used in Lemma 15). Now take some time period $t < T$ and assume the induction hypothesis for $t + 1$. Then we have that

$$\begin{aligned} V^{\star}_{t-1}(x_{t-1}, f_{t-1}) = \underset{u}{\text{maximize}} \quad & f_{t-1}^{\top} u - g_t(x_{t-1}, u) + Q_t(x_{t-1}, f_{t-1}, u) \\ \text{subject to} \quad & F_t u + G_t x_{t-1} \leq b_t \end{aligned} \tag{4.39}$$

where we used the Q function abbreviation of the following expression

$$\begin{aligned} Q_t(x_{t-1}, f_{t-1}, u) &= \mathbb{E}\left[V_t(A_t x_{t-1} + B_t u + C f_{t-1} + \epsilon_t, f_t) \mid f_{t-1}\right] \\ &= \mathbb{E}\left[V_t(A_t x_{t-1} + B_t u + C f_{t-1} + \epsilon_t, h(f_{t-1}, \eta_t)) \mid f_{t-1}\right]. \end{aligned}$$

It then sufficies to simply show that the Q function is jointly concave in u and x_{t-1}, after which we can apply Lemma 15. This follows because for each realization of the noise ϵ_t, η_t, the function $V_t(A_t x_{t-1} + B_t u + C f_{t-1} + \epsilon_t, f_t)$ is concave in u and x_{t-1} due to the induction hypothesis. Finally, taking expecations over ϵ_t and η_t preserves the concavity. □

We finally make the following conjecture based on the previous result that was just proved.

Conjecture 1. *Let $\mu^{\star}$ denote the optimal policy for Problem 4.24 and $\epsilon > 0$. Assume that the linear constraints in* (4.23) *are given such that each set of constraints, at every time t, defines a bounded polytope. Then there exists a neural network large enough and a parameter $\lambda > 0$ such that if*

$$\theta^{\star} \in \arg\max_{\theta} \; V^{\Pi}(\theta, y),$$

then $||\mu_t^{\theta^\star} - \mu_t^\star||_\infty < \epsilon$ *for all* $t = 1, \ldots, T$.

We suspect the conjecture is true because of Proposition 1 and the results in Cybenko [1989], Hornik et al. [1989] that show that any continuous function with a bounded domain can be approximated abritrarily well by a feedforward neural network in terms of the L_∞-norm.

While it remains for this conjecture to be rigorously shown, if it were true it would mean the neural network parameterization is rich enough to nearly recover an optimal policy. Unfortunately, this wouldn't tell us about how good this approximation could be in practice and since the optimization problem remains non-convex we might not be able to give guarantees on the closeness to the solution gotten via our penalized policy-gradient method to a globally optimal solution.

Chapter 5

Conclusion

This thesis explored the enormous subject of large-scale control from different angles, and covered a range of problems their and applications. We saw that solving dynamic programs can be directly useful in applications such as portfolio optimization, collateral management and option pricing, but also can be effectively leveraged to minimize regret in multi-armed bandits via a Bayesian formulation. A key conclusion from this thesis is that, ultimately, what worked best when solving these immensely challenging problems was one of two general approaches that we outline below.

- The first strategy is to carefully examine the most important features of the problem and see if 'very simple' heuristics make sense. This helped when tackling the Collateral Management Problem since we were able to find a connection between greedy policies and the pricing of collateral via duality. The advantage of basic, myopic policies is they're usually efficient to implement and are easier to analyze and reason about. Such an strategy seems most relevant/effective when the problem has complex dynamics that do not allow it to fit into the general frameworks. As for other 'very simple' heuristics, we also saw in Chapter 2 that the limited lookahead trick, in the context of computing Gittins indices (and solving the corresponding stopping problem), provided good approximations to the indices.

- The second strategy, is to see if the problem can formulated in a way that it fits

some general framework such as that of quasi-linear convex control. We found evidence that this class of problems can be effectively solved through popular RL algorithms such as policy gradient, albeit with a modification to the usual algorithm. Another general framework, for which we know good RL solutions, is that of optimal stopping. What's useful is that virtually any kind of option pricing problem can be handled through it.

Before we state future research directions and wrap up this thesis, we will recap what was ultimately achieved during the course of this work.

In Chapter 2, we proposed a novel way for designing Bayesian Multi-Armed Bandit algorithms by treating the problem of minimizing regret as a sequence of separate Markov Decision problems where the discount factor increases from one problem to the next, according to a carefully chosen rate. We showed that the fundamental idea of using such a heuristic results in sub-linear regret and, when applied to a binary bandit problem, that a simple and efficient algorithm with a flat Beta prior achieves the optimal rate of growth in regret.

In Chapter 3, we formulated what we believe is a new and practical model of managing a Prime Broker's business. Based on the model, we stated a dynamic optimization problem that focuses on the question of what collateral a Prime Broker should hypothecate from eligible clients over time, and when it should borrow assets externally. At first glance, the problem is intractable due to it being a high-dimensional dynamic one. In order to address the problem, we propose a practical scheme based on estimating the long-run dual values of assets in a stable, 'ergodic' regime. In particular, we have shown that under such conditions, the algorithm in question is not only asymptotically optimal but enjoys, for many non-trivial cases, a constant optimality gap. Our other contribution is a set of simulation benchmarks based, either fully or in part, on a Prime Broker's data. By running our algorithm and similar competing ones with the data, we have shown that our method can offer notable increases in revenue for the Prime Broker. Moreover, our new simulation benchmark can of course be used and adapted to simulate new collateral management algorithms.

In Chapter 4, we explored and tested general RL methodologies and their applications to financial engineering problems. For option pricing, we uncovered a novel dual-martingale method for computing upper bounds on price via deep learning.

5.1 Future research directions

We now conclude this thesis by listing open questions and possible, future research directions pertaining to each of the main topics:

- **Optimistic Gittins Indices:** First, it remains to be proven that playing arms with maximum (exact) Gittins indices together with the increasing discount factor schedule, does produce an algorithm whose regret matches the Lai-Robbins lower bound. We have a strong reason to suspect this is true due to the findings in our numerical experiments.

  Secondly, it is worth exploring whether the idea of the OGI framework can be extended to contextual bandit problems where dependencies between arms exist. In our setting, the fact that arms were independent allowed us to exploit the Gittins index but there could be other ways to approximate optimal solutions to bandit problems with dependent arms.

- **Collateral Management:** There are several extensions to the model that we think are useful to implement. We reckon all of them could be approached using the idea of estimating the 'right' dual asset prices:

  - Firstly, we have so far focused on the aspect of re-hypothecation where the Prime Broker specifically re-uses the collateral to satisfy future client borrowing demands. It's also possible that Prime Broker would be faced with collateral requirements from external lenders, when they borrow their own stock, and this is a feature of the model that's not yet captured.
  - Secondly, we have implicitly assumed that all client loans in the problem have a term that's longer than the problem horizon; otherwise it would be

possible for assets to return to the Prime Broker after the corresponding loans expired, and also for the collateral that the Prime Broker hypothecated to also be returned to the original client. This is yet again a difficulty we have avoided here, but it would be useful to address fully.

  - Another practically relevant question is whether it's possible to stitch together a network of similar problems as the one considered in the paper. Then the challenge would be finding an equilibrium strategy among multiple agents for how they set borrowing fees and hypothecate collateral. The results of such a work, would likely yield insights on how the securities lending industry as a whole could function better.

- **Deep Reinforcement Learning in Finance:** This chapter offers us a myriad of interesting, open theoretical problems. For example, in the context of the general policy gradient method applied to constrained problems, can we prove a similar result to Fazel et al. [2018], namely that gradient descent recovers at least a local minimum? Another interesting result to prove/disprove, in this space, is whether policy gradient finds the globally optimal policy, if we are to restrict ourselves to the space of *linearly parameterized* ones (like in Moallemi and Sağlam [2015]).

  In general, the great challenge in deep learning is proving anything about how close to global optimality the local minimum solutions we get are, so we appreciate the full potential of neural networks. Also, it would be useful to prove anything about how deep, or wide, a neural network needs to be in order to better approximate various functions and, particularly, within the context of our deep RL applications where we search for good parameterized policies or martingale upper bounds.

Appendix A

Supplementary Material for Optimistic Gittins Indices

A.1 Proof of Lemma 1

Proof. Consider an instance of the MAB with $A = 2$ arms and Bernoulli rewards. We assume that the prior on arm 1 is degenerate with mean $\lambda = 1/2$ while arm 2 has a Beta(α, α) prior where α is a parameter we set later. Then, it is simple to check that $\pi^{G,\gamma}$ must pull arm 2 at the first time period. With probability 1/2, we receive a reward of 0, so that the posterior on arm 2 is given by a Beta$(\alpha, \alpha+1)$ prior. Now, the continuation value from pulling arm 1 at this stage is lower bounded by $\frac{1/2}{1-\gamma}$, while the continuation value from pulling arm 2 is upper bounded by

$$\frac{\alpha}{1+\alpha} + \frac{\gamma \mathbb{E}\left[\max\left(R(y_{2,0}), 1/2\right) \middle| y_{2,0} = (\alpha, \alpha+1)\right]}{1-\gamma}.$$

It follows that any optimal policy must pull arm 1 if

$$\frac{1/2}{1-\gamma} > \frac{\alpha}{1+\alpha} + \frac{\gamma \mathbb{E}\left[\max\left(R(y_{2,0}), 1/2\right) \middle| y_{2,0} = (\alpha, \alpha+1)\right]}{1-\gamma},$$

an inequality which in turn is satisfied if

$$1/2 > \frac{\alpha}{1+\alpha} + \frac{\gamma \mathbb{P}\left(R(y_{2,0}) > 1/2 \,\middle|\, y_{2,0} = (\alpha, \alpha+1)\right)}{1-\gamma},$$

But the right hand side of the above expression goes to 0 as $\alpha \to 0$. Consequently, we can choose an α such that any optimal policy (for the discounted infinite horizon problem) chooses to pull the first arm; let α^* be the largest such α. Since the state of the first arm does not change (the prior on that arm was assumed degenerate), the same condition must hold at subsequent iterations. Consequently, $\pi^{G,\gamma}$ must incur T-period regret lower bounded by $\frac{T}{2}\mathbb{E}\left[(R((\alpha^*, \alpha^*+1)) - 1/2)^+\right]$. The result follows. □

A.2 Proof of Proposition 1

Proof. Recall that the policy π^D uses a doubling trick, which boils down to executing the policy $\pi^{G,1-1/2^{k-1}}$ during periods $\{2^{k-1}, \ldots, 2^k - 1\}$ for each $k \in \mathbb{N}$. Consider the kth epoch and let $n \triangleq 2^{k-1}$, so that that this epoch lasts from periods n to $2n-1$, during which the algorithm employs a fixed discount factor $\gamma_n = 1 - 1/n$.

We begin by bounding the regret just in the kth epoch. At the beginning of the epoch, the policy's information set is $\mathcal{F}_{n-1}$. It is straightforward to show that this is equivalent to the σ-algebra generated by the tuple of sufficient statistics at time n, namely $(y_{1,N_1(n-1)}, \ldots, y_{A,N_A(n-1)})$, which we will write as a random vector $\mathbf{y}_{n-1} \in \mathcal{Y}^A$. We will generally write the sufficient statistics vector at an any time t as $\mathbf{y}_{t-1}$.

For bounding the regret just between the periods n and $2n-1$, it will be helpful to define the notation

$$S_{m_1,m_2}(\pi, \theta) \triangleq \sum_{t=m_1}^{m_2} X_{\pi_t, N_{\pi_t}(t)}$$

as the total rewards accumulated between times m_1 and m_2 by a policy π, when the rewards from the ith arm are distributed according to p_{θ_i}. When there is just a single

subscript m, we let

$$S_m(\pi, \theta) \triangleq S_{1,m}(\pi, \theta)$$

denote, simply, the total rewards from the time 1 up to time m. By the Markovian nature of π^D we are able to conveniently characterize the conditional expected regret in epoch k in the following way. In particular, for any tuple of sufficient statics for the arms $\hat{\mathbf{y}} \in \mathcal{Y}^A$, we have

$$\begin{aligned} \mathbb{E}\left[S_{n,2n-1}(\pi^D, \theta) \,\middle|\, \mathbf{y}_{n-1} = \hat{\mathbf{y}} \right] &= \mathbb{E}\left[S_n(\pi^{G,\gamma_n}, \theta) \,\middle|\, \mathbf{y}_0 = \hat{\mathbf{y}} \right] \\ &\triangleq \mathbb{E}_{\hat{\mathbf{y}}}\left[S_n(\pi^{G,\gamma_n}, \theta) \right], \end{aligned} \tag{A.1}$$

which is a stationarity property that also follows from the fact that π^D uses a fixed discount factor of γ_n on the interval $n, \ldots, 2n-1$. We pause to parse the above equation. Recall that any Bayesian policy tracks sufficient statistics, which is a sequence of random vectors $(\mathbf{y}_t,\ t \in \mathbb{N})$. Thus far, we have always assumed that $\mathbf{y}_0 = \mathbf{y}$ is defined in terms of a global prior q corresponding to a tuple of statistics $\mathbf{y}$. We now depart from this convention to say that the initial statistic can be anything else, and so the prior distribution on θ, which the Gittins policy is aware of, is given by a possibly different prior over the arms $(\hat{q}_1, \ldots, \hat{q}_A)$ corresponding to the vector $\hat{\mathbf{y}}$.

For this step in the proof, we let $H \sim \text{Geo}(1/n)$ be an exogenous geometric random variable that is independent of θ and not observed by the agent. The reason we introduce it is because it can shown that for any $\mathbf{y}' \in \mathcal{Y}^A$, that

$$V^*_\gamma(\mathbf{y}') = \mathbb{E}_{\mathbf{y}'}\left[S_H(\pi^{G,\gamma_n}, \theta) \right],$$

which means that the Gittins policy is optimal for the random horizon H given the prior corresponding to $\mathbf{y}'$. Now let us fix again an arbitrary statistic $\hat{\mathbf{y}}$ (which

corresponds to some prior distribution over θ). We have

$$\begin{aligned}
\mathbb{E}_{\hat{\mathbf{y}}}\left[S_H(\pi^{G,\gamma_n},\theta)\right] &= \mathbb{E}_{\hat{\mathbf{y}}}\left[H\mu^*(\theta) - \text{Regret}\left(\pi^{G,\gamma_n},H,\theta\right)\right] && \text{(A.2)}\\
&= n\mathbb{E}_{\hat{\mathbf{y}}}\left[\mu(\theta^*)\right] - \mathbb{E}_{\hat{\mathbf{y}}}\left[\text{Regret}\left(\pi^{G,\gamma_n},H,\theta\right)\right]\\
&\leq n\mathbb{E}_{\hat{\mathbf{y}}}\left[\mu(\theta^*)\right] - \mathbb{E}_{\hat{\mathbf{y}}}\left[\text{Regret}\left(\pi^{G,\gamma_n},H,\theta\right) \mid H > n\right]\mathbb{P}\left(H > n\right)\\
&\leq n\mathbb{E}_{\hat{\mathbf{y}}}\left[\mu(\theta^*)\right] - \mathbb{E}_{\hat{\mathbf{y}}}\left[\text{Regret}\left(\pi^{G,\gamma_n},n,\theta\right)\right](1-1/n)^n\\
&= n\mathbb{E}_{\hat{\mathbf{y}}}\left[\mu(\theta^*)\right] - \mathbb{E}_{\hat{\mathbf{y}}}\left[\text{Regret}\left(\pi^{G,\gamma_n},n,\theta\right)\right](e^{-1}+o(1)). && \text{(A.3)}
\end{aligned}$$

By Theorem 3, of Lai [1987], there exists (an efficient) policy $\tilde{\pi}$, such that

$$\mathbb{E}_{\hat{\mathbf{y}}}\left[\text{Regret}\left(\tilde{\pi},n,\theta\right)\right] \leq \mathbb{E}_{\hat{\mathbf{y}}}\left[c(\theta)\right]\log^2 n,$$

where $c(\theta)$ is a function of θ. Let $\Delta(\theta)$ denote worst case single period regret, that is, $\Delta(\theta) = \max_i \mu(\theta^*) - \mu(\theta_i)$. Using this notation, we obtain the lower bound,

$$\begin{aligned}
\mathbb{E}_{\hat{\mathbf{y}}}\left[S_H(\pi^{G,\gamma_n},\theta)\right] &\geq \mathbb{E}_{\hat{\mathbf{y}}}\left[S_H(\tilde{\pi},\theta)\right] && \text{(A.4)}\\
&= \mathbb{E}_{\hat{\mathbf{y}}}\left[H\mu(\theta^*) - \text{Regret}\left(\tilde{\pi},H,\theta\right)\right]\\
&\geq \mathbb{E}_{\hat{\mathbf{y}}}\left[H\mu(\theta^*) - \text{Regret}\left(\tilde{\pi},H,\theta\right)\mathbb{1}\left(H \geq e\right) - 2\mathbb{1}\left(H < e\right)\Delta(\theta)\right]\\
&\geq n\mathbb{E}_{\hat{\mathbf{y}}}\left[\mu(\theta^*)\right] - \mathbb{E}_{\hat{\mathbf{y}}}\left[c(\theta)\right]\mathbb{E}\left[(\log(H))^2\mathbb{1}\left(H \geq 3\right)\right]\\
&\quad - 2\mathbb{E}_{\hat{\mathbf{y}}}\left[\Delta(\theta)\right]\\
&\geq n\mathbb{E}_{\hat{\mathbf{y}}}\left[\mu(\theta^*)\right] - \mathbb{E}_{\hat{\mathbf{y}}}\left[c(\theta)\right]\mathbb{E}\left[(\log(H))^2 \mid H \geq 3\right]\mathbb{P}\left(H \geq 3\right)\\
&\quad - 2\mathbb{E}_{\hat{\mathbf{y}}}\left[\Delta(\theta)\right]\\
&\geq n\mathbb{E}_{\hat{\mathbf{y}}}\left[\mu(\theta^*)\right] - \mathbb{E}_{\hat{\mathbf{y}}}\left[c(\theta)\right]\log^2(n+3)\mathbb{P}\left(H \geq 3\right) - 2\mathbb{E}_{\hat{\mathbf{y}}}\left[\Delta(\theta)\right] && \text{(A.5)}
\end{aligned}$$

where (A.4) holds by optimality of the Gittins index. The bound (A.5) follows from the memoryless property of the Geometric distribution, from Jensen's inequality and the fact that function $\log^2 x$ is a concave function on $[e, +\infty)$.

Our next step will involve marginalizing the distribution of θ over possible future values of $\mathbf{y}_{n-1}$. Notice that for any measurable function $Y(\theta)$ that can depend on θ,

we have by the law of iterated expectation that

$$\mathbb{E}_{\mathbf{y}}\left[\mathbb{E}\left[Y(\theta) \mid \mathbf{y}_{n-1}\right]\right] = \mathbb{E}_{\mathbf{y}}\left[Y(\theta)\right]. \tag{A.6}$$

We now turn our attention back to the Bayes' regret incurred by π^D between times n and $2n-1$, and we are ready to bound it as follows. Define for any statistic $\hat{\mathbf{y}}$, policy π, and $T \in \mathbb{N}$, the real-valued function $\text{Regret}(\pi, T, \hat{\mathbf{y}}) \triangleq \mathbb{E}_{\hat{\mathbf{y}}}\left[\text{Regret}\left(\pi, T, \theta\right)\right]$, then we have

$$\begin{aligned}
\mathbb{E}_{\mathbf{y}}\left[n\mu^*(\theta) - S_{n,2n-1}(\pi^D)\right] &= \mathbb{E}_{\mathbf{y}}\left[\mathbb{E}\left[n\mu^*(\theta) - S_{n,2n-1}(\pi^D) \,\middle|\, \mathbf{y}_n\right]\right] \\
&= \mathbb{E}_{\mathbf{y}}\left[\text{Regret}(\pi^{G,\gamma_n}, n, \mathbf{y}_n)\right] && \text{(A.7)} \\
&\leq \mathbb{E}_{\mathbf{y}}\left[\mathbb{E}_{\mathbf{y}_n}\left[c'(\theta)\log^2(n)\right]\right] && \text{(A.8)} \\
&= C_q \log^2 n && \text{(A.9)}
\end{aligned}$$

where $c'(\theta)$ is a constant that depends on the parameter θ and C_q is a constant that depends only on the prior distribution q. Equation (A.7) follows from (A.1), while (A.8) follows from (A.3) and (A.5). The final equation (A.9) uses (A.6). We use the above bound on the kth epoch's regret to bound the complete regret in all of the epochs up to T, as follows,

$$\begin{aligned}
\text{Regret}\left(\pi^D, T\right) &\leq \text{Regret}\left(\pi^D, 2^{\lceil \log_2 T \rceil}\right) \\
&= \sum_{k=1}^{\lceil \log_2 T \rceil} \left(2^{k-1}\mathbb{E}_{\mathbf{y}}[\mu(\theta^*)] - \mathbb{E}_{\mathbf{y}}\left[S_{2^{k-1},2^k-1}(\pi^D)\right]\right) \\
&= O\left(\sum_{k=1}^{\lceil \log_2 T \rceil} k^2\right) \\
&= O(\log^3 T)
\end{aligned}$$

and the result is shown. □

A.3 Properties of the Optimistic Gittins index

This section gives proofs for a few properties of the Optimistic Gittins index that are used throughout the thesis and particularly in the proof of Theorem 1. It shall be useful, in what follows, to define the continuation value for the Whittle's retirement problem (Whittle [1980]) as

$$V_\gamma(y,\lambda) \triangleq \sup_{\tau>0} \mathbb{E}_y \left[\sum_{t=1}^{\tau} \gamma^{t-1} X_{i,t} + \gamma^\tau \frac{\lambda}{1-\gamma} \right],$$

so that the Gittins index is then the solution in λ to $\lambda/(1-\gamma) = V_\gamma(y,\lambda)$. In an analogous fashion, we define the optimistic continuation value, for parameters K and λ, to be

$$V_\gamma^K(y,\lambda) \triangleq \sup_{1\le\tau\le K} \mathbb{E}_y \left[\sum_{t=1}^{\tau} \gamma^{t-1} X_{i,t} + \gamma^\tau \frac{R_{\lambda,K}(\tau, y_{i,\tau-1})}{1-\gamma} \right].$$

From this definition, it follows that the solution for λ to the equation $\lambda/(1-\gamma) = V_\gamma^K(y,\lambda)$ is the Optimistic Gittins index.

Throughout this section, we will sometimes discuss the value of the index at some particular time t during the execution of the algorithm, which depends on the statistic gathered about the arm using information up to but strictly *not including* time t. As such, we will define the number of pulls of arm i up to time $t-1$ as

$$P_i(t) \triangleq N_i(t-1)$$

where we recall $N_i(t)$ is the counter for the number of total number of pulls up to and including t. From the $P_i(t)$ pulls of the arm, the total reward accumulated is defined as

$$S_i(t) \triangleq \sum_{s=1}^{P_i(t)} X_{i,s}.$$

We begin by investigating the effect of the parameter λ, which gives the deterministic payoff in (2.2), on the continuation value $V_\gamma^K(y,\lambda)$ and use that to find out how close an approximation $v_\gamma^K(y)$ is to the Gittins index.

Fact 1. *For any state $y \in \mathcal{Y}$, discount factor γ and parameter K, the function $V_\gamma^K(y, \lambda)$ is convex in λ.*

Proof. Fix an arbitrary state y and discount factor $\gamma \in (0, 1)$. We prove convexity by induction on the parameter K. For $K = 1$, recall from Section 2.2 that

$$V_\gamma^1(y, \lambda) = \mathbb{E}_y \left[X_{i,1}\right] + \gamma \mathbb{E}_y \left[\max(\lambda/(1-\gamma), R(y_{i,0}))\right].$$

Thus the function is convex because it is an expectation over a convex piecewise linear function of random variables $X_{i,1}$ and $R(y_{i,0})$.

Now we show the inductive step. For any $K > 1$, assume that $V_\gamma^{K-1}(y, \lambda)$ is convex. By writing the Bellman equation,

$$V_\gamma^K(y, \lambda) = \mathbb{E}_y \left[X_{i,1}\right] + \gamma \mathbb{E}_y \left[\max\left(\lambda/(1-\gamma), V_\gamma^K(y_{i,1}, \lambda)\right)\right],$$

we again notice an expectation over a maximum of convex functions in λ. This form for $V_\gamma^K(y, \lambda)$ implies that it is also convex in λ. □

Lemma 16. *Suppose that arm rewards are bounded. That is, there exists a constant $B \in \mathbb{R}_+$ such that $X_{i,t} \in [0, B]$ for every arm i and time t.*

Let $v_{i,t}^K$ be the Optimistic Gittins Index of arm i at time t and let η be a scalar, then the following equivalence holds

$$\{v_{i,t}^K < \eta\} = \{(1-\gamma_t) V_{\gamma_t}^K(y_{i,P_i(t)}, \eta) < \eta\}$$

where $y_{i,P_1(t)}$ is the sufficient statistic for estimating the ith arm's parameter θ_i at time t.

Proof. First of all note that for any state y and discount factor γ, the function

$$\begin{aligned}(1-\gamma)V_\gamma^K(y, \lambda) - \lambda \\ = \sup_{1 \le \tau \le K} \mathbb{E}_y \left[\sum_{s=1}^{\tau} \gamma^{s-1}(1-\gamma) X_{i,s} + \mathbb{1}\left(\tau = K\right) \gamma^\tau (R(y_{i,\tau-1}) - \lambda)^+\right], \quad \text{(A.10)}\end{aligned}$$

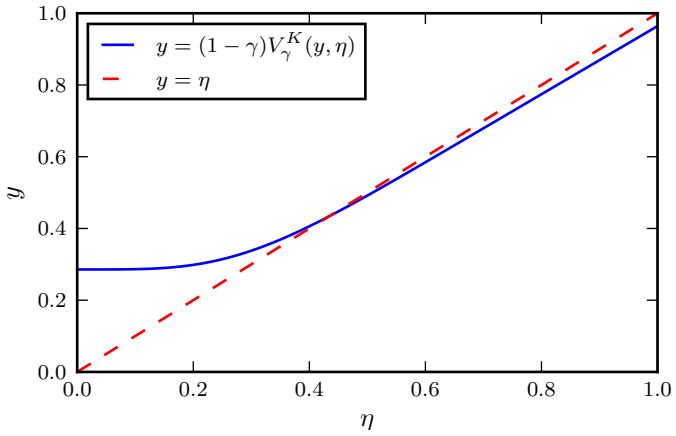

Figure A-1: Visualization of Lemma 16's proof for a instance of the problem with a Beta prior corresponding to the pair $y = (4, 5)$, a discount factor of $\gamma = 0.95$ and $K = 2$. The intersection of the two lines represents the Optimistic Gittins index.

is convex (from Fact 1) yet decreasing in λ. Also notice that at $\lambda = 0$

$$V_\gamma^K(y, 0) = \frac{\mathbb{E}_y[X_{i,1}]}{(1-\gamma)} \geq 0$$

because it is never optimal to retire in the stopping problem. Also, in the other extreme case when $\lambda = B$, the function in question evaluates to

$$V_\gamma^K(y, B) = \mathbb{E}_y[X_{i,1}] + \frac{\gamma B}{(1-\gamma)} \leq \frac{B}{(1-\gamma)}.$$

Thus, consider again the above function of λ, namely, $(1-\gamma)V_\gamma^K(y, \lambda) - \lambda$. Such a function is non-negative for any $\lambda \leq v_\gamma^K(y)$ (since $v_\gamma^K(y)$ is the root of the function) and is also negative for $\lambda > v_\gamma^K(y)$. From these observations, and the fact that y and γ were arbitrary, the result follows. Figure A-1 provides a visualization of this proof. □

A.3.1 Proof of Lemma 2

Proof. Let $K < M$ be two look-ahead parameters used in the definition of OGI. We will show that $V_\gamma^K(y, \lambda) \leq V_\gamma^M(y, \lambda)$ where we recall the definitions of these functions from the beginning of Section A.3.

We begin with a fundamental step. Let τ_1 and τ_2 be any predictable stopping times (i.e. $\mathcal{F}_{t-1}$-measurable random times) such that τ_1 precedes τ_2 almost surely, that is $\tau_1 < \tau_2$. Recall that the expected reward of the ith arm satisfies $\mathbb{E}\left[X_{i,t} \mid \theta_i\right] = \mu(\theta_i)$ for all t. Let $\hat{\theta}_i \in \Theta$ denote a realization of the random variable θ_i and let $\zeta(\hat{\theta}_i)$ be a real-valued, measurable function of $\hat{\theta}_i$. In this case, we have that

$$\begin{aligned}\mathbb{E}\left[\sum_{t=\tau_1+1}^{\tau_2} \gamma^{t-1} X_{i,t} + \gamma^{\tau_2}\frac{\zeta(\hat{\theta}_i)}{1-\gamma} \,\middle|\, \theta_i = \hat{\theta}_i\right] &= \mu(\hat{\theta}_i)\mathbb{E}\left[\sum_{t=\tau_1+1}^{\tau_2} \gamma^{t-1} \,\middle|\, \theta_i = \hat{\theta}_i\right] \\ &\quad + \mathbb{E}\left[\frac{\gamma^{\tau_2}}{1-\gamma} \,\middle|\, \theta_i = \hat{\theta}_i\right]\zeta(\hat{\theta}_i) \\ &\leq \mathbb{E}\left[\gamma^{\tau_1} \mid \theta_i = \hat{\theta}_i\right]\frac{\max(\zeta(\hat{\theta}_i), \mu(\hat{\theta}_i))}{1-\gamma}.\end{aligned}$$

Thus we conclude, because $\hat{\theta}_i$ was arbitrary, that almost surely,

$$\mathbb{E}\left[\sum_{t=\tau_1+1}^{\tau_2} \gamma^{t-1} X_{i,t} + \gamma^{\tau_2}\frac{\zeta(\hat{\theta}_i)}{1-\gamma} \,\middle|\, \theta_i\right] \leq \mathbb{E}\left[\gamma^{\tau_1} \mid \theta_i\right]\frac{\max(\zeta(\hat{\theta}_i), \mu(\theta_i))}{1-\gamma}. \tag{A.11}$$

Let $\tau^\star$ be a stopping time that achieves the supremum in $V_\gamma^M(y, \lambda)$ and define the predictable stopping time $\tau_K^\star \triangleq K \wedge \tau^\star$. Consider the (conditional) cumulative rewards in the definition of $V_\gamma^M(y)$, from time $\tau_K^\star + 1$ onwards, given the sufficient statistic observed at time $\tau_K^\star$. That is,

$$\mathbb{E}\left[\sum_{t=\tau_K^\star+1}^{\tau^\star} \gamma^{t-1} X_{i,t} + \gamma^{\tau^\star} R_{\lambda,M}(\tau^\star, y_{i,\tau^\star-1})/(1-\gamma) \,\middle|\, y_{i,\tau_K^\star-1}\right].$$

We upper bound this random variable as follows. First, we note that, at any time s and for any statistic $\hat{y} \in \mathcal{Y}$, the following statement holds

$$\mathbb{P}\left(R(\hat{y}) \leq r\right) = \mathbb{P}\left(\mu(\theta_i) \leq r \mid y_{i,s} = \hat{y}\right), \qquad \forall r \in \mathbb{R} \tag{A.12}$$

meaning that the posterior distribution of the arm's expected reward $R(y_{i,s})$ is the same as $\mu(\theta_i)$ *conditioned* on having observed statistic $\hat{y}$ about the arm. This holds

by definition of the random variable $R(y)$. Because of this observation, we have that the following inequality holds almost surely,

$$
\begin{aligned}
&\gamma^{\tau^\star} \frac{R_{\lambda,M}(\tau^\star, y_{i,\tau^\star-1})}{1-\gamma} \\
&\quad = \gamma^{\tau^\star} \left(\mathbb{1}\left(\tau^\star = M\right) \frac{\max(\lambda, R(y_{i,\tau^\star-1}))}{1-\gamma} + \mathbb{1}\left(\tau^\star < M\right) \frac{\lambda}{1-\gamma} \right) \\
&\quad = \mathbb{1}\left(\tau^\star = M\right) \gamma^M \frac{\max(\lambda, R(y_{i,M}))}{1-\gamma} + \mathbb{1}\left(\tau^\star < M\right) \gamma^{\tau^\star} \frac{\lambda}{1-\gamma} \\
&\quad \overset{(*)}{=} \mathbb{1}\left(\tau^\star = M\right) \gamma^M \frac{\mathbb{E}\left[\max(\lambda, R(y_{i,M})) \mid y_{i,M}\right]}{1-\gamma} + \mathbb{1}\left(\tau^\star < M\right) \gamma^{\tau^\star} \frac{\lambda}{1-\gamma} \\
&\quad \overset{(\dagger)}{=} \mathbb{1}\left(\tau^\star = M\right) \gamma^M \frac{\mathbb{E}\left[\max(\lambda, \mu(\theta_i)) \mid y_{i,M}\right]}{1-\gamma} + \mathbb{1}\left(\tau^\star < M\right) \gamma^{\tau^\star} \frac{\lambda}{1-\gamma} \\
&\quad \overset{(**)}{\leq} \frac{\mathbb{E}\left[\gamma^{\tau^\star} \max(\lambda, \mu(\theta_i)) \mid y_{i,\tau^\star-1}\right]}{1-\gamma}
\end{aligned}
$$

where $(*)$ and $(**)$ both use the fact that for any t, $\tau^\star \leq t$ is measurable with respect to the σ-algebra generated by $y_{i,t-1}$, namely $\mathcal{F}_{t-1}$. Equation $(\dagger)$ follows from (A.12). Therefore, immediately using the above inequality and conditioning on the event

$\tau^\star > K$, we have that

$$\begin{aligned}
&\mathbb{E}\left[\sum_{t=\tau_K^\star+1}^{\tau^\star} \gamma^{t-1} X_{i,t} + \gamma^{\tau^\star} \frac{R_{\lambda,M}(\tau^\star, y_{i,\tau^\star-1})}{1-\gamma} \,\middle|\, \tau^\star > K,\ y_{i,\tau_K^\star-1}\right] \\
&\quad \le \mathbb{E}\left[\sum_{t=\tau_K^\star+1}^{\tau^\star} \gamma^{t-1} X_{i,t} + \mathbb{E}\left[\gamma^{\tau^\star} \frac{\max(\lambda, \mu(\theta_i)))}{1-\gamma} \,\middle|\, y_{i,\tau^\star-1}\right] \,\middle|\, \tau^\star > K,\ y_{i,\tau_K^\star-1}\right] \\
&\quad = \mathbb{E}\left[\sum_{t=\tau_K^\star+1}^{\tau^\star} \gamma^{t-1} X_{i,t} + \gamma^{\tau^\star} \frac{\max(\lambda, \mu(\theta_i))}{1-\gamma} \,\middle|\, \tau^\star > K,\ y_{i,\tau_K^\star-1}\right] && \text{(A.13)} \\
&\quad = \mathbb{E}\left[\mathbb{E}\left[\sum_{t=K+1}^{\tau^\star} \gamma^{t-1} X_{i,t} + \gamma^{\tau^\star} \frac{\max(\lambda, \mu(\theta_i))}{1-\gamma} \,\middle|\, \theta_i\right] \,\middle|\, \tau^\star > K,\ y_{i,\tau_K^\star-1}\right] && \text{(A.14)} \\
&\quad \le \mathbb{E}\left[\gamma^{\tau_K^\star} \frac{\max(\mu(\theta_i), \lambda)}{1-\gamma} \,\middle|\, \tau^\star > K,\ y_{i,\tau_K^\star-1}\right] && \text{(A.15)} \\
&\quad = \mathbb{E}\left[\gamma^{\tau_K^\star} \frac{\max(R(y_{i,\tau_K^\star-1}), \lambda)}{1-\gamma} \,\middle|\, \tau^\star > K,\ y_{i,\tau_K^\star-1}\right] && \text{(A.16)} \\
&\quad = \mathbb{E}\left[\frac{\gamma^{\tau_K^\star} R_{\lambda,K}(\tau_K^\star, y_{i,\tau_K^\star-1})}{1-\gamma} \,\middle|\, \tau^\star > K,\ y_{i,\tau_K^\star-1}\right] && \text{(A.17)}
\end{aligned}$$

where (A.13), (A.14) use the tower property and (A.15) follows from the bound in (A.11) because $\tau_K^\star < \tau^\star$, almost surely. Equation (A.16) follows from statement (A.12) and that the event $\tau^\star > K$ is $\mathcal{F}_{K-1}$-measurable (we can decide whether to pull arm i or retire based on information up to and including time $K-1$). Finally equation (A.17) is derived by substituting in the definition of $R_{\lambda,K}$ (as given in Section 2.2) and noting that $\tau_K^\star = K$ under the above conditioning.

We now condition on the complement of the previous event we considered, namely, $\tau^\star \le K$. Under that event, $\tau^\star$ occurred early enough before time $K+1$ and thus

$\tau_K^\star = \tau^\star$. Therefore, it follows from this observation that

$$\begin{aligned}
&\mathbb{E}\left[\sum_{t=\tau_K^\star+1}^{\tau^\star} \gamma^{t-1} X_{i,t} + \gamma^{\tau^\star} \frac{R_{\lambda,M}(\tau^\star, y_{i,\tau^\star-1})}{1-\gamma} \,\middle|\, \tau^\star \le K,\ y_{i,\tau_K^\star-1}\right] \\
&\quad = \mathbb{E}\left[\gamma^{\tau^\star} \frac{\lambda}{1-\gamma} \,\middle|\, \tau^\star \le K,\ y_{i,\tau_K^\star-1}\right] \\
&\quad \le \mathbb{E}\left[\gamma^{\tau_K^\star} \frac{R_{\lambda,K}(\tau_K^\star, y_{i,\tau_K^\star-1})}{1-\gamma} \,\middle|\, \tau^\star \le K,\ y_{i,\tau_K^\star-1}\right] \qquad \text{(A.18)}
\end{aligned}$$

where (A.18) is obtained by noting that $R_{\lambda,K}(\tau, y) \ge \lambda$ for any choice of τ, K and y. Thus, by the law of total expectation and (A.17), (A.18), we establish that

$$\begin{aligned}
&\mathbb{E}\left[\sum_{t=\tau_K^\star+1}^{\tau^\star} \gamma^{t-1} X_{i,t} + \gamma^{\tau^\star} \frac{R_{\lambda,M}(\tau^\star, y_{i,\tau^\star-1})}{1-\gamma} \,\middle|\, y_{i,\tau_K^\star-1}\right] \\
&\qquad\qquad \le \mathbb{E}\left[\gamma^{\tau_K^\star} \frac{R_{\lambda,K}(\tau_K^\star, y_{i,\tau_K^\star-1})}{1-\gamma} \,\middle|\, y_{i,\tau_K^\star-1}\right]. \qquad \text{(A.19)}
\end{aligned}$$

We are ready to complete our main argument in this proof by using the above bound and 'breaking up' the $V_\gamma^M(y, \lambda)$ into rewards from times before $\tau_K^\star$ and after (and

bounding the latter terms). More precisely, we obtain that

$$V_\gamma^M(y,\lambda) = \mathbb{E}_y\left[\sum_{t=1}^{\tau^\star}\gamma^{t-1}X_{i,t} + \gamma^{\tau^\star}\frac{R_{\lambda,M}(\tau^\star, y_{i,\tau^\star-1})}{1-\gamma}\right] \tag{A.20}$$

$$= \mathbb{E}_y\left[\sum_{t=1}^{\tau_K^\star}\gamma^{t-1}X_{i,t} + \sum_{t'=\tau_K^\star+1}^{\tau^\star}\gamma^{t'-1}X_{i,t'} + \gamma^{\tau^\star}\frac{R_{\lambda,M}(\tau^\star, y_{i,\tau^\star-1})}{1-\gamma}\right]$$

$$= \mathbb{E}_y\left[\sum_{t=1}^{\tau_K^\star}\gamma^{t-1}X_{i,t} + \mathbb{E}\left[\sum_{t'=\tau_K^\star+1}^{\tau^\star}\gamma^{t'-1}X_{i,t'} + \gamma^{\tau^\star}\frac{R_{\lambda,M}(\tau^\star, y_{i,\tau^\star-1})}{1-\gamma}\,\middle|\, y_{i,\tau_K^\star-1}\right]\right] \tag{A.21}$$

$$\leq \mathbb{E}_y\left[\sum_{t=1}^{\tau_K^\star}\gamma^{t-1}X_{i,t} + \mathbb{E}\left[\gamma^{\tau_K^\star}\frac{R_{\lambda,K}(\tau_K^\star, y_{i,\tau_K^\star-1})}{1-\gamma}\,\middle|\, y_{i,\tau_K^\star-1}\right]\right] \tag{A.22}$$

$$= \mathbb{E}_y\left[\sum_{t=1}^{\tau_K^\star}\gamma^{t-1}X_{i,t} + \gamma^{\tau_K^\star}\frac{R_{\lambda,K}(\tau_K^\star, y_{i,\tau_K^\star-1})}{1-\gamma}\right] \tag{A.23}$$

$$\leq \sup_{1\leq\tau\leq K}\mathbb{E}_y\left[\sum_{t=1}^{\tau}\gamma^{t-1}X_{i,t} + \gamma^{\tau}\frac{R_{\lambda,K}(\tau, y_{i,\tau-1})}{1-\gamma}\right]$$

$$= V_\gamma^K(y,\lambda) \tag{A.24}$$

where Equations (A.21), (A.23) use the tower property and (A.22) is immediately derived by using the bound of (A.19). Finally, we note that an almost identical proof can be given to show that $V_\gamma^K(y,\lambda) \geq V_\gamma(y,\lambda)$ where the lower bound is the continuation value used to compute the Gittins index.

We have shown that for any λ and y, that $V_\gamma^K(y,\lambda)$ is non-increasing in K, and that $V_\gamma(y,\lambda)$ is a lower bound to this sequence. We make use of these facts to now prove that $v_\gamma^K(y)$ is also non-increasing in K. To this end, let us suppose for contradiction that there exist two integers $K_1 \leq K_2$ and $v_\gamma^{K_1}(y) < v_\gamma^{K_2}(y)$. From Lemma 16 we know that

$$V_\gamma^{K_2}(y, v_\gamma^K(y)) > v_\gamma^K(y)/(1-\gamma) = V_\gamma^K(y, v_\gamma^K(y)), \tag{A.25}$$

which contradicts the claim just shown. Therefore, $v_\gamma^K(y)$ must also be a non-increasing sequence in K. The same argument can be used to further show that

$v_\gamma^K(y) \geq v_\gamma(y)$.

We now turn our attention to proving the convergence property stated in the Lemma. The first step will be to prove that for all $y \in \mathcal{Y}$ and $\lambda \in \mathbb{R}_+$, that

$$\lim_{K\to\infty} V_\gamma^K(y,\lambda) = V_\gamma(y,\lambda). \tag{A.26}$$

Indeed, we upper bound the optimistic continuation value for a fixed parameter M as follows:

$$\begin{aligned}
V_\gamma^M(y,\lambda) &= \sup_{1\leq\tau\leq M} \mathbb{E}_y\left[\sum_{t=1}^{\tau}\gamma^{t-1}X_{i,t} + \frac{\gamma^\tau R_{\lambda,M}(\tau, y_{i,\tau-1})}{1-\gamma}\right] \\
&= \sup_{1\leq\tau\leq M} \mathbb{E}_y\left[\sum_{t=1}^{\tau}\gamma^{t-1}X_{i,t} + \frac{\gamma^\tau\lambda}{1-\gamma} + \frac{\gamma^\tau R_{\lambda,M}(\tau, y_{i,\tau-1})}{1-\gamma} - \frac{\gamma^\tau\lambda}{1-\gamma}\right] \\
&\leq \sup_{\tau\geq 1} \mathbb{E}_y\left[\sum_{t=1}^{\tau}\gamma^{t-1}X_{i,t} + \frac{\gamma^\tau\lambda}{1-\gamma}\right] + \sup_{1\leq\tau\leq M} \mathbb{E}_y\left[\frac{\gamma^\tau R_{\lambda,M}(\tau, y_{i,\tau-1})}{1-\gamma} - \frac{\gamma^{\tau-1}\lambda}{1-\gamma}\right] \\
&= V_\gamma(y,\lambda) + \sup_{1\leq\tau\leq M} \mathbb{E}_y\left[\frac{\gamma^\tau[R_{\lambda,M}(\tau, y_{i,\tau-1}) - \lambda]}{1-\gamma}\right] \\
&\leq V_\gamma(y,\lambda) + \gamma^M \mathbb{E}_y\left[\frac{R_{\lambda,M}(M, y_{i,M-1}) - \lambda}{1-\gamma}\right] \\
&= V_\gamma(y,\lambda) + \gamma^M \mathbb{E}_y\left[\frac{(R(y_{i,M-1}) - \lambda)^+}{1-\gamma}\right] \\
&\leq V_\gamma(y,\lambda) + \gamma^M \mathbb{E}_y\left[\frac{|R(y_{i,M-1})|}{1-\gamma}\right] \\
&= V_\gamma(y,\lambda) + \gamma^M \mathbb{E}_y\left[\frac{|\mu(\theta_i)|}{1-\gamma}\right], \qquad \text{(A.27)}
\end{aligned}$$

where equation (A.27) follows from the definition of the random variable $R(.)$ and the law of iterated expectation. Now because $0 < \gamma < 1$ and $\mathbb{E}_y\left[|\mu(\theta_i)|\right] < \infty$, the right hand side above converges to $V_\gamma(y,\lambda)$. Finally, notice that $V_\gamma^M(y,\lambda) \geq V_\gamma(y,\lambda)$, and from this equation (A.26) follows. To finish the proof, we note that $V_\gamma^K(y,\lambda)$ is continuous in λ. Therefore, if we fix ϵ, there is an integer $K = K(\epsilon)$ that is large

enough so that

$$\begin{aligned}\left|v_\gamma^K(y) - v_\gamma(y)\right| &= \left|V_\gamma^K(y, v_\gamma^K(y)) - V_\gamma(y, v_\gamma(y))\right| \\ &\leq \left|V_\gamma^K(y, v_\gamma(y)) - V_\gamma(y, v_\gamma(y)) + \epsilon\right| \\ &\leq \left|V_\gamma^K(y, v_\gamma(y)) - V_\gamma(y, v_\gamma(y))\right| + \epsilon \\ &\leq 2\epsilon.\end{aligned}$$

Then, we take the limit as $\epsilon \downarrow 0$ and the Lemma is shown. □

The next Lemma will be the final property of the function V_γ^K that we prove. This will subsequently be used in the proof of Lemma 3.

Lemma 17. *Let i be any arm. For any look-ahead parameter $K \in \mathbb{Z}_+$, discount factor γ and any constant η, we have*

$$\mathbb{E}_y\left[V_\gamma^K(y_{i,1}, \eta)\right] \geq V_\gamma^K(y, \eta)$$

where we recall that $y_{i,1}$ is the summary statistic corresponding to the posterior obtained from pulling arm i once.

Proof. For any $\hat{y} \in \mathcal{Y}$, let $\tau^\star(\hat{y})$ be the (predictable) optimal stopping time for the problem (involving computing V_γ^K) whose initial state is $y_{i,0} = \hat{y}$. With this notation in hand, we conclude that

$$\begin{aligned}\mathbb{E}_y\left[V_\gamma^K(y_{i,1}, \eta)\right] &= \mathbb{E}_y\left[\mathbb{E}_{y_{i,1}}\left[\sum_{s=1}^{\tau^\star(y_{i,1})} \gamma^{s-1} X_{i,s} + \frac{\gamma^{\tau^\star(y_{i,1})} R_{\eta,K}(\tau, y_{i,\tau^\star(y_{i,1})-1})}{1-\gamma}\right]\right] && \text{(A.28)} \\ &\geq \mathbb{E}_y\left[\mathbb{E}_{y_{i,2}}\left[\sum_{s=1}^{\tau^\star(y)} \gamma^{s-1} X_{i,s} + \frac{\gamma^{\tau^\star(y)} R_{\eta,K}(\tau, y_{i,\tau^\star(y)-1})}{1-\gamma}\right]\right] && \text{(A.29)} \\ &= \mathbb{E}_y\left[\sum_{s=1}^{\tau^\star(y)} \gamma^{s-1} X_{i,s} + \frac{\gamma^{\tau^\star(y)} R_{\eta,K}(\tau, y_{i,\tau^\star(y)-1})}{1-\gamma}\right] && \text{(A.30)} \\ &= V_\gamma^K(y, \eta)\end{aligned}$$

where (A.28), (A.30) both follow from the tower property and (A.29) is due to the sub-optimality of the stopping rule $\tau^\star(y)$ when the actual starting state is $y_{i,1}$. Intuitively, we lose out revenue by throwing away information about the arm. □

A.4 Results for the frequentist regret bound

This section contains proofs of results required to show Theorem 1. It is helpful to go over the definitions and some general properties of the Optimistic Gittins index given in Section A.3 when reading this.

A.4.1 Definitions and properties of Binomial distributions.

We list notation and facts related to Beta and Binomial distributions, which are used through this section.

Definition 5. *$F^B_{n,p}(.)$ is the CDF of the Binomial distribution with parameters n and p, and $F^\beta_{a,b}(.)$ is the CDF of the Beta distribution with parameters a and b.*

Lemma 18. *Let a and b be positive integers and $y \in [0,1]$,*

$$F^\beta_{a,b}(y) = 1 - F^B_{a+b-1,y}(a-1)$$

Proof. Proof is found in Agrawal and Goyal [2012]. □

Lemma 19. *The median of a Binomial(n,p) distribution is either $\lceil np \rceil$ or $\lfloor np \rfloor$.*

Proof. A proof of this fact can be found in Jogdeo and Samuels [1968]. □

Corollary 1 (Corollary of Fact 19)**.** *Let n be a positive integer and $p \in (0,1)$. For any non-negative integer $s < np$*

$$F^B_{n,p}(s) \leq 1/2$$

Lemma 20. *Let n be a positive integer and $p \in [0,1]$. Then for any $k \in \{0,\ldots,n\}$,*

$$(1-p)F^B_{n-1,p}(k) \leq F^B_{n,p}(k) \leq F^B_{n-1,p}(k)$$

Proof. To prove $F^B_{n,p}(k) \leq F^B_{n-1,p}(k)$, we let $X_1, \ldots, X_n$ be i.i.d samples from a Bernoulli(p) distribution. We then have

$$F^B_{n,p}(k) = \mathbb{P}\left(\sum_{i=1}^{n} X_i \leq k\right) \leq \mathbb{P}\left(\sum_{i=1}^{n-1} X_i \leq k\right) = F^B_{n-1,p}(k)$$

Now to prove $(1-p)F^B_{n-1,p}(k) \leq F^B_{n,p}(k)$, it's enough to observe that $F^B_{n,p}(k) = pF^B_{n-1,p}(k-1) + (1-p)F^B_{n-1,p}(k)$. □

Ratio of Binomial CDFs.

Lemma 21. *Let $0 < q < p < 1$. Let n be a positive integer such that $e^{\frac{n}{2}d(q,p)} \geq (n+1)^4$ and let k be a non-negative integer such that $k < nq$. It then follows that*

$$F^B_{n,q}(k)/F^B_{n,p}(k) > e^{\frac{n}{2}d(q,p)}.$$

Proof. From the method of types (see Cover and Thomas [2012]), we have for any $r \in (0,1)$ and $j < nr$

$$\frac{e^{-nd(j/n,r)}}{(1+n)^2} \leq F^B_{n,r}(j) \leq (n+1)^2 e^{-nd(j/n,r)}. \tag{A.31}$$

Because $k < nq < np$, by applying (A.31) to both the numerator and denominator, we get

$$\frac{F^B_{n,q}(k)}{F^B_{n,p}(k)} \geq \frac{e^{-nd(k/n,q)}}{(n+1)^4 e^{-nd(k/n,p)}} = \frac{e^{n(d(k/n,p)-d(k/n,q))}}{(n+1)^4}.$$

Examining the exponent, we find

$$\begin{aligned} d(k/n,p) - d(k/n,q) &= \frac{k}{n}\log\frac{q}{p} + \left(1-\frac{k}{n}\right)\log\frac{1-q}{1-p} \\ &> q\log\frac{q}{p} + (1-q)\log\frac{1-q}{1-p} \\ &= d(q,p) \end{aligned}$$

where the bound holds because the expression is decreasing in k, and $k < nq$. Therefore,

$$\frac{F^B_{n,q}(k)}{F^B_{n,p}(k)} > \frac{e^{nd(q,p)}}{(n+1)^4} = \frac{e^{\frac{n}{2}d(q,p)}}{(n+1)^4} e^{\frac{n}{2}d(q,p)} \geq e^{\frac{n}{2}d(q,p)}. \tag{A.32}$$

The final lower bound in (A.32) follows from the assumption on n. □

A.4.2 Proof of Lemma 3

Proof. The proof hinges on showing that for any K, which is the number of look-ahead steps used to compute the Optimistic Gittins index, that

$$\mathbb{P}\left(v^K_{1,t} < \eta\right) = O\left(\frac{1}{t^{1+h_\eta}}\right) \tag{A.33}$$

where $h_\eta > 0$ is some constant that depends on η. After showing the above statement, the result would follow due to convergence of the series $\sum_{t=1}^{\infty} \mathbb{P}\left(v^K_{1,t} < \eta\right)$. The first step will be to show that for any $K \geq 1$ and any $\zeta \geq 0$, that there exists $h'_\eta > 0$, such that

$$\mathbb{P}\left((1-\gamma_t)V^K_{\gamma_t}(y_{1,P_1(t)}, \eta) < \eta + \zeta/t\right) = O_{\eta,\zeta}\left(\frac{1}{t^{1+h'_\eta}}\right), \tag{A.34}$$

where $V^K_{\gamma_t}$ is the continuation value defined in Section A.3 and $O_{\eta,\zeta}$ means that the constant in front the big-Oh depends on both ζ and η. After showing the above claim, Lemma 16 would imply Equation (A.33) because we know from that result that,

$$\begin{aligned}\mathbb{P}\left(v^K_{1,t} < \eta\right) &= \mathbb{P}\left((1-\gamma_t)V^K_{\gamma_t}(y_{1,P_1(t)}, \eta) < \eta\right) \\ &= O\left(\frac{1}{t^{1+h_\eta}}\right)\end{aligned}$$

for some $h_\eta > 0$. The second equation above is just a special case of (A.34) when $\zeta = 0$.

Ultimately, showing equation (A.34), and thus proving the Lemma, is an induction over the parameter K and we begin with the base case, which requires some work using properties of the Beta and Binomial distributions.

Proof of the base case

Let us fix $\zeta \geq 0$. We prove that when the algorithm uses a look-ahead parameter of $K = 1$, that there exists a positive constant h_η such that

$$\mathbb{P}\left((1-\gamma_t)V^1_{\gamma_t}(y_{1,P_1(t)},\eta) < \eta + \zeta/t\right) = O_{\eta,\zeta}\left(\frac{1}{t^{1+h_\eta}}\right). \tag{A.35}$$

First, we define $\delta := (\theta_1 - \eta)/2$ and $\eta' := \eta + \delta$. In other words, δ is half the distance between η and θ_1; η' is the point half-way. Recall that $P_i(t)$ refers to the counting process for the number of pulls of arm i up to *but not including* time t and that $S_i(t)$ is the corresponding total reward (or number of successes from all the Bernoulli trials). Showing this base case consists of showing two claims:

Claim 1: $\{(1-\gamma_t)V^1_{\gamma_t}(y_{1,P_1(t)},\eta) < \eta + \zeta/t\} \subseteq \left\{F^B_{P_1(t)+1,\eta'}(S_1(t)) < \frac{\zeta+1}{\delta t}\right\}$

Let $V_t \sim$Beta$(S_1(t)+1, P_1(t) - S_1(t) + 1)$ be the agent's posterior on the expected reward from the optimal arm (notice that $y_{1,P_1(t)} = (S_1(t)+1, P_1(t) - S_1(t)+1)$ in this case). Using the simplified equation for the continuation value when $K = 1$, namely $V^1_{\gamma_t}$ (see Equation (2.3)),

$$(1-\gamma_t)V^1_{\gamma_t}\left((S_1(t)+1, P_1(t) - S_1(t)+1),\eta\right) = \mathbb{E}\left[V_t\right] + \gamma_t \mathbb{E}\left[(\eta - V_t)^+\right],$$

we find that

$$\begin{aligned}
\left\{(1-\gamma_t)V^1_{\gamma_t}(y_{1,N_1(t)},\eta) < \eta + \frac{\zeta}{t}\right\} &= \left\{\mathbb{E}\left[V_t\right] + \gamma_t\mathbb{E}\left[(\eta - V_t)^+\right] < \eta + \frac{\zeta}{t}\right\} \\
&= \left\{(1-1/t)\mathbb{E}\left[(\eta - V_t)^+\right] < \mathbb{E}\left[\eta - V_t\right] + \frac{\zeta}{t}\right\} \qquad \text{(A.36)} \\
&= \left\{\mathbb{E}\left[(V_t - \eta)^+\right] < \frac{1}{t}\mathbb{E}\left[(\eta - V_t)^+\right] + \frac{\zeta}{t}\right\} \\
&\subseteq \left\{\mathbb{E}\left[(V_t - \eta)^+\right] < \frac{\zeta+1}{t}\right\} \qquad \text{(A.37)}
\end{aligned}$$

where (A.36) follows from the definition of γ_t and (A.37) is due to V_t, η both lying in the interval $[0,1]$. We approximate the conditional expectation in (A.37) with the following bound:

$$\begin{aligned}
\mathbb{E}\left[(V_t-\eta)^+\right] &= \mathbb{E}\left[(V_t-\eta)\mathbb{1}\left(V_t \geq \eta\right)\right] \\
&= \mathbb{E}\left[(V_t-\eta)\mathbb{1}\left(\eta+\delta > V_t \geq \eta\right)\right] \\
&\quad + \mathbb{E}\left[(V_t-\eta)\mathbb{1}\left(V_t \geq \eta+\delta\right)\right] \\
&> \mathbb{E}\left[(V_t-\eta)\mathbb{1}\left(V_t \geq \eta+\delta\right)\right] \\
&\geq \delta\mathbb{P}\left(V_t \geq \eta'\right) \\
&= \delta(1-F^{\beta}_{S_1(t)+1,P_1(t)-S_1(t)+1}(\eta')) \\
&= \delta F^{B}_{P_1(t)+1,\eta'}(S_1(t)) \qquad \text{(A.38)}
\end{aligned}$$

where the final equality is due to Fact 18. The claim then follows from the above bound and (A.37). We proceed with the second part of the base case's proof:

Claim 2: $\mathbb{P}\left(F^{B}_{P_1(t)+1,\eta'}(S_1(t)) < \frac{\zeta+1}{\delta t}\right) = O\left(\frac{1}{t^{1+h_\eta}}\right)$ **for some** $h_\eta > 0$

Let us fix the sequence $f_t \triangleq -\frac{\log(\delta t/(\zeta+1))}{\log(1-\eta')} - 1 = O(\log t)$. We then have by a straightforward decomposition that

$$\begin{aligned}
\mathbb{P}\left(F^{B}_{P_1(t)+1,\eta'}(S_1(t)) < \frac{\zeta+1}{\delta t}\right) &= \mathbb{P}\left(F^{B}_{P_1(t)+1,\eta'}(S_1(t)) < \frac{\zeta+1}{\delta t},\ P_1(t) > f_t\right) \\
&\quad + \mathbb{P}\left(F^{B}_{P_1(t)+1,\eta'}(S_1(t)) < \frac{\zeta+1}{\delta t},\ P_1(t) \leq f_t\right). \qquad \text{(A.39)}
\end{aligned}$$

Then notice that for the second term in the RHS of (A.39) we have the following bound,

$$\begin{aligned}
\mathbb{P}\left(F^B_{P_1(t)+1,\eta'}(S_1(t)) < \frac{\zeta+1}{\delta t},\ P_1(t) \le f_t\right) &\le \mathbb{P}\left(F^B_{P_1(t)+1,\eta'}(0) < \frac{\zeta+1}{\delta t},\ P_1(t) \le f_t\right) \\
&= \mathbb{P}\left((1-\eta')^{P_1(t)+1} < \frac{\zeta+1}{\delta t},\ P_1(t) \le f_t\right) \\
&\le \mathbb{P}\left((1-\eta')^{f_t+1} < \frac{\zeta+1}{\delta t}\right) \\
&= 0. \qquad \text{(A.40)}
\end{aligned}$$

Now we use the following fact to correspondingly bound the left term on the RHS of (A.39). Define the function

$$F^{-B}_{n,p}(u) := \inf\{x : F^B_{n,p}(x) \ge u\}$$

which is the inverse CDF. Then it is known that if $U \sim \text{Uniform}(0,1)$, then $F^{-B}_{n,p}(U) \sim \text{Binomial}(n,p)$. Furthermore, the event $F^B_{n,p}(F^{-B}_{n,p}(U)) \ge U$ occurs with probability 1 due to the definition of the inverse CDF.

Now let us only consider large t, in particular $t > M = M(\theta_1, \eta')$ where:

1. M is such that $e^{d(\eta',\theta_1)f_M/2} > (f_M+1)^4$ (we need this condition when we use Lemma 21)

2. $M > \frac{4(\zeta+1)}{(1-\eta')\delta}$

3. $\lceil f_M \rceil > 0$ and $F^B_{t',\eta'}(t'\eta') > 1/4$ for all $t' > \lceil f_M \rceil$. Note that there is a large enough integer for this because $F^B_{\lceil f_t \rceil,\eta'}(f_t\eta') \to \frac{1}{2}$ as $t \to \infty$.

Suppose that $t > M$. It then follows that the event

$$\left\{F^B_{P_1(t),\eta'}(S_1(t)) < \frac{\zeta+1}{(1-\eta')\delta t},\ S_1(t) \ge P_1(t)\eta',\ P_1(t) > f_t\right\}$$

has measure zero because of the assumptions made on M. Therefore if $t > M$, we

have

$$
\begin{aligned}
\mathbb{P}\Bigg(F^{B}_{P_1(t)+1,\eta'}(S_1(t)) < \frac{\zeta+1}{\delta t},\ P_1(t) > f_t \Bigg) & \\
&\leq \mathbb{P}\left(F^{B}_{P_1(t),\eta'}(S_1(t)) < \frac{\zeta+1}{(1-\eta')\delta t},\ P_1(t) > f_t \right) && \text{(A.41)} \\
&= \mathbb{P}\left(F^{B}_{P_1(t),\eta'}(S_1(t)) < \frac{\zeta+1}{(1-\eta')\delta t},\ S_1(t) < P_1(t)\eta',\ P_1(t) > f_t \right) \\
&\leq \mathbb{P}\left(F^{B}_{P_1(t),\theta_1}(S_1(t))e^{P_1(t)D} < \frac{\zeta+1}{(1-\eta')\delta t},\ P_1(t) > f_t \right) && \text{(A.42)} \\
&\leq \mathbb{P}\left(F^{B}_{P_1(t),\theta_1}(S_1(t))e^{f_t D} < \frac{\zeta+1}{(1-\eta')\delta t} \right) \\
&= \mathbb{P}\left(F^{B}_{P_1(t),\theta_1}(F^{-B}_{P_1(t),\theta_1}(U)) < \frac{e^{-f_t D}(\zeta+1)}{(1-\eta')\delta t} \right) && \text{(A.43)} \\
&\leq \mathbb{P}\left(U < \frac{e^{-f_t D}(\zeta+1)}{(1-\eta')\delta t} \right) \\
&= \frac{e^{-f_t D}(\zeta+1)}{(1-\eta')\delta t} \\
&= \mathcal{O}_{\eta,\zeta}\left(\frac{1}{t^{1+Dc_{\eta'}}} \right) && \text{(A.44)}
\end{aligned}
$$

where $D = d(\eta', \theta_1) > 0$ and $c_{\eta'} = -\log^{-1}(1-\eta') > 0$ are constant. The bound (A.41) holds due to Fact (20). Bound (A.42) follows from an application of Lemma 21 and the fact that $t > M$. Equation (A.43) follows from $S_1(t) \sim \text{Binomial}(P_1(t), \theta_1)$ and the inverse sampling technique. By combining bounds (A.44), (A.40) and (A.39), we finally obtain the result for the base case by taking $h_\eta = Dc_{\eta'}$.

Proof of the inductive step

Now, suppose that for $K - 1 \geq 1$ and any $\zeta \geq 0$, the following induction hypothesis holds

$$
\mathbb{P}\left((1-\gamma_t) V^{K-1}_{\gamma_t}(y_{1,P_1(t)}, \eta) < \eta + \frac{\zeta}{t} \right) = O_{\eta,\zeta}\left(\frac{1}{t^{1+h_\eta}} \right)
$$

for some $h_\eta > 0$. We prove the same result for the next integer K. Observe that when t is large enough, using the Bellman equation for V_γ^K, we have

$$\begin{aligned}
&\mathbb{P}\left((1-\gamma_t)V_{\gamma_t}^K(y_{1,P_1(t)},\eta) < \eta + \frac{\zeta}{t}\right) \\
&\quad= \mathbb{P}\Big((1-\gamma_t)\mathbb{E}\left[X_{1,t} \mid y_{1,P_1(t)}\right] \\
&\qquad\qquad + \gamma_t\mathbb{E}\left[\max(\eta,(1-\gamma_t)V_{\gamma_t}^{K-1}(y_{1,P_1(t)+1},\eta)) \mid y_{1,P_1(t)}\right] < \eta + \frac{\zeta}{t}\Big) \quad \text{(A.45)} \\
&\quad\le \mathbb{P}\left(\left(1-\frac{1}{t}\right)\mathbb{E}\left[(1-\gamma_t)V_{\gamma_t}^{K-1}(y_{1,P_1(t)+1},\eta) \mid y_{1,P_1(t)}\right] < \eta + \frac{\zeta}{t}\right) \\
&\quad\le \mathbb{P}\left(\left(1-\frac{1}{t}\right)(1-\gamma_t)V_{\gamma_t}^{K-1}(y_{1,P_1(t)},\eta) < \eta + \frac{\zeta}{t}\right) \quad \text{(A.46)} \\
&\quad\le \mathbb{P}\left((1-\gamma_t)V_{\gamma_t}^{K-1}(y_{1,P_1(t)},\eta) < \frac{t}{t-1}\left(\eta + \frac{\zeta}{t}\right)\right) \\
&\quad\le \mathbb{P}\left((1-\gamma_t)V_{\gamma_t}^{K-1}(y_{1,P_1(t)},\eta) < \eta + \frac{\eta}{t-1} + \frac{\zeta}{t-1}\right) \\
&\quad\le \mathbb{P}\left((1-\gamma_t)V_{\gamma_t}^{K-1}(y_{1,P_1(t)},\eta) < \eta + \frac{\zeta+1}{t}\right) \\
&\quad= O_{\eta,\zeta}\left(\frac{1}{t^{1+h_\eta}}\right) \quad \text{(A.47)}
\end{aligned}$$

where the final inequality holds when t is large enough because $\eta < 1$, equation (A.45) results from an expansion of Bellman's equation and bound (A.46) follows from Lemma 17. Finally, equation (A.47) follows from the induction hypothesis. □

A.4.3 Proof of Lemma 4

Proof. See the main proof of Theorem 1 to recall the definition of constants η_1, η_3 and their relationship with θ_2 and θ_1. As an abbreviation we let $L = L(T)$. Moreover, because for any arm i $v_{i,t}^K \le v_{i,t}^{K-1} \le \ldots \le v_{i,t}^1$ (Lemma 2), it will be sufficient to consider this proof only for $v_{2,t}^1$, which we also will abbreviate as $v_{2,t} \triangleq v_{2,t}^1$. Similarly, we will abbreviate the notation for the OGI policy as π^{OG} and suppress the parameter K.

Firstly, by the law of total probability and the definition of $P_i(t)$ in Section A.3,

we find that

$$\begin{aligned}
&\sum_{t=1}^{T} \mathbb{P}(v_{2,t} \geq \eta_3,\ N_2(t-1) \geq L,\ \pi_t^{\mathrm{OG}} = 2) \\
&\qquad = \sum_{t=1}^{T} \mathbb{P}\left(v_{2,t} \geq \eta_3,\ P_2(t) \geq L,\ S_2(t) < \lfloor P_2(t)\eta_1 \rfloor,\ \pi_t^{\mathrm{OG}} = 2\right) \\
&\qquad\qquad + \sum_{t=1}^{T} \mathbb{P}\left(v_{2,t} \geq \eta_3,\ P_2(t) \geq L,\ S_2(t) \geq \lfloor P_2(t)\eta_1 \rfloor,\ \pi_t^{\mathrm{OG}} = 2\right) \\
&\qquad \leq \sum_{t=1}^{T} \mathbb{P}\left(v_{2,t} \geq \eta_3,\ P_2(t) \geq L,\ S_2(t) < \lfloor P_2(t)\eta_1 \rfloor\right) \\
&\qquad\qquad + \sum_{t=1}^{T} \mathbb{P}\left(\pi_t^{\mathrm{OG}} = 2,\ S_2(t) \geq \lfloor P_2(t)\eta_1 \rfloor\right), \qquad \text{(A.48)}
\end{aligned}$$

where $S_2(t)$ is also defined in Section A.3 as the total reward from the second arm observed up to time $t-1$. Let $V_t \sim \mathrm{Beta}(S_2(t)+1, P_2(t)-S_2(t)+1)$ denote the agent's posterior on the second arm at time t, then

$$\begin{aligned}
&\sum_{t=1}^{T} \mathbb{P}(v_{2,t} \geq \eta_3,\ P_2(t) \geq L,\ S_2(t) < \lfloor P_2(t)\eta_1 \rfloor) \\
&\qquad = \sum_{t=1}^{T} \mathbb{P}\Big(\mathbb{E}\left[V_t\right] + \gamma_t \mathbb{E}\left[(\eta_3 - V_t)^+\right] \geq \eta_3, \\
&\qquad\qquad\qquad P_2(t) \geq L,\ S_2(t) < \lfloor P_2(t)\eta_1 \rfloor\Big) \\
&\qquad = \sum_{t=1}^{T} \mathbb{P}\left(\frac{\mathbb{E}\left[(\eta_3 - V_t)^+\right]}{\mathbb{E}\left[(V_t - \eta_3)^+\right]} \leq t,\ P_2(t) \geq L,\ S_2(t) < \lfloor P_2(t)\eta_1 \rfloor\right) \qquad \text{(A.49)}
\end{aligned}$$

where the first equality follows from Lemma 16 and the simplified form of the continuation value (defined in Section A.3) when $K = 1$. The following result lets us bound (A.49),

Lemma 22. *Let $0 < x < y < 1$. For any non-negative integers s and k with $s < \lfloor kx \rfloor$, it holds that*

$$\frac{\mathbb{E}\left[(y-V)^+\right]}{\mathbb{E}\left[(V-y)^+\right]} \geq \frac{(y-x)\exp(kd(x,y))}{2}$$

where $V \sim Beta(s+1, k-s+1)$.

Proof. See Appendix A.4.3. □

Therefore, from equation (A.49) and Lemma 22, we find that whenever $T > \left(\frac{2}{\eta_3-\eta_1}\right)^{1/\epsilon} =: T^*(\epsilon,\theta)$,

$$\begin{aligned}
&\sum_{t=1}^{T} \mathbb{P}(v_{2,t} \geq \eta_3,\ P_2(t) \geq L,\ S_2(t) < \lfloor P_2(t)\eta_1 \rfloor) \\
&\qquad \leq \sum_{t=1}^{T} \mathbb{P}\left((\eta_3-\eta_1)\exp\{P_2(t)d(\eta_1,\eta_3)\} \leq 2t,\ P_2(t) \geq L\right) \\
&\qquad \leq \sum_{t=1}^{T} \mathbb{P}\left((\eta_3-\eta_1)\exp\{Ld(\eta_1,\eta_3)\} \leq 2t\right) \\
&\qquad = \sum_{t=1}^{T} \mathbb{P}\left((\eta_3-\eta_1)T^{1+\epsilon} \leq 2t\right) \\
&\qquad = 0. \qquad \text{(A.50)}
\end{aligned}$$

All that is left is to bound the second term in (A.48), and to do so we apply the following Lemma whose proof is in Appendix A.4.3

Lemma 23. *There exist positive constants $C = C(\theta_2, \eta_1)$ and $x' > \theta_2$ such that*

$$\sum_{t=1}^{T} \mathbb{P}\left(S_2(t) \geq \lfloor P_2(t)\eta_1 \rfloor,\ \pi_t^{\mathrm{OG}} = 2\right) \leq K + \frac{1}{1-e^{-d(x',\theta_2)}}$$

Combining Lemma 23, (A.50), (A.48) and (A.49) shows the claim. □

Proof of Lemma 22.

Proof. We upper bound the denominator as follows. Given that $s < \lfloor kx \rfloor$, we have $s \leq kx - 1$. Let $B(a,b)$ denote the Beta function for parameters $a, b > 0$, that is

$$B(a,b) \triangleq \int_0^1 t^{a-1}(1-t)^{b-1}\, dt,$$

which is used in the definition of the Beta CDF. We can derive an upper bound on the denominator in the following way. Namely, we have

$$\begin{aligned}
\mathbb{E}\left[(V-y)^+\right] &= \frac{1}{B(s+1,k-s+1)}\int_y^1 (t-y)t^s(1-t)^{k-s}\,dt \\
&= \frac{1}{B(s+1,k-s+1)}\int_y^1 t^{s+1}(1-t)^{k-s}\,dt - y\mathbb{P}\left(V \geq y\right) \\
&= \frac{B(s+2,k-s+1)}{B(s+1,j-s+1)}\left(\frac{1}{B(s+2,k-s+1)}\right)\int_y^1 t^{s+1}(1-t)^{k-s}\,dt \\
&\quad - y\mathbb{P}\left(V \geq y\right) \\
&= \frac{s+1}{k+2}F^B_{k+2,y}(s+1) - y\mathbb{P}\left(V \geq y\right) \qquad \text{(A.51)} \\
&\leq \frac{s+1}{k+2}F^B_{k+2,y}(s+1) \\
&\leq F^B_{k,y}(kx) \\
&\leq \exp\left\{-kd(x,y)\right\} \qquad \text{(A.52)}
\end{aligned}$$

where we use Fact 18 and the definition of the Beta CDF to establish equation (A.51). The final bound in (A.52) is the result of the Chernoff-Hoeffding theorem and Fact 20. Let $\delta := y - x$, and note that $s < kx \implies s \leq \lfloor (k+1)x \rfloor$ due to s being integer, whence

$$\begin{aligned}
\mathbb{E}\left[(y-V)^+\right] &= \mathbb{E}\left[(y-V)\mathbb{1}\left(V \leq y\right)\right] \\
&= \mathbb{E}\left[(y-V)\mathbb{1}\left(y-\delta \leq V \leq y\right)\right] + \mathbb{E}\left[(y-V)\mathbb{1}\left(V < y-\delta\right)\right] \\
&> \mathbb{E}\left[(y-V)\mathbb{1}\left(V < y-\delta\right)\right] \\
&\geq \delta\mathbb{E}\left[\mathbb{1}\left(V < y-\delta\right)\right] \qquad \text{(A.53)} \\
&= \delta\mathbb{P}\left(V < x\right) \\
&= \delta\left(1 - F^B_{k+1,x}(s)\right) \qquad \text{(A.54)} \\
&\geq \delta/2 \qquad \text{(A.55)}
\end{aligned}$$

where equation (A.54) relies on Fact 18. The bound (A.55) is justified from Fact 19 and $s \leq \lfloor (k+1)x \rfloor$. Thus using the inequalities for both the numerator and denomi-

nator, we obtain the desired bound. □

Proof of Lemma 23.

Proof. The steps in this proof follow a similar one in Agrawal and Goyal [2013] but we show them for completeness. We bound the number of times the sub-optimal arm's mean is overestimated. Let τ_ℓ be the time step in which the sub-optimal arm is sampled for the ℓ^{th} time. Because for any x, $\lim_{n\to\infty} \frac{\lfloor nx \rfloor}{nx} = 1$, we can let N be a large enough integer so that if $\ell \geq N$, then $\eta_1 \frac{\lfloor \ell\eta_1 \rfloor}{\ell\eta_1} > x' := (\theta_2 + \eta_1)/2 > \theta_2$. In that case,

$$\sum_{t=1}^{T} \mathbb{P}(S_2(t) \geq \lfloor P_2(t)\eta_1 \rfloor,\ \pi_t^{\text{OG}} = 2) \tag{A.56}$$

$$\begin{aligned}
&\leq \mathbb{E}\left[\sum_{\ell=1}^{T} \sum_{t=\tau_\ell}^{\tau_{\ell+1}-1} \mathbb{1}\left(S_2(\ell) \geq \lfloor P_2(\ell)\eta_1 \rfloor\right) \mathbb{1}\left(\pi_t^{\text{OG}} = 2\right)\right] \\
&= \mathbb{E}\left[\sum_{\ell=1}^{T} \mathbb{1}\left(S_2(\tau_\ell) \geq \lfloor (\ell-1)\eta_1 \rfloor\right) \sum_{t=\tau_\ell}^{\tau_{\ell+1}-1} \mathbb{1}\left(\pi_t^{\text{OG}} = 2\right)\right] \\
&= \mathbb{E}\left[\sum_{\ell=0}^{T-1} \mathbb{1}\left(S_2(\tau_{\ell+1}) \geq \lfloor \ell\eta_1 \rfloor\right)\right] \\
&\leq N + \sum_{\ell=N+1}^{T-1} \mathbb{P}\left(S_2(\tau_{\ell+1}) \geq \ell\eta_1 \frac{\lfloor \ell\eta_1 \rfloor}{\ell\eta_1}\right) \\
&\leq N + \sum_{\ell=N+1}^{T-1} \mathbb{P}\left(S_2(\tau_{\ell+1}) \geq \ell x'\right) \\
&\leq N + \sum_{\ell=1}^{\infty} \exp(-\ell d(x', \theta_2)) \qquad \text{(A.57)} \\
&= N + \frac{1}{1 - e^{-d(x',\theta_2)}}
\end{aligned}$$

where (A.57) follows from the Chernoff-Hoeffding theorem and the fact that $S_2(\tau_{\ell+1})$ is drawn from a Binomial$(P_2(\ell+1), \theta_2) \equiv$ Binomial(ℓ, θ_2) distribution. □

A.5 Further experiment results

A.5.1 Bayes UCB experiment

This experiment is motivated by Kaufmann et al. [2012] and in it we simulate the Bernoulli bandit problem with a $T = 500$ and two arms. Since we are interested in measuring expected regret over the prior, we draw the arms' mean rewards at random from the uniform distribution. There are 5,000 independent trials and we show the results in Figures A-2. OGI offers notable performance improvements over both Thompson Sampling and IDS for this modest horizon.

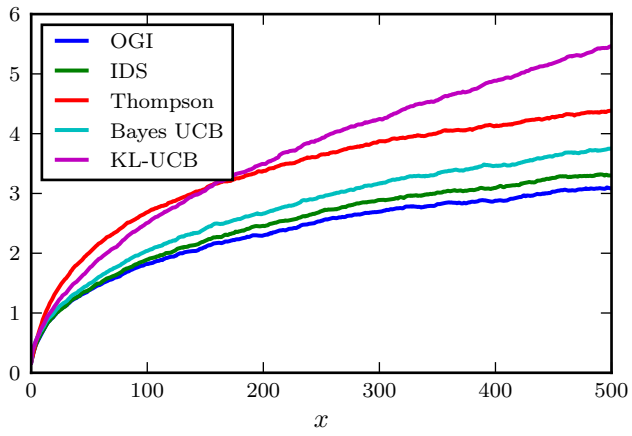

Figure A-2: Frequentist regret. The OGI policy is configured $K = 1$ and $\alpha = 100$.

A.5.2 Additional tables for Section 3.4

α	β	OGI(1)	OGI(3)	OGI(5)	Gittins
1	1	0.760	0.721	0.712	0.703
1	2	0.571	0.522	0.511	0.500
1	3	0.452	0.401	0.389	0.380
1	4	0.374	0.321	0.312	0.302
2	1	0.853	0.818	0.809	0.800
2	2	0.702	0.657	0.646	0.635
2	3	0.591	0.543	0.530	0.516
2	4	0.508	0.458	0.445	0.434
3	1	0.893	0.864	0.855	0.845
3	2	0.771	0.729	0.719	0.707
3	3	0.671	0.626	0.613	0.601
3	4	0.592	0.545	0.532	0.518
4	1	0.916	0.890	0.882	0.872
4	2	0.813	0.776	0.765	0.754
4	3	0.724	0.682	0.670	0.658
4	4	0.651	0.607	0.593	0.581

Table A.1: Optimistic and exact Gittins Indices when $\gamma = 0.9$ for different Beta-Bernoulli parameters

α	β	OGI(1)	OGI(3)	OGI(5)	Gittins
1.0	1.0	0.817	0.784	0.774	0.761
1.0	2.0	0.637	0.590	0.577	0.560
1.0	3.0	0.514	0.463	0.449	0.433
1.0	4.0	0.430	0.376	0.364	0.348
2.0	1.0	0.890	0.860	0.851	0.838
2.0	2.0	0.752	0.710	0.698	0.681
2.0	3.0	0.643	0.596	0.581	0.562
2.0	4.0	0.558	0.509	0.494	0.475
3.0	1.0	0.921	0.896	0.887	0.874
3.0	2.0	0.811	0.773	0.762	0.744
3.0	3.0	0.715	0.672	0.658	0.639
3.0	4.0	0.637	0.591	0.575	0.556
4.0	1.0	0.938	0.916	0.908	0.895
4.0	2.0	0.847	0.812	0.801	0.784
4.0	3.0	0.763	0.722	0.709	0.690
4.0	4.0	0.691	0.648	0.633	0.613

Table A.2: Optimistic and exact Gittins Indices when $\gamma = 0.95$ for different Beta-Bernoulli parameters

Appendix B

Supplementary Material for Collateral Management

B.1 Proofs

B.1.1 Proof of Lemma 8

Proof. For the first part of the proof, we need to show that $L(\lambda; x)$ is concave in its argument. Now because

$$L(\lambda; x) = \sum_{t=1}^{T} \lambda^\top \bar{\delta}_t - \sum_{t=1}^{T} \mathbb{E}\left[\sup_{u_t \in U(\xi_t)} \lambda^\top u_t\right] - \lambda^\top x$$

we need to check that $\mathbb{E}\left[\sup_{u_t \in U(\xi_t)} \lambda^\top u_t\right]$ is convex in λ, but this is true since this is a supremum over linear functions of λ.

Now to show the second result we will derive the KKT conditions for (3.18). First of all we introduce K-dimensional Lagrange multipliers $\mu_1 \geq 0$, corresponding to constraint $\lambda \leq c$, and $\mu_2 \geq 0$ for the non-negativity constraints $\lambda \geq 0$. Next we form the Lagrangian for this problem which is

$$\ell(\lambda, \mu_1, \mu_2; x) \triangleq L(\lambda; x) + (c - \lambda)^\top \mu_1 + \lambda^\top \mu_2.$$

Therefore, we obtain that

$$\begin{aligned}\partial_\lambda \ell(\lambda, \mu_2, \mu_2; x) &= \partial_\lambda L(\lambda; x) - \mu_1 + \mu_2 \\ &= \partial_\lambda \left(\sum_{t=1}^{T} \lambda^\top \bar{\delta}_t - \sum_{t=1}^{T} \mathbb{E}\left[\sup_{u_t \in U(\xi_t)} \lambda^\top u_t \right] - \lambda^\top x \right) - \mu_1 + \mu_2 \\ &= \sum_{t=1}^{T} \bar{\delta}_t - \sum_{t=1}^{T} \partial_\lambda \mathbb{E}\left[\sup_{u_t \in U(\xi_t)} \lambda^\top u_t \right] - x - \mu_1 + \mu_2\end{aligned}$$

These observations are enough to put together the KKT conditions. These are that λ^* is an optimal solution to the problem if and only if (due to convexity) all of the following conditions hold true:

$$0 \in \sum_{t=1}^{T} \bar{\delta}_t - \sum_{t=1}^{T} \partial_{\lambda^*} \mathbb{E}\left[\sup_{u_t \in U(\xi_t)} (\lambda^*) \top u_t \right] - x - \mu_1 + \mu_2 \tag{B.1}$$

$$(c - \lambda^*)^\top \mu_1 = 0 \tag{B.2}$$

$$(\lambda^*)^\top \mu_2 = 0 \tag{B.3}$$

$$\mu_1, \mu_2 \geq 0 \tag{B.4}$$

$$0 \leq \lambda \leq c. \tag{B.5}$$

Thus if λ^* is an optimal solution, by condition (B.1), there exist sub-gradient vectors $\bar{u}_t \in \partial_\lambda \mathbb{E}\left[\sup_{u_t \in U(\xi_t)} \lambda^\top u_t\right]$ such that

$$\sum_{t=1}^{T} \bar{\delta}_t - \sum_{t=1}^{T} \bar{u}_t - x - \mu_1 + \mu_2 = 0. \tag{B.6}$$

We prove the first statement. Let us suppose that for an arbitrary asset k, the optimal component of λ^* for that asset satisfies $\lambda_k^* < c_k$. We now show that the first Lagrange multiplier corresponding to that asset is zero, namely $\mu_{k,1} = 0$. This follows simply from observing, from KKT condition (B.2), that

$$0 = \sum_{k=1}^{K} (c_k - \lambda_k^*) \mu_{k,1}$$

and that $c_k - \lambda_k^* \geq 0$, $\mu_1 \geq 0$ for all k (which follows from the feasibility conditions (B.4) and (B.5)). The above observation along with the stationarity condition, given by (B.6), imply that

$$\sum_{t=1}^{T} \bar{\delta}_{k,t} - \sum_{t=1}^{T} \bar{u}_{k,t} - x_k = -\mu_{k,2} \leq 0$$

where the inequality follows from condition (B.4).

To show the second statement, assume that $\sum_{t=1}^{T} \bar{\delta}_{k,t} < \sum_{t=1}^{T} \bar{u}_{k,t} + x_k$. Then the stationarity condition (B.6) implies that $\mu_{k,1} < \mu_{k,2}$, which in turn gives us that the variable $\mu_{k,2}$ is positive. Finally, from this and the complementary slackness condition (B.3), we conclude that $\lambda_k^* = 0$. □

Proof of Proposition 3

Proof. First fix λ and ξ_t and note that for any point u_t^* that achieves the supremum over $\sup_{u \in U(\xi)} \lambda^\top u$, we have

$$u_t^* \in \partial_\lambda \left(\sup_{u_t \in U(\xi_t)} \lambda^\top u_t \right).$$

However, from the uniqueness and existence of the point u_t^* (Assumption 1), we have that $u_t^* = u_t^*(\xi; \lambda)$. Thus, we can write this statement as

$$\partial_\lambda \left(\sup_{u_t \in U(\xi_t)} \lambda^\top u_t \right) = \{u_t^*(\xi; \lambda)\} \tag{B.7}$$

where we can now, intuitively, simply take the gradient with respect to λ, since the gradient is precisely the only element of a singleton sub-differential. Therefore, because $U(\xi)$ is a bounded convex set and $|\sup_{\lambda, u \in U(\xi)} \lambda^\top u| < \infty$, we can take expectations on both sides of Equation B.7 (and apply the monotone convergence theorem in order to interchange gradient and expectation) to derive (3.20). Finally (3.21) follows from Lemma 8 and (3.20). □

Proof of Lemma

Proof. The proof boils down to showing that $z_t^{S,\lambda^*} = (\Delta_t - x_t)^+$. However it's enough to show that for all t, $x_t = x - \sum_{s=1}^{t-1} \delta_s + \sum_{s=1}^{t-1} z_s^{S,\lambda^*} + \sum_{s=1}^{t-1} u_s^{S,\lambda^*}$, which can be done by induction. The base case is immediate. Now, let us assume that the equation holds for t and show it also holds for $t+1$:

$$\begin{aligned}
x_{t+1} &= (x_t - \Delta_t)^+ \\
&= \left(x - \sum_{s=1}^{t-1} \delta_s + \sum_{s=1}^{t-1} z_s^{S,\lambda^*} + \sum_{s=1}^{t-1} u_s^{S,\lambda^*} - \Delta_t \right)^+ \\
&= \left(x - \sum_{s=1}^{t} \delta_s + \sum_{s=1}^{t-1} z_s^{S,\lambda^*} + \sum_{s=1}^{t} u_s^{S,\lambda^*} \right)^+ \\
&= x - \sum_{s=1}^{t} \delta_s + \sum_{s=1}^{t-1} z_s^{S,\lambda^*} + \sum_{s=1}^{t} u_s^{S,\lambda^*} \\
&\quad - \underbrace{\left(\sum_{s=1}^{t} \delta_s - \sum_{s=1}^{t-1} z_s^{S,\lambda^*} - \sum_{s=1}^{t} u_s^{S,\lambda^*} - x \right)^+}_{z_t^{S,\lambda^*}} \\
&= x - \sum_{s=1}^{t} \delta_s + \sum_{s=1}^{t} z_s^{S,\lambda^*} + \sum_{s=1}^{t} u_s^{S,\lambda^*}.
\end{aligned}$$

Finally, we complete the proof, by letting t be arbitrary and showing that

$$\begin{aligned}
(\Delta_t - x_t)^+ &= \left(\Delta_t + \sum_{s=1}^{t-1} \delta_s - \sum_{s=1}^{t-1} z_s^{S,\lambda^*} - \sum_{s=1}^{t-1} u_s^{S,\lambda^*} - x \right)^+ \\
&= \left(\sum_{s=1}^{t} \delta_s - \sum_{s=1}^{t-1} z_s^{S,\lambda^*} - \sum_{s=1}^{t} u_s^{S,\lambda^*} - x \right)^+ \\
&= z_t^{S,\lambda^*}.
\end{aligned}$$

□

Proof of Lemma 10

Proof. First note that for any $t \geq 1$ and k

$$\Delta_{k,t} - x_{k,t} = (\Delta_{k,t} - x_{k,t})^+ - (x_{k,t} - \Delta_{k,t})^+$$

which, since $x_{k,t+1} = (x_{k,t} - \Delta_{k,t})^+$, implies that

$$(\Delta_{k,t} - x_{k,t})^+ = x_{k,t+1} - x_{k,t} + \Delta_{k,t}.$$

We therefore have through a telescoping sum that

$$\begin{aligned}
\sum_{t=1}^{T} \mathbb{E}\left[(\Delta_{k,t} - x_{k,t})^+\right] &= \sum_{t=1}^{T} \mathbb{E}\left[\Delta_{k,t}\right] + \sum_{t=1}^{T} \mathbb{E}\left[x_{k,t+1} - x_{k,t}\right] \\
&= \sum_{t=1}^{T} \mathbb{E}\left[\Delta_{k,t}\right] + \mathbb{E}\left[x_{k,T+1}\right] - \mathbb{E}\left[x_{k,1}\right] \\
&= \sum_{t=1}^{T} \mathbb{E}\left[\Delta_{k,t}\right] + \mathbb{E}\left[x_{k,T+1}\right] - x_k
\end{aligned}$$

where we also used that $x_{k,1} = x_k$ almost surely. □

Bibliography

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

Rajeev Agrawal. Sample mean based index policies with $\mathcal{O}(\log n)$ regret for the Multi-armed Bandit problem. *Advances in Applied Probability*, pages 1054–1078, 1995.

Shipra Agrawal and Navin Goyal. Analysis of Thompson Sampling for the Multi-armed Bandit problem. In *COLT*, 2012.

Shipra Agrawal and Navin Goyal. Further Optimal Regret Bounds for Thompson Sampling. In *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics*, pages 99–107, 2013.

Jean-Yves Audibert and Sébastien Bubeck. Regret bounds and minimax policies under partial monitoring. *Journal of Machine Learning Research*, 11(Oct):2785–2836, 2010.

Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the Multi-armed Bandit problem. *Machine learning*, 47(2-3):235–256, 2002.

Alberto Bemporad and Carlo Filippi. An algorithm for approximate multiparametric convex programming. *Computational optimization and applications*, 35(1):87–108, 2006.

Alberto Bemporad, Francesco Borrelli, and Manfred Morari. Min-max control of constrained uncertain discrete-time linear systems. *IEEE Transactions on automatic control*, 48(9):1600–1606, 2003.

Aharon Ben-Tal, Laurent El Ghaoui, and Arkadi Nemirovski. *Robust optimization*. Princeton University Press, 2009.

Donald A Berry and Bert Fristedt. *Bandit problems: sequential allocation of experiments (Monographs on statistics and applied probability)*. Springer, 1985.

Dimitri P Bertsekas and Sergey Ioffe. Temporal differences-based policy iteration and applications in neuro-dynamic programming. *Lab. for Info. and Decision Systems Report LIDS-P-2349, MIT, Cambridge, MA*, 1996.

Dimitri P Bertsekas and John N Tsitsiklis. Neuro-dynamic programming: an overview. In *Decision and Control, 1995., Proceedings of the 34th IEEE Conference on*, volume 1, pages 560–564. IEEE, 1995.

Dimitris Bertsimas and Andrew W Lo. Optimal control of execution costs. *Journal of Financial Markets*, 1(1):1–50, 1998.

Dimitris Bertsimas and José Niño-Mora. Conservation laws, extended polymatroids and Multi-armed Bandit problems; a polyhedral approach to indexable systems. *Mathematics of Operations Research*, 21(2):257–306, 1996.

Dimitris Bertsimas and Melvyn Sim. The price of robustness. *Operations research*, 52(1):35–53, 2004.

Dimitris Bertsimas and John N Tsitsiklis. *Introduction to linear optimization*, volume 6. Athena Scientific Belmont, MA, 1997.

Dimitris Bertsimas, David B Brown, and Constantine Caramanis. Theory and applications of robust optimization. *SIAM review*, 53(3):464–501, 2011.

Russell N Bradt, SM Johnson, and Samuel Karlin. On sequential designs for maximizing the sum of n observations. *The Annals of Mathematical Statistics*, 27(4): 1060–1074, 1956.

Jhelum Chakravorty and Aditya Mahajan. Multi-armed Bandits, Gittins index, and its calculation. *Methods and Applications of Statistics in Clinical Trials: Planning, Analysis, and Inferential Methods*, 2:416–435, 2013.

Olivier Chapelle and Lihong Li. An empirical evaluation of Thompson Sampling. In *Advances in neural information processing systems*, pages 2249–2257, 2011.

Thomas M Cover and Joy A Thomas. *Elements of Information Theory*. John Wiley & Sons, 2012.

George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

Daniela Pucci De Farias and Benjamin Van Roy. The linear programming approach to approximate dynamic programming. *Operations research*, 51(6):850–865, 2003.

Daniela Pucci De Farias and Benjamin Van Roy. On constraint sampling in the linear programming approach to approximate dynamic programming. *Mathematics of operations research*, 29(3):462–478, 2004.

Vijay V Desai, Vivek F Farias, and Ciamac C Moallemi. Pathwise optimization for optimal stopping problems. *Management Science*, 58(12):2292–2308, 2012.

John Duchi, Shai Shalev-Shwartz, Yoram Singer, and Tushar Chandra. Efficient projections onto the l 1-ball for learning in high dimensions. In *Proceedings of the 25th international conference on Machine learning*, pages 272–279. ACM, 2008.

Richard Evans and Jim Gao. Deepmind ai reduces google data centre cooling bill by 40%. *DeepMind blog*, 20, 2016.

Maryam Fazel, Rong Ge, Sham M Kakade, and Mehran Mesbahi. Global convergence of policy gradient methods for linearized control problems. *arXiv preprint arXiv:1801.05039*, 2018.

Carlos E Garcia, David M Prett, and Manfred Morari. Model predictive control: theory and practiceâĂŤa survey. *Automatica*, 25(3):335–348, 1989.

Aurélien Garivier. The KL-UCB algorithm for bounded stochastic bandits and beyond. In *COLT*, 2011.

Nicolae Gârleanu and Lasse Heje Pedersen. Dynamic trading with predictable returns and transaction costs. *The Journal of Finance*, 68(6):2309–2340, 2013.

John C Gittins. Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 148–177, 1979.

Paul Glasserman and Xingbo Xu. Robust portfolio control with stochastic factor dynamics. *Operations Research*, 61(4):874–893, 2013.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

Shixiang Gu, Timothy Lillicrap, Ilya Sutskever, and Sergey Levine. Continuous deep q-learning with model-based acceleration. In *International Conference on Machine Learning*, pages 2829–2838, 2016.

Martin B Haugh and Leonid Kogan. Pricing american options: a duality approach. *Operations Research*, 52(2):258–270, 2004.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. *arXiv preprint arXiv:1709.06560*, 2017.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Kumar Jogdeo and Stephen M Samuels. Monotone convergence of binomial probabilities and a generalization of Ramanujan's equation. *The Annals of Mathematical Statistics*, pages 1191–1195, 1968.

Sham M Kakade. A natural policy gradient. In *Advances in neural information processing systems*, pages 1531–1538, 2002.

Michael N Katehakis and Herbert Robbins. Sequential choice from several populations. *Proceedings of the National Academy of Sciences of the United States of America*, 92(19):8584, 1995.

Michael N Katehakis and Arthur F Veinott Jr. The Multi-armed Bandit problem: Decomposition and computation. *Mathematics of Operations Research*, 12(2):262–268, 1987.

Michael N Katehakis, Uriel G Rothblum, et al. Finite state Multi-armed Bandit problems: Sensitive-discount, average-reward and average-overtaking optimality. *The Annals of Applied Probability*, 6(3):1024–1034, 1996.

Emilie Kaufmann. On Bayesian index policies for sequential resource allocation. *arXiv preprint arXiv:1601.01190*, 2016.

Emilie Kaufmann, Nathaniel Korda, and Rémi Munos. Thompson Sampling: An asymptotically optimal finite-time analysis. In *Algorithmic Learning Theory*, pages 199–213. Springer, 2012.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Nathaniel Korda, Emilie Kaufmann, and Rémi Munos. Thompson sampling for 1-dimensional exponential family bandits. In *Advances in Neural Information Processing Systems*, pages 1448–1456, 2013.

Tze Leung Lai. Adaptive treatment allocation and the Multi-armed Bandit problem. *The Annals of Statistics*, pages 1091–1114, 1987.

Tze Leung Lai and Herbert Robbins. Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, 6(1):4–22, 1985.

Tor Lattimore. Regret analysis of the finite-horizon Gittins index strategy for Multi-armed Bandits. In *Conference on Learning Theory*, pages 1214–1245, 2016.

Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

Francis A Longstaff and Eduardo S Schwartz. Valuing american options by simulation: a simple least-squares approach. *The review of financial studies*, 14(1):113–147, 2001.

Odalric-Ambrym Maillard, Rémi Munos, Gilles Stoltz, et al. A Finite-Time Analysis of Multi-armed Bandits problems with Kullback-Leibler Divergences. In *COLT*, pages 497–514, 2011.

Hongzi Mao, Mohammad Alizadeh, Ishai Menache, and Srikanth Kandula. Resource management with deep reinforcement learning. November 2016. URL `https://www.microsoft.com/en-us/research/publication/resource-management-deep-reinforcement-learning/`.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Ciamac Moallemi and Mehmet Sağlam. Dynamic portfolio choice with linear rebalancing rules. 2015.

Warren B Powell and Ilya O Ryzhov. *Optimal Learning*, volume 841. John Wiley & Sons, 2012.

Herbert Robbins. Some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society*, 58(5):527–535, 1952.

Leonard CG Rogers. Monte carlo valuation of american options. *Mathematical Finance*, 12(3):271–286, 2002.

Dan Russo and Benjamin Van Roy. Learning to optimize via Information-Directed Sampling. In *Advances in Neural Information Processing Systems*, pages 1583–1591, 2014.

Steven L Scott. A modern Bayesian look at the Multi-armed Bandit. *Applied Stochastic Models in Business and Industry*, 26(6):639–658, 2010.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063, 2000.

William R Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, pages 285–294, 1933.

John N Tsitsiklis. A short proof of the Gittins index theorem. *The Annals of Applied Probability*, pages 194–199, 1994.

John N Tsitsiklis and Benjamin Van Roy. Optimal stopping of markov processes: Hilbert space theory, approximation algorithms, and an application to pricing high-dimensional financial derivatives. *IEEE Transactions on Automatic Control*, 44(10): 1840–1851, 1999.

Pravin Varaiya, Jean Walrand, and Cagatay Buyukkoc. Extensions of the Multi-armed Bandit problem: The discounted case. *IEEE transactions on automatic control*, 30(5):426–439, 1985.

Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4): 279–292, 1992.

Richard Weber et al. On the Gittins index for Multi-armed Bandits. *The Annals of Applied Probability*, 2(4):1024–1033, 1992.

Peter Whittle. Multi-armed Bandits and the Gittins index. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 143–149, 1980.

Peter Whittle. Restless Bandits: Activity allocation in a changing world. *Journal of applied probability*, pages 287–298, 1988.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Reinforcement Learning*, pages 5–32. Springer, 1992.

Yi-Ching Yao et al. Some results on the Gittins index for a normal reward process. In *Time Series and Related Topics*, pages 284–294. Institute of Mathematical Statistics, 2006.